ალგორითმები და მონაცემთა სტრუქტურები

ალექსანდრე გამყრელიძე

შესავალი

ჩვენს ყოველდღიურ ცხოვრებაში ალგორითმები უაღრესად დიდ როლს თამაშობენ ისე, რომ ჩვენ ამას ვერც კი ვამჩნევთ. უფრო მეტიც, ბევრმა არც კი იცის, თუ რა არის ალგორითმი. არა და ალგორითმები ყოველ ფეხის ნაბიჯზე გხვდება, ამ სიტყვის პირდაპირი მნიშვნელობით -- ადამიანის სიარული გარკვეული თვალსაზრისით ალგორითმია: მარცხენა ფეხი გადადგი წინ, ტანი გადახარე ოდნავ წინ, მარჯვენა ფეხი გადაგი წინ და ეს პროცესი თავიდან გაიმეორე მანამ, სანამ სიარულის შეწყვეტა მოგინდება. სხვათა შორის, ეს ცენტრალური ალგორითმია რობოტოტექნიკაში და დღეისათვის ბოლომდე განხორციელებული არ არის -- როგორც აღმოჩნდა, ასეთი ერთი შეხედვით მარტივი ალგორითმის რეალიზაცია ძალიან რთულია.

მეორე მაგალითად ფშავური ხინკლის გაკეთების ალგორითმი შეიძლება მოვიყვანოთ:

<u>მონაცემები:</u>

ხორცი, ხახვი, რეჰანი, ქონდარი, წითელი წიწაკა, პილპილი, მარილი, ფქვილი

<u>ალგორითმის მუშაობის შედეგი:</u> ფშავური ხინკალი

<u>ალგორითმის მუშაობის აღწერა:</u>

ალგორითმი „ფშავური ხინკალი"

მონაცემები: ხორცი, ხახვი, რეჰანი, ქონდარი, წითელი წიწაკა, პილპილი, მარილი, ფქვილი, წყალი

1. გააკეთე ბულიონი: ძვლები ჩაყარე ქვაბში, დაასხი იმდენი წყალი, რომ დაიფაროს და ნელ ცეცხლზე ადუღე. როცა გასინჯავ და უკვე წყალ-წყლა აღარ იქნება, გადმოდგი და გაატარე წვრილ ბადეში, რომ ძვლების ნარჩენები არ შეყვეს. ამის შემდეგ გააცივე და გვერდზე გადადგი.
2. ხორცი, წიწაკა, ხახვი, რეჰანი და ქონდარი ცალ-ცალკე წვრილდ აკეპე.
3. ხორცს დაასხი მარილითა და წიწაკით გაზავებული ნელ-თბილი ბულიონი და აზილე. შემდეგ კიდევ დაასხი და აზილე. ეს პროცედურა გაიმეორე მანამ, სანამ ბულიონს არ შეიწოვს და თავზე კიდევ ცოტა არ დადგება.
4. არსებულ ფარშს შეურიე დარჩენილი ხახვი, წიწაკა, პილპილი და მწვანილი (გემოვნებით).
5. შემდეგ აიღე ზუსტად იმდენივე ბულიონი, რამდენიც დაჭირდა ხორცს და შეურიე მარილი ისე, რომ სიმლაშე საკმაოდ ეტყობოდეს. ამ ბულიონით მოზილე საკმაოდ მაგარი ცომი.
6. დაადგი ბევრი წყალი ძალიან მაღალ ცეცხლზე.
7. ცომიდან ჩამოჭერი მოგრძო ნაჭერი, თოკივით დაამრგვალე და დაჭერი პატარა ნაჭრებად. ეს ნაჭრები ცალ-ცალკე გააბრტყელე თხელ, მრგვალ დისკებად. კოვზით აიღე ფარში, ცომის დისკებზე დადე და გაახვიე.
8. შემდეგ ჩაყარე მდუღარე, მარილიან წყალში და დაახლოებით 10წთ. ხარშე.

ალგორითმი დასრულებულია

ზემოთ მოყვანილ ხინკლის ალგორითმში შემდეგი რამ არის გასათვალისწინებელი: „ცომის მოზელვის" პროცესი თავის მხრივ ალგორითმია, რომელიც პერიოდულად უნდა გაგრძელდეს მანამ, სანამ ცომი სასურველ კონსისტენციას არ მიაღწევს (ასეთივე რამ შეიძლება ითქვას ხორცის აკეპვის პროცედურაზეც). ესე იგი, აქ ჩართულია კიდევ შემოწმების მექანიზმი: თუ კონსისტენცია კარგია, მაშინ ალგორითმი დაასრულე. თუ არა, იგივე გაიმეორე.

ზოგადად, ალგორითმი რაიმე ამოცანის გადაჭრის გზაა, მაგრამ ამ გადაჭრისას უნდა გავითვალისწინოთ შემდეგი სამი პუნქტი:

1. ალგორითმი უნდა შედგებოდეს ერთი ან რამოდენიმე ბიჯისაგან;
2. როცა ალგორითმი ერთი ბიჯის შესრულებას დაასრულებს, იგი შემდგომი ბიჯის შესრულებაზე უნდა გადავიდეს;
3. ბიჯები შეიძლება პერიოდულად გამეორდეს, მაგრამ საერთო ჯამში ყოველი ალგორითმის ბიჯების საერთო რაოდენობა სასრული უნდა იყოს -- ალგორითმი როდესღაც უნდა გაჩერდეს.

ალგორითმებში მნიშვნელოვანია ორი ასპექტი:

1. სისწორე -- ეს ალგორითმი მართლა იმას აკეთებს, რაც მოეთხოვება?
2. სისწრაფე -- რამდენ ბიჯს ანდომებს ალგორითმი დაწყებიდან დამთავრებამდე?

ჩვენს ირგვლივ ძალიან ბევრი ამოცანა არსებობს: ხინკლის მოხარშვიდან დაწყებული და კოსმოსში რაკეტების გაგზავნით დამთავრებული. ბუნებრივად წამოიჭრა შეკითხვა: შეიძლება თუ არა ყველა ამოცანა ალგორითმულად გადაიჭრას? როგორც აღმოჩნდა, არსებობს ისეთ ამოცანათა სიმრავლე, რომლებსაც ალგორითმულად ვერ ამოვხსნით. უფრო მეტიც -- გაცილებით მეტია ისეთი ამოცანები, რომლებსაც ალგორითმულად ვერ ამოვხსნით, ვიდრე ისეთები, რომლებსაც შეიძლება მოვუგონოთ ალგორითმი. ეს კი იმას ნიშნავს, რომ ადამიანის ცხოვრებაში გაცილებით მეტი რამ არის ისეთი, რომელსაც კომპიტერი ვერ ამოხსნის, ვიდრე ისეთი, რომელსაც „ხელოვნური ინტელექტი" დაძლევს.

როგორც აღმოჩნდა, ალგორითმულად ამოხსნად ამოცანებს შორისაც არსებობს ისეთი ამოცანები, რომელთა დღეისათვის ცნობილი ალგორითმებით ამოხსნაც ძალიან დიდ დროს მოითხოვს, ანუ უმეტეს შემთხვევებში ჩვენს ხელთ არსებული უძლიერესი გამომთვლელი მანქანებით ასობით ათას წელს მოანდომებდა -- ბიჯების რაოდენობა ძალიან სწრაფად იზრდება. მაგრამ მთავარი აქ ისაა, რომ არ არის ცნობილი, შეიძლება თუ არა ასეთი ამოცანებისათვის დაიწეროს ისეთი ალგორითმი, რომელიც უფრო სწრაფი იქნებოდა.

როდესაც წამოიჭრება ახალი ამოცანა, პირველ რიგში უნდა დავადგინოთ, შეიძლება თუ არა მისი ალგორითმულად ამოხსნა. თუ არ შეიძლება, მაშინ უნდა დავადგინოთ, როგორ შევცვალოთ ამ ამოცანის პირობები ისე, რომ იგი ამოხსნადი გახდეს და, ამავდროულად, რაც შეიძლება ახლოს იყოს ამ დასმულ ამოცანასთან.
თუ ამოცანა ამოხსნადია, უნდა დავადგინოთ, შეიძლება თუ არა მისი სწრაფად ამოხსნა? თუ არ შჰეიძლება, მაშინ უნდა დავადგინოთ, როგორ შევცვალოთ ამ ამოცანის პირობები ისე, რომ იგი ამოხსნადი გახდეს და, ამავდროულად, რაც შეიძლება ახლოს იყოს ამ დასმულ ამოცანასთან (ევრისტიკების შექმნა) ან ისეთი სწრაფი ალგორითმი შევქმნათ, რომელიც ზუსტად იმავე მონაცემებზე და პირობებში ზუსტ პასუხთან მიახლოვებულ პასუხს მოგვცემს (მიახლოებითი ალგორითმები).
მაგრამ თუ სწრაფი ალგორითმის შექმნა შესაძლებელია, როგორ შევქმნათ ოპტიმალური ალგორითმი, ანუ ისეთი, რომ მასზე სწრაფი ალგორითმი არ არსებობდეს.

ამ საკითხების გარკვევაში გვეხმარება თეორიული ინფორმატიკის ერთ-ერთი განხრა -- ალგორითმების თეორია, რომლის შესავალსაც ჩვენ აქ განვიხილავთ.

1 ალგორითმების მარტივი მაგალითები

1.1 მგელი, კურდღელი და სტაფილო

განვიხილოთ ბევრისათვის კარგად ცნობილი ამოცანა მგლის, კურდღლისა და სტაფილოს შესახებ (ეს ამოცანა უფრო კარგადაა ცნობილი, როგორც მგლის, ცხვრისა და კომბოსტოს ამოცანა):

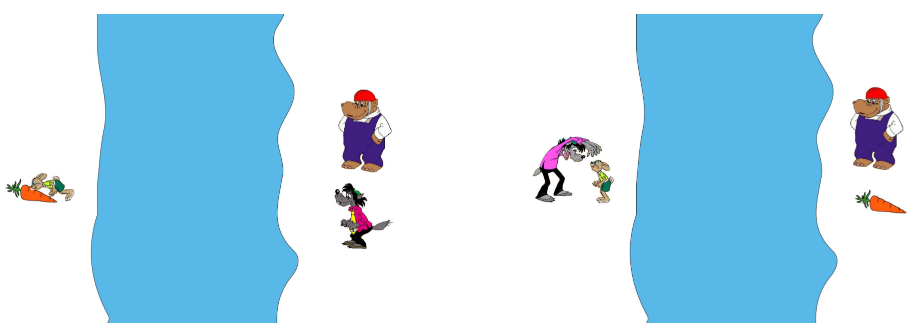
მდინარის ერთ ნაპირზე იმყოფებიან ბეჰემოტი, მგელი, კურდღელი და სტაფილო (ნახ.1). ბეჰემოტს აქვს ნავი, რომელშიც ეტევა მხოლოდ იგი და ერთი რომელიმე სხვა მგზავრი: მგელი, კურდღელი ან სტაფილო.

(ა) (ბ)

ნახ. 1:

სანამ ბეჰემოტი სხვა ცხოველებთან ერთადაა ნაპირზე, ისინი კარგად იქცევიან და ერთმანეთს არ დაერევიან. მაგრამ საკმარისია მან მარტო დატოვოს ერთ ნაპირზე კურდღელი და მგელი, რომ ეს უკანასკნელი კურდღელს ეტაკება. თვით კურდღელი კი მარტო დარჩენილ სტაფილოს შესჭამს.

თუ მგელი სტაფილოთი დარჩება ერთ ნაპირზე მარტო, არაფერი არ მოხდება.



ნახ. 2:

ამოცანა მდგომარეობს შემდეგში: დაწერეთ ალგორითმი, რომლის მეშვეობითაც ბეჰემოტი თავისი ნავით სამივეს გადაიყვანს მეორე ნაპირზე.

პირველ რიგში უნდა ჩამოვაყალიბოთ ამოცანა: მოცემულობა, საბოლოო შედეგი და ალგორითმის მსვლელობისას დადებული შეზღუდვები.

მოცემულია: მდინარე და მის ერთ ნაპირზე მყოფი ნავი, ბეჰემოტი, მგელი, კურდღელი და სტაფილო (ნახ. 1 (ა)).

შედეგი: ეს ყველა მეორე ნაპირზე ერთად მყოფი (ნახ. 1 (ბ)).

შეზღუდვა: ცხოველები გადაჰყავს ბეჰემოტს ორ ადგილიანი ნავით (პირველი შეზღუდვა -- ნავში უნდა იჯდეს ბეჰემოტი, რომელსაც მხოლოდ ერთი ადგილი რჩება თავისუფალი და, აქედან გამომდინარე, მეორე ნაპირზე ერთ ჯერზე შეუძლია გადაიყვანოს ან მხოლოდ მგელი, ან მხოლოდ კურდღელი, ან მხოლოდ სტაფილო). მგლისა და კუდღლის მარტო დატოვება არ შეიძლება, ასევე არ შეიძლება კურდღლისა და სტაფილოს მარტო დატოვება (მეორე და მესამე შეზღუდვა).

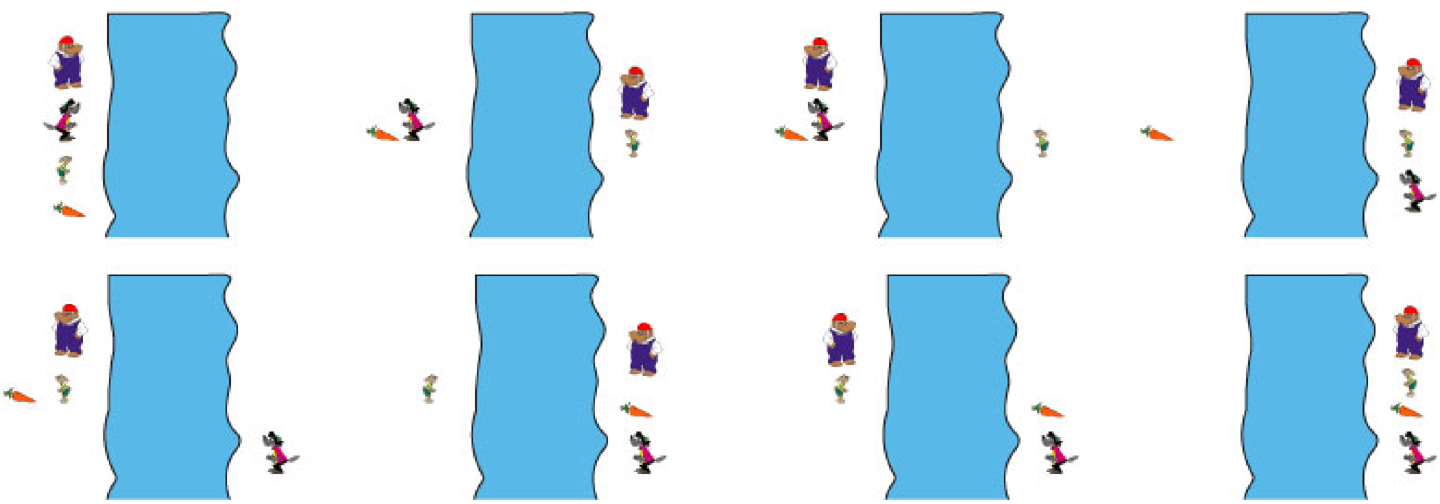
ამ ამოცანის ამოსახსნელად შეიძლება გამოვიყენოთ შემდეგი ალგორითმი, რომლის ყოველი ბიჯის ნახატი წარმოდგენილია 3-ში (დავუშვათ, რომ დასაწყისში ყველა მდინარის მარცხენა ნაპირზეა და ბოლოს მარჯვენა ნაპირზე უნდა იყოს):

ალგორითმი „მგელი, კურდღელი და სტაფილო”

მონაცემები: მდინარე და მის მარცხენა ნაპირზე განთავსებული ბეჰემოტი, მგელი, კურდღელი და სტაფილო;

1. მარჯვენა ნაპირზე გადაიყვანე კურდღელი ;
2. დაბრუნდი მარცხენა ნაპირზე ;
3. მარჯვენა ნაპირზე გადაიყვანე მგელი ;
4. მარცხენა ნაპირზე გადაიყვანე კურდღელი ;
5. მარჯვენა ნაპირზე გადაიტანე სტაფილო ;
6. დაბრუნდი მარცხენა ნაპირზე ;
7. მარჯვენა ნაპირზე გადაიყვანე კურდღელი .

ალგორითმი დასრულებულია



ნახ. 3: ალგორითმის თითოეული ბიჯი

პირველ რიგში უნდა დავამტკიცოთ ამ ალგორითმის სისწორე: რომ მისი საწყისი მონაცემებით გაშვებისას სასურველი შედეგი მიიღება და რომ ამ ალგორითმის მსვლელობისას ამოცანის არც ერთი პირობა არ ირღვევა (არ ხდება ისეთი რამ, რაც ზემოთ ჩამოთვლილ შეზღუდვებს დაარღვევდა).

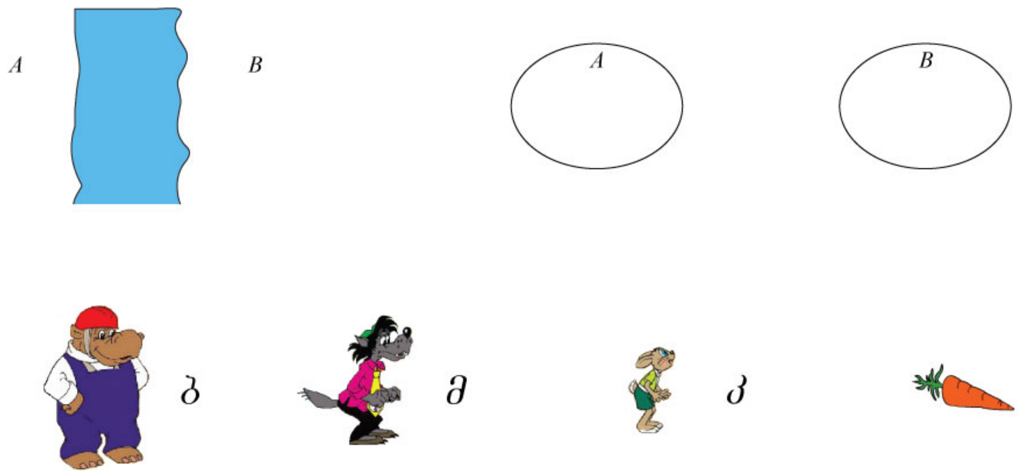
სავარჯიშო 1.1: დაამტკიცეთ ამ ალგორითმის სისწორე.

შემდეგ უნდა გამოვითვალოთ მისი სისწრაფე, ანუ რამდენ ბიჯს ანდომებს იგი დასაწყისიდან გაჩერებამდე.

სავარჯიშო 1.2: დაითვალეთ ამ ალგორითმის ბიჯების რაოდენობა.

როგორც წესი, ყოველდღიური ამოცანის დასმისას დიდი ინფორმაცია არ არის მნიშვნელოვანი. მაგალითად, არ არის საინტერესო, თუ რა ფორმისა ან სიგანისაა მდინარე, რა ფერისაა ნავი და ა.შ. ჩვენ გვაინტერესებს მხოლოდ ის ინფორმაცია, რომელიც ამოცანის პირობისთვისაა მნიშვნელოვანი. მაგალითად ის, რომ ერთ ჯერზე მხოლოდ ორი მგზავრი ეტევა ნავში და ერთ-ერთი მგზავრი აუცილებლად ბეჰემოტია. თუ ჩვენ მარცხენა ნაპირს

დავარქმევთ A, ხოლო მარჯვენას კი B, ეს ორი ნაპირი შეგვიძლია გამოვსახოთ ორი სიმრავლით, რომელსაც აგრეთვე სიმრავლე A და სიმრავლე B ეცოდება. ყოველ ცხოველს შევუსაბამებთ ერთ ასოს -- ბეჰემოტი $\Rightarrow$ ბ, მგელი $\Rightarrow$ მ, კურდღელი $\Rightarrow$ კ და სტაფილო $\Rightarrow$ ს (ნახ. 4).



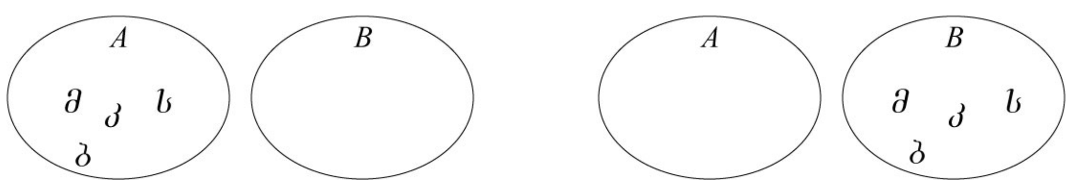
ნახ. 4:

მაშინ საწყისი და საბოლოო პირობები შემდეგნაირი იქნება (ნახ. 5). მათემატიკურ ენაზე კი დასმული ამოცანის პირობა ასე შეიძლება ჩამოყალიბდეს:

მოცემულია: ორი სიმრავლე $A =$ {ბ, მ, კ, ს} და $B = \emptyset$.

შედეგი: $A = \emptyset$ და $B =$ {ბ, მ, კ, ს}.

შეზღუდვა: ყოველ ჯერზე იმ სიმრავლიდან, რომელიც შეიცავს ასო „ბ", მეორე სიმრავლეში უნდა გადავიტანოთ ეს ასო და კიდევ ერთი ან ნული ასო. ის სიმრავლე, რომელიც **არ** შეიცავს ასო „ბ", **არ** უნდა შეიცავდეს ერთად ასოებს {მ, კ} და {კ, ს}.



ნახ. 5:

ამოცანის პირობა ოდნავ გამარტივდება, თუ ასოების ნაცვლად გარკვეულ რიცხვებს ავიღებთ: ბეჰემოტი $\Rightarrow$ 10, მგელი $\Rightarrow$ 1, კურდღელი $\Rightarrow$ 2 და სტაფილო $\Rightarrow$ 3. მაშინ კურდღლისა და სტაფილოს ან კურდღლისა და მგლის ერთ ნაპირზე ყოფნა იმას ნიშნავს, რომ შესაბამისი სიმრავლის ელემენტების ჯამი კენტია, ხოლო ის ფაქტი, რომ ბეჰემოტი რომელიგაცა ნაპირზე არ იმყოფება, იმას ნიშნავს, რომ შესაბამისი ელემენტების ჯამი ნაკლებია 10-ზე.

სავარჯიშო 1.3: ზემოთ ნახსენები ამოცანა ჩამოაყალიბეთ რიცხვებისათვის.

სავარჯიშო 1.4: წინა სავარჯიშოში ჩამოყალიბებული ამოცანისათვის დაწერეთ ალგორითმი და მისი ყოველი ბიჯისათვის შესაბამისი სიმრავლეები ჩამოწერეთ.

სავარჯიშო 1.5: დაამტკიცეთ წინა სავარჯიშოში დაწერილი ალგორითმის სისწორე და დაითვალეთ მისი ბიჯების რაოდენობა.

სავარჯიშო 1.6: განიხილეთ შემდეგი ალგორითმი:

ალგორითმი „მგელი, კურდღელი და სტაფილო" (სწრაფი ვერსია)

მონაცემები: მდინარე და მის მარცხენა ნაპირზე განთავსებული ბეჰემოტი, მგელი, კურდღელი და სტაფილო;

1. მარჯვენა ნაპირზე გადაიყვანე კურდღელი ;
2. დაბრუნდი მარცხენა ნაპირზე ;
3. მარჯვენა ნაპირზე გადაიყვანე მგელი ;
4. დაბრუნდი მარცხენა ნაპირზე ;
5. მარჯვენა ნაპირზე გადაიტანე სტაფილო .

ალგორითმი დასრულებულია

სავარჯიშო 1.7: მივიღებთ თუ არა ამ ალგორითმის მუშაობის შემდეგ იმ შედეგს, რომელიც ამოცანაშია მოთხოვნილი? არის თუ არა ეს ყველაზე სწრაფი ალგორითმი იმ ალგორითმთა შორის, რომელიც ამ ამოცანას ხსნის?

შენიშვნა: აქამდე ჩვენ განვიხილავდით შემთხვევას, როდესაც დასაწყისში ყველა მარცხენა ნაპირზე დგას. ზუსტად იგივე მსჯელობის ჩატარება შეიძლება იმ შემთხვევისათვის, როდესაც ყველა მარჯვენა ნაპირზე დგას. ამ შემთხვევისათვის ალგორითმი ანალოგიური იქნება. არც იმას აქვს მნიშვნელობა, თუ რა თანმიმდევრობით ჩამოვთვლით ცხოველებს მოცემულობაში. ეს ყოველთვის ასე არაა, როგორც შემდეგი მარტივი მაგალითი გვიჩვენებს:

მოცემულია ორი რიცხვი. გამოითვალეთ $\frac{\text{პირველი რიცხვი}}{\text{მეორე რიცხვი}}$.

ცხადია, რომ აქ გადამწყვეტი მნიშვნელობა აქვს რიცხვების თანმიმდევრობას.

სავარჯიშო 1.8: განვიხილოთ n მთელი რიცხვის ზრდადობით დალაგების ამოცანა. რა არის ამ ამოცანაში მოცემული? რა უნდა იყოს მისი საბოლოო შედეგი?

სავარჯიშო 1.9: მოიყვანეთ შემდეგი ამოცანის ალგორითმი: მოცემული 10 ცალი მთელი რიცხვისათვის დაითვალეთ კენტ რიცხვთა ჯამი. მინიშნება: ყოველ ბიჯზე უნდა შევამოწმოთ, არის თუ არა მოცემული რიცხვი კენტი.
რამდენ ბიჯს მოითხოვს ასეთი ლგორითმი? რიცხვის კენტობის შემოწმება და მიმატების ოპერაცია თითო-თითო ბიჯად ჩათვალეთ.
რა არის ამ ამოცანის მონაცემი? რა არის შედეგი? როგორია პირობაზე დადებული შეზღუდვა?

ამოცანა: ორი დიდი ხნის უნახავი მათემატიკოსი ერთმანეთს ხვდება. ერთი ეუბნება: მე სამი შვილი მყავს. ერთ რამეს გეტყვი და თუ გამოიცნობ მათ ასაკს: მათი ასაკის ნამრავლია 36.
მეორე ეუბნება: ვერ გამოვიცნობ, დამატებით სხვა პირობა მჭირდება. პირველი ეტყვის: მათი ასაკის ჯამი შენს წინ მდებარე სახლის ფანჯრების რაოდენობის ტოლია. მეორე შეხედავს სახლს და ეტყვის: ერთი დამატებითი პირობა კიდევ მჭირდება.
პირველი ეტყვის: უფროსს ლურჯი თვალები აქვს. ამით მეორე სამივეს ასაკს გამოიცნობს.

შეკითხვა: რამდენი წლის არიან შვილები?

2 ამოცანათა რეკურსიული აღწერა

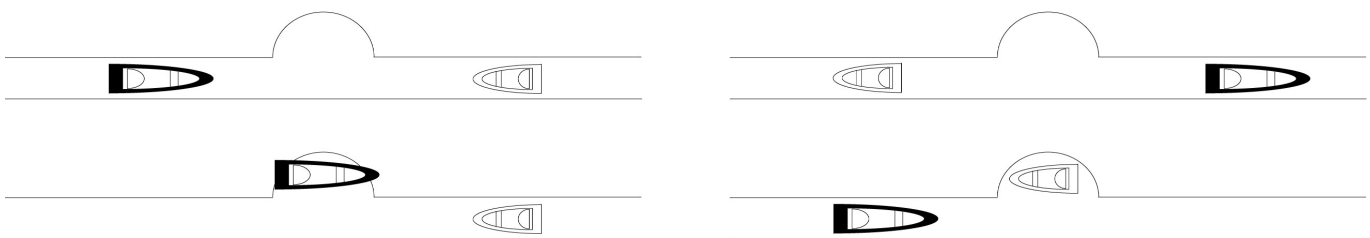
2.1 ამოცანა ნავების შესახებ

მოცემულია: ვიწრო მდინარე პატარა ყურეთი. მდინარეში ყურეს მარცხნივ გრძელი შავი ნავი და მარჯვნივ - მოკლე თეთრი ნავი.

შედეგი: მდინარეში ყურეს მარცხნივ მოკლე თეთრი ნავი და მარჯვნივ - გრძელი შავი ნავი (ნავებმა ერთმანეთს გვერდი უნდა აუქციონ).

შეზღუდვა: მდინარე იმდენად ვიწროა, რომ სიგანეში მხოლოდ ერთი ნავი ეტევა. ყურეში ეტევა მხოლოდ თეთრი ნავი. შავი ნავი ყურეში არ ეტევა.

ნახ. 6-ში გრაფიკულადაა ნაჩვენებია ამოცანის მონაცემი, შედეგი და შეზღუდვები.



ნახ. 6:

იმისათვის, რომ ერთმა თეტღმა ნავმა შავს გვერდი აუქციოს, საჭიროა შემდეგი ალგორითმის ჩატარება:

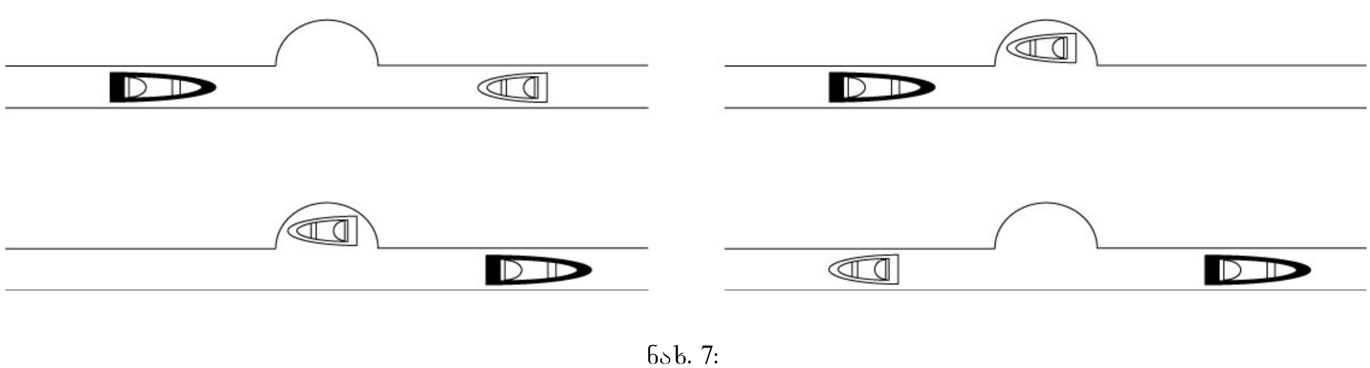
ალგორითმი „ერთი ნავის გაყვანა"

მონაცემები:
ვიწრო მდინარე პატარა ყურეთი, ყურეს მარცხნივ გრძელი შავი ნავი და მარჯვნივ - მოკლე თეთრი ნავი.

1. თეთრი ნავი შევიდეს ყურეში;
2. შავმა ნავმე გაიაროს;
3. თეთრი ნავი გამოვიდეს ყურედან.

ალგორითმი დასრულებულია

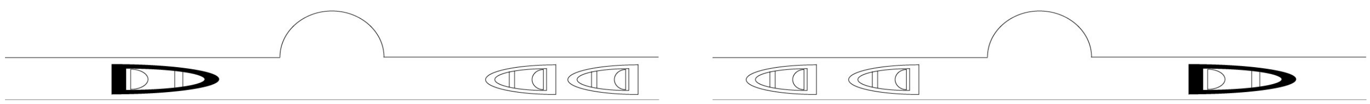
ადვილი საჩვენებელია, რომ ეს ალგორითმი ამოცანის საბოლოო შედეგს მოგვცემს და მისი არც ერთი ბიჯი ამოცანის შეზღუდვებს არ ეწინააღმდეგება (ნახ. 7).



ნახ. 7:

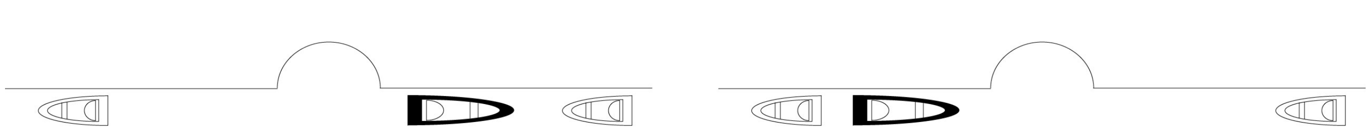
ზემოთ მოყვანილი ალგორითმი აღვნიშნოთ როგორც A_1. ესე იგი, თუ ვიტყვით, რომ ზემოთ მოყვანილ საწყის პირობაზე ჩატარებულია ალგორითმი A_1, შედეგად ვიღებთ ზემოთვე მოყვანილ საბოლოო შედეგს.

ახლა კი განვიხილოთ ისეთი შემთხვევა, როდესაც ყურეს მარჯვნივ არა ერთი, არამედ ორი ნავია განთავსებული. ნახ. 6-ში გრაფიკულადაა ნაჩვენებია ამ ამოცანის მონაცემი და შედეგი. ამ ამოცანას ჩვენ ვუწოდებთ „ორი ნავი".



ნახ. 8:

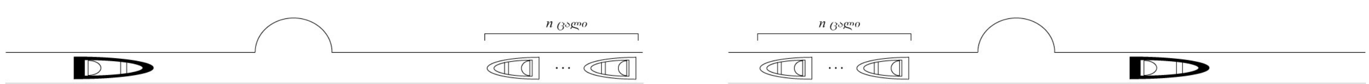
თუ პირველ რიგში ჩავატარებთ იგივე სამ ბიჯს, რაც ალგორითმში A_1, მივიღებთ ისეთ სიტუაციას, როგორიც ნაჩვენებია ნახ. 9-ში (მარცხნივ). შემდეგ, თუ შავი ნავი წავა უკან ყურეს მარცხნივ, შეიქმნება ისეთივე სიტუაცია, როგორც წინა ამოცანაში (ნახ. 9 მარჯვნივ)



ნახ. 9:

შავი ნავის უკან გასვლის პროცესი აღვნიშნოთ როგორც U. ესე იგი, თუ საწყისი მდგომარეობაა ისეთი, როგორც ნახ. 8-ში მარცხნივ და ჯერ ჩავატარებთ ალგორითმს A_1, მივიღებთ ისეთ ვითარებას, როგორც ნახ. 9-ში მარცხნივ. თუ შემდეგ კიდევ ჩავატარებთ ალგორითმს U, მივიღებთ ისეთ ვითარებას, როგორც ნახ. 9-ში მარჯვნივ. აღსანიშნავია, რომ ამ შემთხვევაში შეიქმნა ისეთივე ვითარება, როგორც ამოცანაში „ერთი ნავი". ეს კი იმას ნიშნავს, რომ თუ გამოვიყენებთ ალგორითმს A_1, საბოლოო მდგომარეობას მივაღწევთ.
ასე რომ, ალგორითმი A_2, რომელიც ამოცანას „ორი ნავი" ხსნის, შემდეგნაირად შეიძლება ჩაიწეროს: $A_2 = A_1, U, A_1$ (ჯერ ჩაატარე ალგორითმი A_1, შემდეგ ალგორითმი U და ბოლოს ისევ ალგორითმი A_1.

ახლა კი დავუშვათ, რომ ალგორითმი A_n n თეთრი ნავის გვერდის აქცევას ახერხებს (ნახ. 10). აქამდე ჩვენ განვიხილეთ, თუ როგორია A_n, თუ $n = 1$, ან $n = 2$.

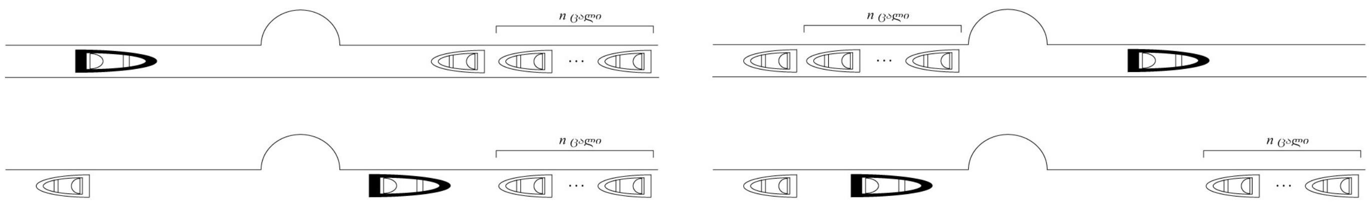


ნახ. 10:

თუ განვიხილავთ $n+1$ ნავის გვერდის აქცევის ამოცანას ისეთი საწყისი და საბოლოო მდგომარეობებით, რომლებიც ნაჩვენებია ნახ. 11-ში (ზემოთ) და ჩავატარებთ ალგორითმებს A_1, U, მივიღებთ ისეთ სიტუაციას, რომელიც გვქონდა n ნავის გვერდის აქცევის ამოცანაში (ნახ. 11 ქვემოთ).

ეს კი იმას ნიშნავს, რომ A_n ალგორითმის გამოყენების შემდეგ მიიღება საბოლოო მდგომარეობა (ნახ. 11 ზემოთ მარჯვნივ).

საბოლოოდ მივიღებთ შემდეგ ჩანაწერს: $A_{n+1} = A_1, U, A_n$. თუ ვიცით, როგორია ალგორითმი A_1, ადვილი გამოსათვლელია ალგორითმი A_2 (აქ $n+1 = 2$ და $n = 1$): ჯერ ჩავატარებთ ალგორითმს A_1, შემდეგ U და შემდეგ ისევე A_1. A_3 ალგორითმის ჩასატარებლად ჯერ უნდა ჩავატაროთ A_1, შემდეგ U და შემდეგ A_2. ასე ნაბიჯ-ნაბიჯ



ნახ. 11:

შეიძლება გამოვითვალოთ A_n ნებისმიერი ნატურალური n რიცხვისათვის: $A_n = A_1, U, A_{n-1} = A_1, U, A_1, U, A_{n-2} = A_1, U, A_1, U, A_1, U, A_{n-2} = A_1, U, A_1, U, \cdots, A_1$ (n-ჯერ).

<u>სავარჯიშო 2.1:</u> რისი ტოლია A_7? (მაგ.: $A_3 = A_1, U, A_2 = A_1, U, A_1, U, A_1$)

მნიშვნელოვანია ის ფაქტი, რომ ალგორითმი A_n იყენებს „თავის თავს", მხოლოდ უფრო დაბალი პარამეტრით (მაგ. $A_2 = A_1, U, A_1$; $A_7 = A_1, U, A_6$ და ა.შ.)
იმ შემთხვევაში, როდესაც ალგორითმი თავის თავს იყენებს, მას „რეკურსიული" ეწოდება. ესე იგი, $A_n = A_1, U, A_{n-1}$ ალგორითმის ეს ჩანაწერი რეკურსიულია.
აღსანიშნავია ისიც, რომ ნებისმიერი რეკურსიული ალგორითმი შეიძლება არარეკურსიული სახითაც ჩაიწეროს (განიხილეთ წინა სავარჯიშოს მაგალითი).

იმისათვის, რომ დავამტკიცოთ ამ ამოცანის სისწორე, უნდა გამოვიყენოთ მათემატიკური ინდუქციის პრინციპი, რომელიც ხშირად გამოიყენება ალგორითმების თეორიაში და ძალიან მნიშვნელოვან როლსაც თამაშობს:

- <u>რეკურსიის დასაწყისი:</u> A_1 ალგორითმი სწორია (ამის გადამოწმება ადვილია);
- <u>რეკურსიის დაშვება:</u> დავუშვათ, A_n ალგორითმი სწორია რაღაცა n ნატურალური რიცხვისათვის (და მასზე პატარა ყველა რიცხვისათვის);
- <u>რეკურსიის ბიჯი:</u> დავამტკიცოთ, რომ $A_{n+1} = A_1, U, A_n$ სწორია.

თუ დავამტკიცებთ, რომ A_{n+1} ალგორითმი სწორია და გვეცოდინება, რომ A_1 სწორია, მაშინ დავუშვებთ, რომ $n = 1$ და ამით დამტკიცდება, რომ $A_{n+1} = A_2$ სწორია. თუ A_2 სწორია და დავამტკიცებთ, რომ A_{n+1} სწორია, დამტკიცდება, რომ A_3 სწორია და ა.შ. ნებისმიერი ნატურალური რიცხვისათვის.

ახლა კი დავამტკიცოთ $A_{n+1} = A_1, U, A_n$ ალგორითმის სისწორე: A_1, U ალგორითმების შესრულების შემდეგ წარმოიშვება ზუსტად ისეთივე სიტუაცია, როგორც n ნავის გაყვანის ამოცანაში. ხოლო ინდუქციის დაშვების თანახმად A_n ალგორითმი n ნავის გაყვანის ამოცანას სწორად ხსნის. ასე რომ, A_1, U, A_n $n + 1$ ნავის გაყვანის ამოცანას სწორად ხსნის.

Q.E.D.

<u>სავარჯიშო 2.2:</u> სწორად ამოხსნის თუ არა შემდეგი ალგორითმი $A_n = A_{n-1}, U, A_1$ n ნავის გაყვანის ამოცანას?

ალგორითმის სისწორის მტკიცების შემდეგ საჭიროა მისი სისწრაფის, ანუ ბიჯების რაოდენობის დადგენა. A ალგორითმის ბიჯების რაოდენობას შემდეგნაირად აღნიშნავენ: $T(A)$. ჩვენს შემთხვევაში გვექნება $T(A_n)$.
რადგან ჯერ უნდა შევასრულოთ ალგორითმი A_1, ამის შემდეგ ალგორითმი U და ბოლოს ალგორითმი A_{n-1}, მაშინ A_n ალგორითმის ბიჯების რაოდენობა იქნება: $T(A_n) = T(A_1) + T(U) + T(A_{n-1})$ (ჯერ იმდენი, რამდენიც საჭიროა A_1 ალგორითმისათვის, შემდეგ იმდენი, რამდენიც საჭიროა U ალგორითმისათვის და ბოლოს იმდენი, რამდენიც საჭიროა A_{n-1} ალგორითმისათვის).
ეს ფორმულაც ჩაწერილია რეკურსიული სახით, რადგან იგი თავის თავს იყენებს, მხოლოდ უფრო დაბალი პარამეტრებით. მაგრამ მისი ჩაწერა არარეკურსიული სახითაც შეიძლება:

ჩვენ ვიცით, რომ $T(A_1) = 3$ და $T(U) = 1$ (შესაბამისი ალგორითმების გადამოწმებით ამაში ადვილად ვრწმუნდებით). აქედან გამომდინარე, ვიღებთ:

$$T(A_n) = T(A_{n-1}) + 4.$$

თავის მხრივ, $T(A_{n-1}) = T(A_{n-2}) + 4$, $T(A_{n-2}) = T(A_{n-3}) + 4$ ...

აქედან გამომდინარე,

$$T(A_n) = T(A_{n-1}) + 1 \cdot 4 = T(A_{n-2}) + 2 \cdot 4 = T(A_{n-3}) + 3 \cdot 4 = \cdots = T(A_1) + (n-1) \cdot 4 = 3 + (n-1) \cdot 4 = 4 \cdot n - 1.$$

სავარჯიშო 2.3: დაამტკიცეთ, რომ ამაზე უფრო სწრაფი ალგორითმი ვერ იარსებებს.

სავარჯიშო 2.4: განვიხილოთ კენტ რიცხვთა მიმდევრობა: $a_1 = 1$, $a_n = a_{n-1} + 2$. მათემატიკური ინდუქციის გამოყენებით დაამტკიცეთ: $a_n = 2n - 1$.

სავარჯიშო 2.5: მათემატიკურ ინდუქციაზე დაყრდნობით დაამტკიცეთ: თუ $a_1 = 1$ და $a_n = a_{n-1} + 2$ (კენტ რიცხვთა მიმდევრობა), მაშინ $\sum_{i=1}^{n} a_i = a_1 + a_2 + \cdots + a_n$ (პირველი n კენტი რიცხვის ჯამი) გამოითვლება ფორმულით:

$$\sum_{i=1}^{n} a_i = n^2.$$

სავარჯიშო 2.6: მოცემულია რეკურსიული ფორმულა: $S_n = S_{n-1} + 1$, $S_1 = 1$.. გახსენით რეკურსია (ჩაწერეთ S_n არარეკურსიული სახით ისე, როგორც ეს ნავების ალგორითმის ბიჯების რაოდენობის გამოთვლისას გავაკეთეთ).

სავარჯიშო 2.7: მოცემულია რეკურსიული ფორმულა: $K_n = K_{n-1} + n$, $K_1 = 1$. გახსენით რეკურსია.

სავარჯიშო 2.8: მოცემულია რეკურსიული ფორმულა: $L_n = 2 \cdot L_{n-1} + 1$, $L_1 = 1$. გახსენით რეკურსია.

2.2 ჰანოის კოშკების ამოცანა

1883 წელს ფრანგმა მათემატიკოსმა ედუარდ ლუკასმა დასვა შემდეგი ამოცანა:

მოცემულია: სამი ძელი A, B, C. A ძელზე ჩამოცმულია სხვადასხვა ზომის n რგოლი ისე, რომ დიდ რგოლს უფრო პატარა ადევს -- შექმნილია პირამიდა (ნახ. 12 (ა)).

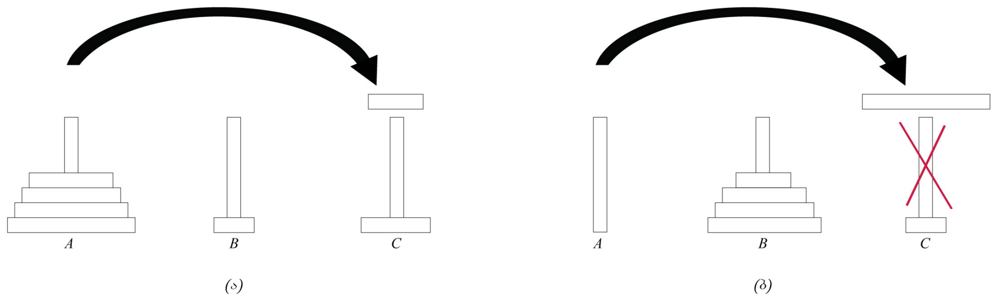
ნახ. 12: ჰანოის კოშკების ამოცანის საწყისი და საბოლოო მდგომარეობები

შედეგი: A ძელზე აგებული პირამიდა C ძელზე (ნახ. 12 (ბ)).

შეზღუდვა: თითო ჯერზე ერთი ძელიდან მეორეზე უნდა გადავიტანოთ ერთი და მხოლოდ ერთი რგოლი, რომელიც ყველაზე მაღლა დევს. ამავე დროს არ შეიძლება პატარა ზომის რგოლზე დიდი ზომის რგოლის დადება.

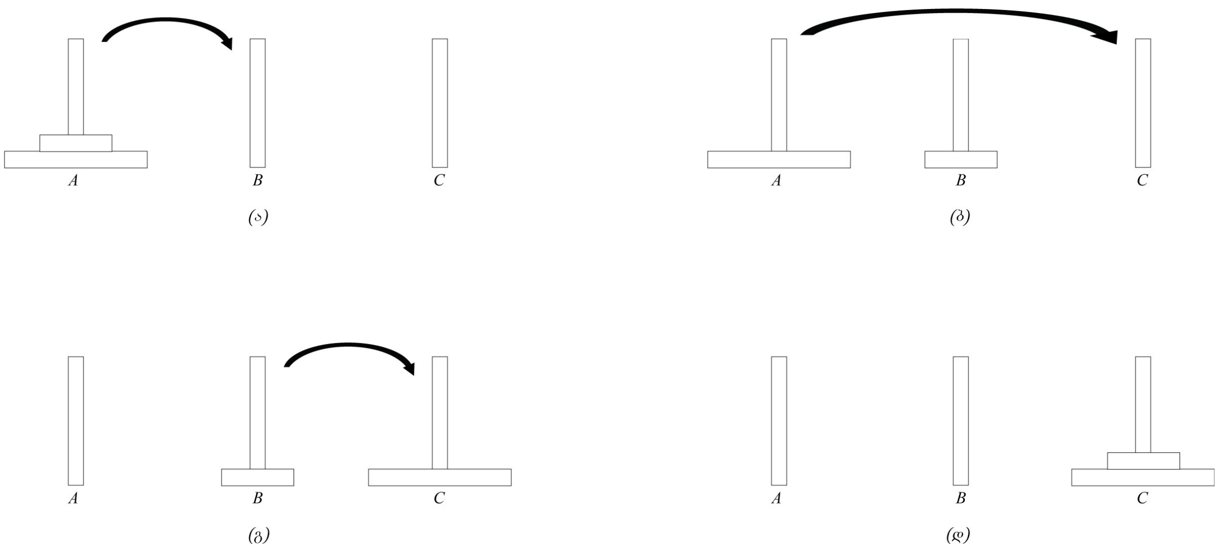
დავუშვათ, მოცემულია ერთ რგოლიანი პირამიდა. ცხადია, რომ მისი ერთი ძელიდან მეორეზე გადასატანად საკმარისია ერთი მოქმედება. თუ ეს ერთი რგოლი A ძელიდან C ძელზე გადაგვაქვს, ამ პროცედურას ვუწოდებთ $A_1^{A,C}$.

იმისათვის, რომ ორ რგოლიანი პირამიდა A ძელიდან C ძელზე გადავიტანოთ, საჭიროა შემდეგი მოქმედებების ჩატარება:



ნახ. 13: დასაშვები *(ა)* და აკრძალული *(ბ)* სვლები

1. A ძელიდან ზედა რგოლი გადაიტანე B ძელზე (ჩაატარე $A_1^{A,B}$, ნახ. 14 *(ბ)*);
2. A ძელიდან ზედა რგოლი გადაიტანე C ძელზე (ჩაატარე $A_1^{A,C}$, ნახ. 14 *(გ)*);
3. B ძელიდან ზედა რგოლი გადაიტანე C ძელზე (ჩაატარე $A_1^{B,C}$, ნახ. 14 *(დ)*).



ნახ. 14: ორ რგოლიანი პირამიდის გადატანისათვის საჭირო ოპერაციები

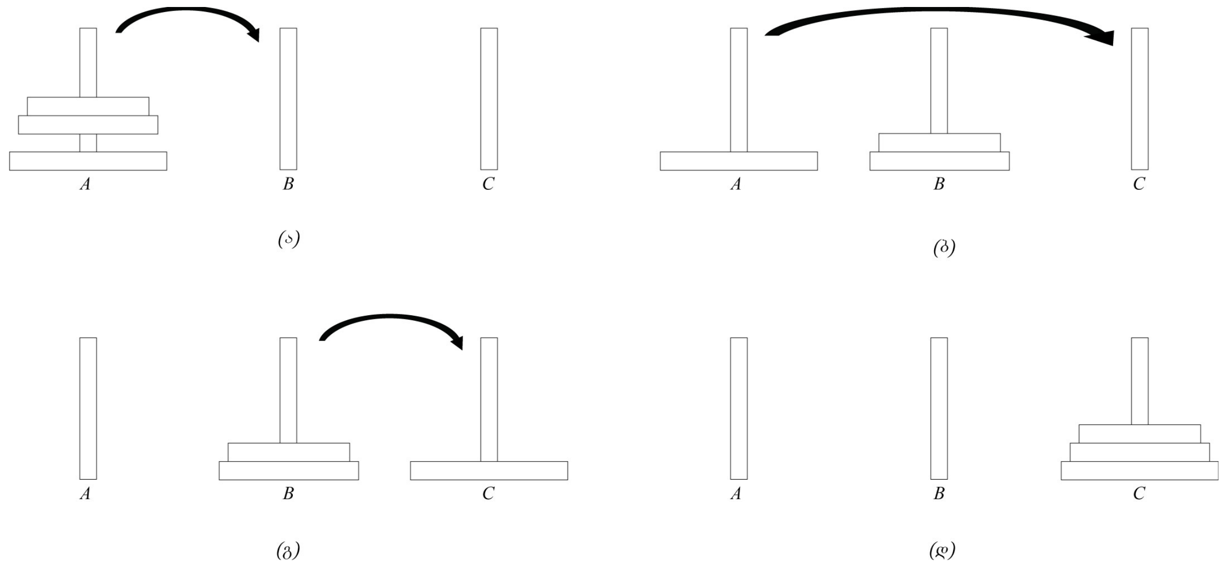
ორ რგოლიანი პირამიდის A ძელიდან C ძელზე გადატანის ალგორითმი (ანუ ზემოთ მოყვანილი სამ ბიჯიანი პროცესი) აღვნიშნოთ როგორც $A_2^{A,C}$.

ზოგადად, n რგოლის ერთი ძელიდან მეორეზე გადატანის ალგორითმი შემდეგნაირად შეიძლება აღინიშნოს: $A_n^{X_1,X_2}$. აქ $n \in \mathbb{N}$, $X_1, X_2 \in \{A, B, C\}$ და $X_1 \neq X_2$. ამრიგად, $A_{13}^{C,A}$ ნიშნავს ალგორითმს, რომელიც C ძელზე აწყობილ 13 რგოლიან პირამიდას A ძელზე გადაიტანს, ხოლო $A_{108}^{B,A}$ კი იმ ალგორითმს, რომელიც B ძელზე აწყობილ 108 რგოლიან პირამიდას A ძელზე გადაიტანს.

თუ ვიცით, როგორ გადავიტანოთ ორ რგოლიანი პირამიდა ერთი ძელიდან მეორეზე, ადვილად შევადგენთ ალგორითმს $A_3^{A,C}$:

სამ რგოლიანი პირამიდა განვიხილოთ, როგორც ქვედა დიდ რგოლზე დადგმული ორ რგოლიანი პირამიდა (ნახ. 17 *(ა)*).

ამრიგად, $A_2^{A,B}$ ალგორითმით შეიძლება ზედა ორ რგოლიანი პირამიდის გადატანა B ძელზე (ნახ. 17 *(ბ)*), შემდეგ $A_1^{A,C}$ ალგორითმით ქვედა რგოლი გადაგვაქვს A ძელიდან C ძელზე (ნახ. 17 *(გ)*) და ბოლოს ისევე $A_2^{B,C}$ ალგორითმით ორ რგოლიანი პირამიდა გადაგვაქვს B ძელიდან C ძელზე (ნახ. 17 *(დ)*).



ნახ. 15: სამ რგოლიანი პირამიდის გადატანისათვის საჭირო ოპერაციები

ეს ალგორითმი რეკურსიულად შემდეგნაირად შეიძლება ჩაიწეროს: $A_3^{A,B} = [A_2^{A,B}, A_1^{A,C}, A_2^{B,C}]$ (ჯერ შეასრულე $A_2^{A,B}$, შემდეგ $A_1^{A,C}$ და ამის შემდეგ $A_2^{B,C}$.

აღსანიშნავია, რომ $A_2^{A,B}$ და $A_2^{B,C}$ თვითონ რამოდენიმე ბიჯისაგან შედგება: $A_2^{A,B} = [A_1^{A,C}, A_1^{A,B}, A_2^{C,B}]$ და $A_2^{B,C} = [A_1^{B,A}, A_1^{B,C}, A_2^{A,C}]$.

სავარჯიშო 2.9: რეკურსიულად ჩაწერეთ $A_3^{B,C}$, $A_3^{C,A}$, $A_3^{A,B}$, $A_3^{B,A}$ და $A_3^{C,B}$ (იხ. ზემოთ მოყვანილი ანალოგიური ჩანაწერი $A_3^{A,C}$).

თუ ვიცით, როგორ გადავიტანოთ 3 რგოლიანი პირამიდა ერთი ძელიდან მეორეზე, რეკურსიულად შეიძლება $A_4^{X_1,X_2}$ ალგორითმის დადგენა. მაგ., $A_4^{A,C} = [A_3^{A,B}, A_1^{A,C}, A_3^{B,C}]$.

სავარჯიშო 2.10: რეკურსიულად ჩაწერეთ $A_4^{B,C}$, $A_4^{C,A}$, $A_4^{A,B}$, $A_4^{B,A}$ და $A_4^{C,B}$ (იხ. ზემოთ მოყვანილი ანალოგიური ჩანაწერი $A_3^{A,C}$).

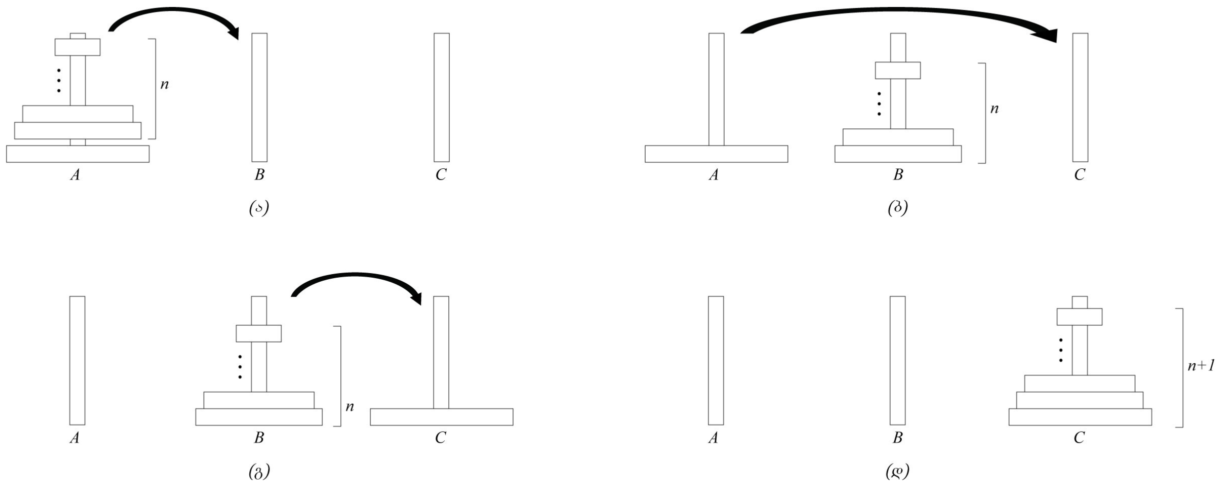
თუ ვიცით, როგორია n რგოლიანი პირამიდის ერთი ძელიდან მეორეზე გადატანის ალგორითმი $A_n^{X_1,X_2}$, ადვილად შევადგენთ $n+1$ რგოლიანი ალგორითმის გადატანის ალგორითმს $A_{n+1}^{X_1,X_2}$ (შესაბამისი მოქმედებები ნაჩვენებია ნახ. 16 -ში):

$$A_{n+1}^{X_1,X_2} = [A_n^{X_1,X_3}, A_1^{X_1,X_2}, A_n^{X_3,X_2}], \quad X_1 \neq X_2 \neq X_3, \quad X_1, X_2, X_3 \in \{A, B, C\}.$$

როგორც ყველა წინა მაგალითში, აქაც n ცალი რგოლის გადატანა ერთდროულადაა ნაჩვენები იმის და მიუხედავად, რომ $A_n^{X_1,X_2}$ რამოდენიმე ბიჯისაგან შედგება.

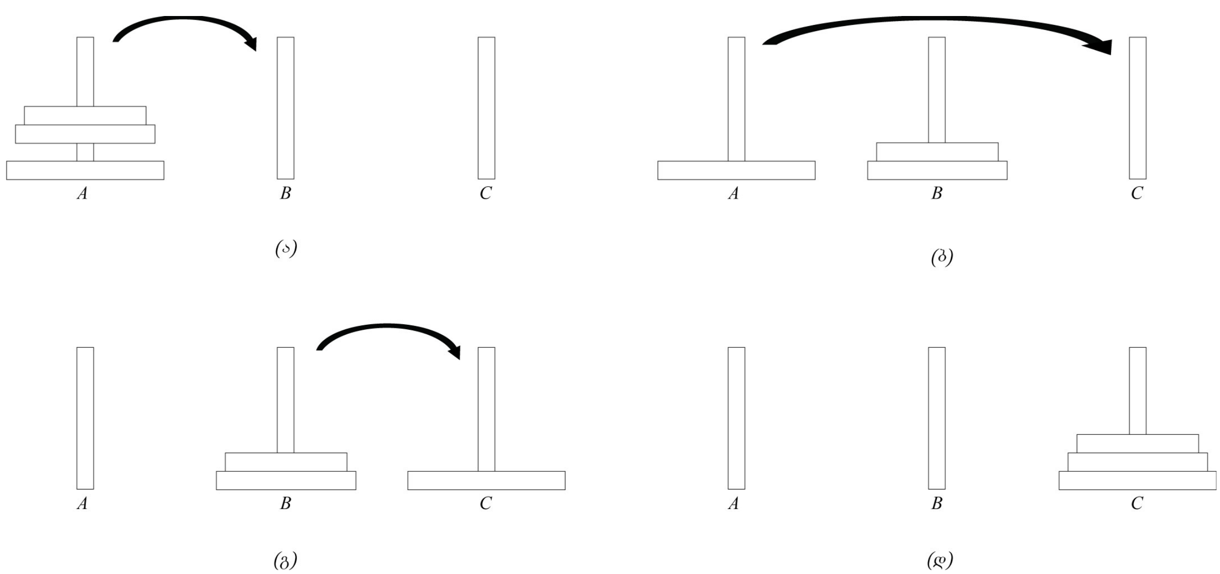
სავარჯიშო 2.11: რას აღნიშნავს შემდეგი ჩანაწერები: $A_7^{B,C}$, $A_{12}^{C,B}$, $A_4^{B,C}$?

ადვილი შესამოწმებელია, რომ $A_1^{X_1,X_2}$ ალგორითმის შესრულებისას ამოცანის პირობა არ ირღვევა. თუ განვიხილავთ $A_2^{A,C}$ ალგორითმის რეკურსიულ ჩანაწერს, დავინახავთ, რომ პირველ რიგში უნდა შევასრულოთ ალგორითმი $A_1^{A,B}$. ადვილი სანახავია, რომ ამ ალგორითმის შესრულებისასაც პირობა არ ირღვევა. შემდეგ უნდა შევასრულოთ $A_1^{A,C}$. რადგან C ძელზე რგოლი არ დევს, მასზე A ძელიდან რგოლის გადატანა შესაძლებელია (პირობა არ დაირღვევა) და ჩ ძელზე ყველაზე დიდი რგოლი იდება. ბოლოს უნდა ჩავატაროთ $A_1^{B,C}$. ეს შესაძლებელია, რადგან C ძელზე ყველაზე დიდი რგოლი დევს.



ნახ. 16: $n+1$ რგოლიანი პირამიდის გადატანისათვის საჭირო ოპერაციები

ანალოგიური მსჯელობით შეიძლება დავამტკიცოთ, რომ თუ $A_3^{A,C}$ ალგორითმს ჩავწერთ ისე, როგორც ზემოთ განვიხილეთ და მას თანმიმდევრულად შევასრულებთ, ამოცანის პირობა არ ირღვევა: პირველ რიგში უნდა შესრულდეს $A_2^{A,B}$ (ნახ. 17 *(ა)*). ეს შესაძლებელია, რადგან B და C ძელები ცარიელია და A ძელზე ქვემოთ ყველაზე დიდი რგოლი დევს, რომელზეც პირობის თანახმად სხვა ნებისმიერი რგოლის დადება შეიძლება. ასე რომ, ამ ოპერაციების შესრულების დროს ამოცანის პირობა არ დაირღვევა. შედეგად მივიღებთ A ძელზე ერთ ყველაზე დიდ რგოლს და B ძელზე კი ორ რგოლიან პირამიდას (ნახ. 17 *(ბ)*). შემდეგ უნდა ჩავატაროთ $A_1^{A,C}$. ესეც არ არღვევს ამოცანის პირობას, რადგან ამ მომენტისათვის C ძელი ცარიელია. შედეგად მივიღებთ C ძელზე ერთ ყველაზე დიდ რგოლს და B ძელზე კი ორ რგოლიან პირამიდას, ხოლო A ძელი კი ცარიელი იქნება (ნახ. 17 *(გ)*). ბოლოს უნდა შევასრულოთ $A_2^{B,C}$. ესეც შესაძლებელია, რადგან A ძელი ცარიელია და C ძელზე ყველაზე დიდი რგოლი დევს, რომელზედაც ყველა დანარჩენი რგოლის დადება შეიძლება. ამ ოპერაციების ჩატარების შედეგად ამოცანის საბოლოო შედეგს მივიღებთ (ნახ. 17 *(დ)*).



ნახ. 17: სამ რგოლიანი პირამიდის გადატანისათვის საჭირო ოპერაციები

სავარჯიშო 2.12: დავუშვათ, მოცემულია შემდეგი ჩანაწერი: $A_3^{A,C} = [A_1^{A,B}, A_2^{A,C}, A_1^{B,C}]$. სიტყვიერად ახსენით, რა ოპერაციები უნდა შესრულდეს ამ ჩანაწერის შესაბამისად. ირღვევა თუ არა ამ ალგორითმის შესრულებისას ჰანოის კოშკების ამოცანის პირობა?

სავარჯიშო 2.13: მათემატიკურ ინდუქციაზე დაყრდნობით დაამტკიცეთ $A_{n+1}^{A,C} = [A_n^{A,B}, A_1^{A,C}, A_n^{B,C}]$ ალგორითმის სისწორე.

სავარჯიშო 2.14: ზემოთ მოყვანილ მსჯელობაში, $A_3^{A,C}$ ალგორითმის სისწორის მტკიცებისას, რამოდენიმეჯერ აღვნიშნეთ, რომ ერთი ძელი ცარიელია (C ან A). რა საჭიროა ეს შენიშვნა სისწორის მტკიცებისას?

იმისათვის, რომ დავადგინოთ, თუ რამდენ ბიჯს ანდომებს ალგორითმი A_n, განვიხილოთ მისი რეკურსიული ჩანაწერი: $A_n^{A,C} = [A_{n-1}^{A,B}, A_1^{A,C}, A_{n-1}^{B,C}]$.

რაიმე K ალგორითმის ბიჯების რაოდენობა შემდეგნაირად აღიწერება: $T(K)$. ამრიგად, $T(A_n^{A,C})$ $A_n^{A,C}$) ალგორითმის ბიჯების რაოდენობაა.

სავარჯიშო 2.15: რას აღნიშნავს $T(A_{n+3}^{A,C}))$, $T(A_3^{C,B}))$, $T(A_7^{A,C}))$?

სავარჯიშო 2.16: რისი ტოლია $T(A_1^{A,C})$ და $T(A_2^{A,C})$?

სავარჯიშო 2.17: დაამტკიცეთ, რომ $T(A_1^{A,C}) = T(A_1^{B,C})$ და ზოგადად: $T(A_n^{X_1,X_2}) = T(A_n^{Y_1,Y_2})$ $\forall X_1, X_2, Y_1, Y_2 \in \{A,B,C\}$ და $X_1 \neq X_2$, $Y_1 \neq Y_2$ (არ აქვს მნიშვნელობა, რომელი ძელიდან რომელზე გადავაწყობთ პირამიდას - ბიჯების რაოდენობა უცვლელია).

რადგან $A_n^{A,C} = [A_{n-1}^{A,B}, A_1^{A,C}, A_{n-1}^{B,C}]$, ჯერ უნდა შესრულდეს $A_{n-1}^{A,B}$, შემდეგ $A_1^{A,C}$ და ბოლოს $A_{n-1}^{B,C}$. აქედან გამომდინარე,

$$T(A_n^{A,C}) = T(A_{n-1}^{A,B}) + T(A_1^{A,C}) + T(A_{n-1}^{B,C}) = 2 \cdot T(A_{n-1}^{A,B}) + 3$$

(იხ. წინა სავარჯიშოები).

სავარჯიშო 2.18: მათემატიკური ინდუქციის გამოყენებით დაამტკიცეთ:

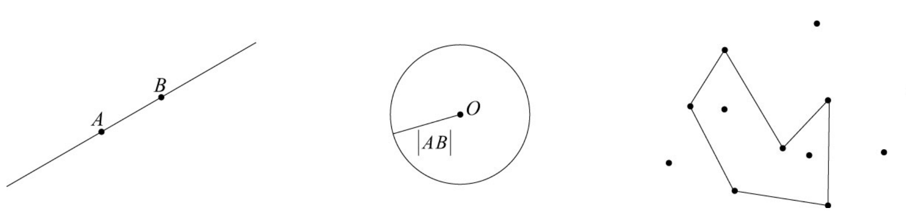
$$T(A_n^{A,C}) = 2^n - 1.$$

2.3 ძველი ბერძნული ამოცანები

ანტიკურ საბერძნეთში დასვეს ე.წ. „ფარგლითა და სახაზავით აგების" გეომეტრიული ამოცანები. აღსანიშნავია, რომ რამოდენიმე ამოცანა 2000 წელზე მეტ ხანს ამოუხსნელი რჩებოდა, სანამ XIX საუკუნეში მათემატიკურად არ დამტკიცდა, რომ მათი ალგორითმული გადაჭრა შეუძლებელია. ეს, ალბათ, ყველაზე ძველი ამოცანებია, რომელთაც ალგორითმული ამოხსნა არ აქვთ.

მოცემულია: ფარგალი, სახაზავი და ორი წერტილი სიბრტყეზე; რაიმე გეომეტრიული ფიგურა;
რაიმე ნამდვილი რიცხვი ξ.

შეზღუდვა: სახაზავით შეიძლება მოცემულ ორ წერტილზე A და B წრფის გავლება. თუ მოცემულია ნებისმიერი ორი წერტილი A, B და ნებისმიერი მესამე წერტილი O, ფარგლით შეიძლება O წერტილიდან $|A,B|$ სიგრძის რადიუსის მქონე წრეწირის შემოვლება (ნახ. 18).



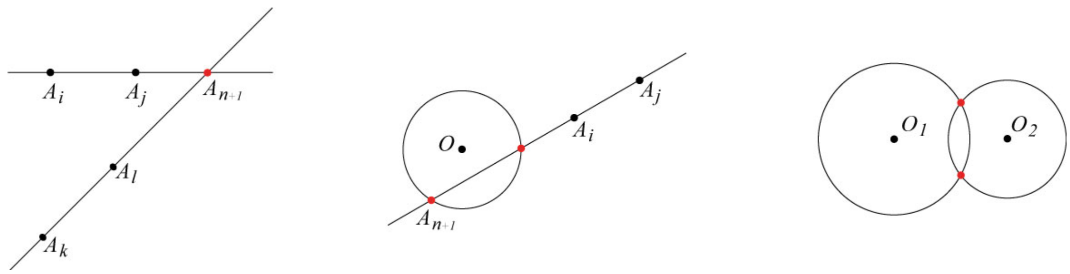
ნახ. 18: სახაზავით (მარცხნივ), ფარგლით (შუაში) და წერტილებზე აგებული ფიგურები

თუ მოცემულია უკვე აგებულ წერტილთა რაიმე სიმრავლე $\mathcal{S} = \{A_1, A_2, ..., A_n\}$, ამ სიმრავლის რამოდენიმე წერტილზე გავლებულ შეკრულ ტეხილს ფარგლითა და სახაზავით აგებული ფიგურა ეწოდება.

ახალი A_{n+1} წერტილი ითვლება ფარგლითა და სახაზავით აგებულად, თუ:

- $\exists\, A_i, A_j, A_k, A_l \in \mathcal{S}$ და A_{n+1} არის A_i, A_j წერტილებზე გავლებული წრფისა და A_k, A_l წერტილებზე გავლე-ბული წრფის გადაკვეთის წერტილი (ნახ. 19 მარცხნივ);
- $\exists\, A_i, A_j, A_k, A_l, O \in \mathcal{S}$ და A_{n+1} არის A_i, A_j წერტილებზე გავლებული წრფისა და O წერტილზე $|A_k, A_l|$ სიგრძის რადიუსის მქონე წრეწირის გადაკვეთის წერტილი (ნახ. 19 შუაში);
- $\exists\, A_i, A_j, A_k, A_l, O_1, O_2 \in \mathcal{S}$ და A_{n+1} არის O_1 წერტილზე $|A_k, A_l|$ სიგრძის რადიუსის მქონე წრეწირისა და O_2 წერტილზე $|A_i, A_j|$ სიგრძის რადიუსის მქონე წრეწირის გადაკვეთის წერტილი (ნახ. 19 მარჯვნივ).

შენიშვნა: $A_i, A_j, A_k, A_l, O_1, O_2 \in \mathcal{S}$ წერტილთა შორის რამოდენიმე შეიძლება ერთმანეთს ემთხვეოდეს.



ნახ. 19: ფარგლითა და სახაზავით ახალი წერტილების აგების შესაძლებლობები

რაიმე გეომეტრიული ფიგურა ითვლება აგებულად, თუ ფარგლითა და სახაზავით ზემოთ აღწერილი წესების დაცვით აიგება ისეთი სიმრავლე $\mathcal{S}$, რომ მასში მოიძებნოს ისეთი წერტილები, რომელთა ტეხილებით შეერთება ამ საძიებელ ფიგურას მოგვცემს.

რაიმე რიცხვი ξ ითვლება აგებულად, თუ ფარგლითა და სახაზავით ზემოთ აღწერილი წესების დაცვით აიგება ისეთი სიმრავლე $\mathcal{S}$, რომ მასში მოიძებნოს ორი წერტილი, რომელთა შორის მანძილია ξ.

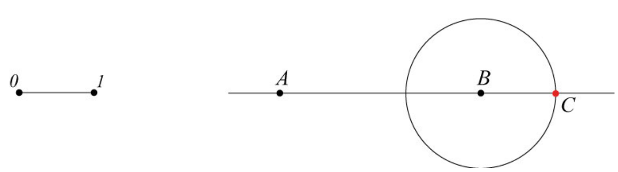
შედეგი: მოცემული გეომეტრიული ფიგურისთვისა ან რიცხვისთვის დაადგინეთ, შეიძლება თუ არა მათი ფარ-გლითა და სახაზავით აგება.

დასაწყისისათვის მოცემულია ორი წერტილი A და B, რომელთა შორის მანძილი ერთის ტოლადაა მიჩნეული: $|A, B| = 1$. სხვა სიტყვებით რომ ვთქვათ, აგებულია რიცხვი 1. იმისათვის, რომ ავაგოთ რიცხვი 2 (ანუ ფარგლისა და სახაზავის მეშვეობით ავაგოთ ისეთი წერტილები, რომელთა შორის მანძილი ორის ტოლია), შემდეგი ალგორითმი უნდა გამოვიყენოთ:

მოცემულია: ორი წერტილი A და B.

1. A და B წერტილებზე გაავლე წრფე;
2. ფარგლით შემოხაზე წრეწირი ცენტრით B წერტილში და რადიუსით 1;

   ეს წრეწირი AB წრფეს გადაკვეთს ორ წერტილში: D (წერტილიდან მარცხნივ) და ახალ C წერტილში, B წერტილიდან მარჯვნივ.
3. პასუხად გამოიტანე ორი წერტილი: A და C.



ნახ. 20: $|A, B| + 1$ სიგრძის მონაკვეთის აგება

ეს ალგორითმი აღვნიშნოთ როგორც N. თუ მისი მონაცემებია A და B წერტილები, $N(A,B)=(A,C)$. ადვილი საჩვენებელია, რომ $|A,C|=|A,B|+1$.

ესე იგი, თუ მოცემულია ორი წერტილი A და B, რომელთა შორის მანძილია 1, შეიძლება $n \in \mathbb{N}$ რიცხვის აგება შემდეგი რეკურსიული ალგორითმით:

- $P_1=(A,B)$;
- $P_n=N(P_{n-1})$.

სავარჯიშო 2.19: გამოითვალეთ $T(P_n)$. ჩათვალეთ, რომ ორ წერტილზე წრფის გავლების, მოცემულ ორ წერტილს შორის მანძილის ფარგლით მონიშვნისა და მოცემულ წერტილზე რაიმე რადიუსით წრეწირის გავლების ბიჯების რაოდენობა ერთის ტოლია.

სავარჯიშო 2.20: მოცემულია ოთხი წერტილი A,B,C,D. რა ალგორითმით შეიძლება $|A,B|+|C,D|$ სიგრძის მონაკვეთის აგება? გამოითვალეთ ამ ალგორითმის ბიჯების რაოდენობა და დაამტკიცეთ მისი სისწორე.

სავარჯიშო 2.21: მოცემულია ორი ცერტილი A,B, სადაც $|A,B|>1$. შეადგინეთ ალგორითმი, რომელიც $|A,B|-1$ სიგრძის მონაკვეთს ააგებს. გამოითვალეთ ამ ალგორითმის ბიჯების რაოდენობა და დაამტკიცეთ მისი სისწორე.

თუ მოცემულია ორი წერტილი A და B, ადვილად შეიძლება $[A,B]$ მონაკვეთის შუა პერპენდიკულარული წრფის აგება, ანუ ისეთი ორი წერტილის აგება, რომლებზე გამავალი წრფეც ამ მონაკვეთის პერპენდიკულარულია და მის შუა წერტილზე გადის (ცხადია, რომ იგივე ალგორითმით შეიძლება ამავე მონაკვეთის შუა წერტილის დადგენა):

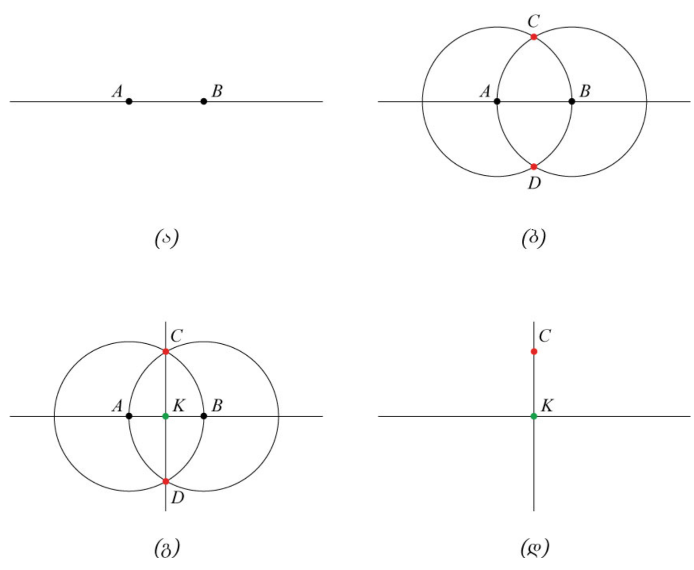
მოცემულია: ორი წერტილი A და B (ნახ. 21 *(ა)*).

- A წერტილზე შემოავლე $|A,B|$ რადიუსის წრეწირი;
- B წერტილზე შემოავლე $|A,B|$ რადიუსის წრეწირი (ნახ. 21 *(ბ)*)

  შედეგი: ამ ორი წრეწირის გადაკვეთის წერტილები C და D.
- შეაერთე C და D წერტილები წრფით (ნახ. 21 *(გ)*).

  შედეგი: ამ წრფისა და A,B მონაკვეთის გადაკვეთის წერტილი K.
- გამოიტანე პასუხი: ორი წერტილი C და K (ნახ. 21 *(დ)*).

ცხადია, რომ C და K წერტილებზე გავლებული წრფე $[A,B]$ მონაკვეთის შუა პერპენდიკულარულია.



ნახ. 21: $[A,B]$ მონაკვეთის შუა პერპენდიკულარულის აგება

ეს ალგორითმი აღვნიშნოთ როგორც $P(A,B)$. ამრიგად, $P(A,B) = (C,K)$, სადაც K $[A,B]$ მონაკვეთის შუა წერტილია.

თუ მოცემულია ორი წერტილი A და B და ერთი წერტილი C, რომელის არ ემთხვევა A წერტილს, მაშინ შეიძლება C წერტილიდან (A,B) წრფეზე პერპენდიკულარული წრფის დაშვება, ანუ ისეთი D წერტილის აგება (A,B) წრფეზე, რომ (C,D) წრფე (A,B) წრფის პერპენდიკულარული იყოს:

მოცემულია: ორი წერტილი A და B და ერთი წერტილი C, რომელიც არ დევს (A,B) წრფეზე (ნახ. 22 *(ა)*).

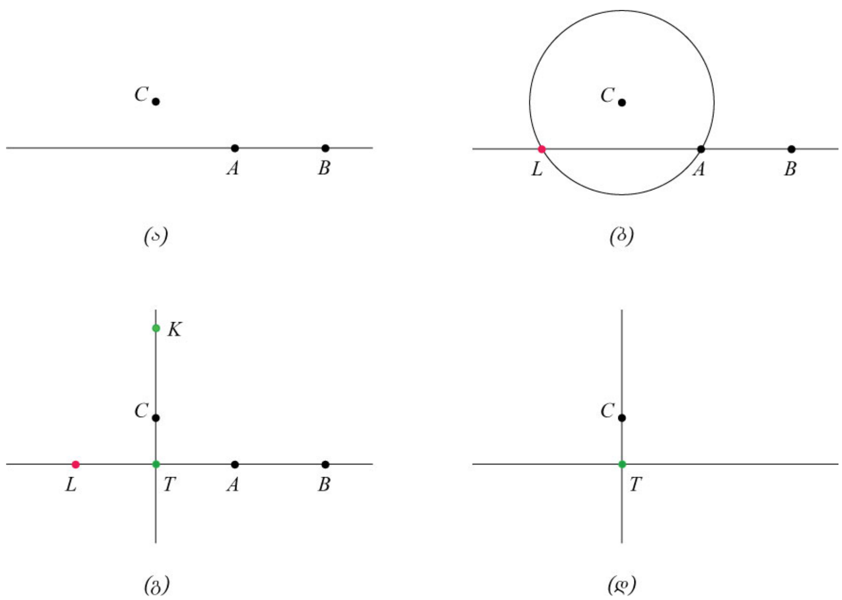
- C წერტილზე შემოავლე $|A,C|$ რადიუსის წრეწირი (ნახ. 22 *(ბ)*);

  შედეგი: ამ წრეწირისა და (A,B) წრფის გადაკვეთის მეორე წერტილი L.

- ჩაატარე ალგორითმი $P(A,L)$.

  შედეგი: ორი წერტილი K და T, რომელთაგან T დევს (A,B) წრფეზე (ნახ. 22 *(გ)*).

- გამოიტანე პასუხი: ორი წერტილი C და T (ნახ. 22 *(დ)*).



ნახ. 22: წერტილიდან წრფეზე პერპენდიკულარულის დაშვების პროცესი

სავარჯიშო 2.22: ზემოთ მოყვანილ ალგორითმში A და L წერტილებზე უნდა ჩავატაროთ $P(A,L)$ ალგორითმი. დაწვრილებით აღწერეთ ნახაზებით ეს პროცესი, რომლის შედეგადაც მიიღება K და T წერტილები.

სავარჯიშო 2.23: რა მოხდება, თუ C წერტილში $|A,C|$ რადიუსით გავლებული წრეწირი (A,B) წრფეს მხოლოდ ერთ წერტილში გადაკვეთს და მეორე L წერტილი არ მიიღება?

სავარჯიშო 2.24: ზემოთ მოყვანილ ალგორითმში, $P(A,L)$ ალგორითმის შესრულების შემდეგ, რატომ მიიღება ორი დამატებითი წერტილი K და T?

სავარჯიშო 2.25: დაამტკიცეთ, რომ (C,T) წრფე (A,B) წრფის პერპენდიკულარულია.

სავარჯიშო 2.26: მოცემულია ერთ წრფეზე მყოფი სამი წერტილი A, B და მათ შორის მდებარე C. რა ალგორითმით შეიძლება C წერტილიდან (A,B) წრფის პერპენდიკულარული წრფის აგება?

<u>სავარჯიშო 2.27:</u> მოცემულია ორი წერტილი A და B და ერთი წერტილი C, რომელიც არ დევს (A,B) წრფეზე. რა ალგორითმით შეიძლება C წერტილიდან (A,B) წრფის პარალელური წრფის აგება (ანუ ისეთი D წერტილის აგება, რომ (C,D) წრფე (A,B) წრფის პარალელური იყოს)?

თუ აგებულია ორი რიცხვი $a_1, a_2 \in \mathbb{N}$, ანუ A_1, A_2, A_3, A_4 ისეთი, რომ $|A_1, A_2| = a_1$ და $|A_3, A_4| = a_2$, მაშინ შეიძლება ისეთი ორი B_1, B_2 წერტილის აგება ფარგლითა და სახაზავით, რომ $|B_1, B_2| = a_1 \cdot a_2$:

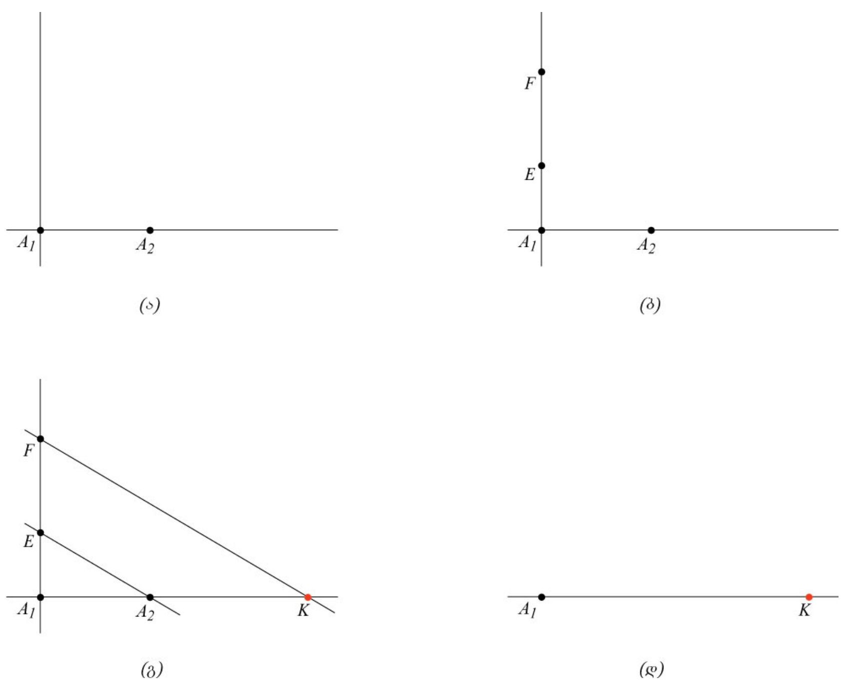
მოცემულია: ოთხი წერტილი A_1, A_2, A_3, A_4, სადაც $|A_1, A_2| = a_1$ და $|A_3, A_4| = a_2$.

- A_1 წერტილზე გაავლე (A_1, A_2) წრფის პერპენდიკულარული წრფე (ნახ. 23 *(ა)*) ;
- ამ წრფეზე A_1 წერტილიდან გადაზომე ერთის ტოლი მონაკვეთი;

  შედეგი: (A_1, A_2) წრფის პერპენდიკულარულ წრფეზე მდებარე წერტილი E, სადაც $|A_1, E| = 1$ (ნახ. 23 *(ბ)*).
- იგივე წრფეზე A_1 წერტილიდან გადაზომე $|A_3, A_4|$ სიგრძის მონაკვეთი;

  შედეგი: (A_1, A_2) წრფის პერპენდიკულარულ წრფეზე მდებარე წერტილი F, სადაც $|A_1, F| = |A_3, A_4|$ (ნახ. 23 *(ბ)*).
- E და A_2 წერტილებზე გაავლე წრფე;
- F წერტილიდან გაავლე $|E, A_2|$ წრფის პარალელური წრფე;

  შედეგი: ამ წრფისა და (A_1, A_2) წრფის გადაკვეთის წერტილი K (ნახ. 23 *(გ)*).
- გამოიტანე პასუხი: ორი წერტილი A_1 და K (ნახ. 23 *(დ)*).



ნახ. 23: $|A_1, A_2| \cdot |A_1, F|$ სიგრძის მონაკვეთის აგების პროცესი

<u>სავარჯიშო 2.28:</u> სამკუთხედების მსგავსებით დაამტკიცეთ, რომ $|A_1, K| = a_1 \cdot a_2$.

<u>სავარჯიშო 2.29:</u> დაამტკიცეთ, რომ თუ $a_2 < 1$, ალგორითმი მაინც სწორად მუშაობს.

ანალოგიურად შეიძლება $\frac{a_1}{a_2}$ სიგრძის მონაკვეთის აგება, თუ მოცემულია ოთხი წერტილი A_1, A_2, A_3, A_4, სადაც $|A_1, A_2| = a_1$ და $|A_3, A_4| = a_2$.

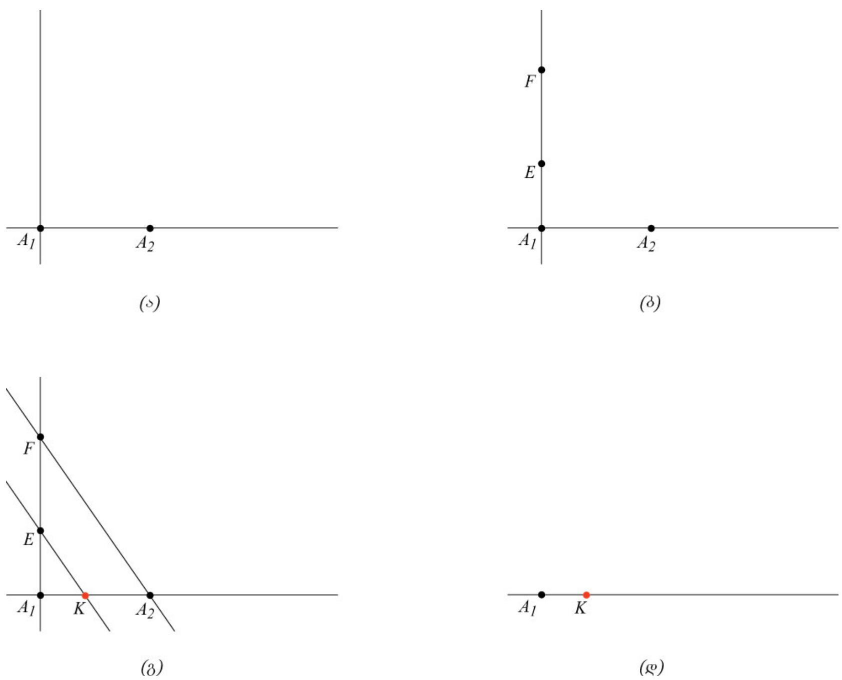
მოცემულია: ოთხი წერტილი A_1, A_2, A_3, A_4, სადაც $|A_1, A_2| = a_1$ და $|A_3, A_4| = a_2$.

- A_1 წერტილზე გაავლე (A_1, A_2) წრფის პერპენდიკულარული წრფე (ნახ. 24 *(ა)*);
- ამ წრფეზე A_1 წერტილიდან გადაზომე ერთის ტოლი მონაკვეთი;

  შედეგი: (A_1, A_2) წრფის პერპენდიკულარულ წრფეზე მდებარე წერტილი E, სადაც $|A_1, E| = 1$ (ნახ. 24 *(ბ)*) .
- იგივე წრფეზე A_1 წერტილიდან გადაზომე $|A_3, A_4|$ სიგრძის მონაკვეთი;

  შედეგი: (A_1, A_2) წრფის პერპენდიკულარულ წრფეზე მდებარე წერტილი F, სადაც $|A_1, F| = |A_3, A_4|$ (ნახ. 24 *(ბ)*).
- F და A_2 წერტილებზე გაავლე წრფე;
- E წერტილიდან გაავლე $|F, A_2|$ წრფის პარალელური წრფე;

  შედეგი: ამ წრფისა და (A_1, A_2) წრფის გადაკვეთის წერტილი K (ნახ. 24 *(გ)*).
- გამოიტანე პასუხი: ორი წერტილი A_1 და K (ნახ. 24 *(დ)*).



ნახ. 24: $\frac{|A_1, A_2|}{|A_1, F|}$ სიგრძის მონაკვეთის აგების პროცესი

სავარჯიშო 2.30: სამკუთხედების მსგავსებით დაამტკიცეთ, რომ $|A_1, K| = \frac{a_1}{a_2}$.

ამრიგად ჩვენ გვაქვს ნებისმიერი რაციონალური რიცხვის აგების მეთოდი, ანუ ფარგლითა და სახაზავით მთლიანად შეიძლება აიგოს ნებისმიერი რაციონალურ რიცხვი $a \in \mathbb{Q}$.

ბუნებრივია შემდეგი შეკითხვა: შეიძლება თუ არა ირაციონალური რიცხვების აგება ფარგლითა და სახაზავით? პირველი ასეთი რიცხვი არის $\sqrt{2}$, რომელიც პითაგორას თეორემაზე დაყრდნობით აიგება:

მოცემულია: ორი წერტილი A_1, A_2.

- A_1 წერტილზე გაავლე (A_1, A_2) წრფის პერპენდიკულარული წრფე;
- ამ წრფეზე A_1 წერტილიდან გადაზომე ერთის ტოლი მონაკვეთი;

შედეგი: (A_1, A_2) წრფის პერპენდიკულარულ წრფეზე მდებარე წერტილი E, სადაც $|A_1, E| = 1$.

- გამოიტანე პასუხი: ორი წერტილი A_1 და E.

სავარჯიშო 2.31: დახაზეთ ზემოთ მოყვანილი ალგორითმის დიაგრამები ისე, როგორც ეს წინა ალგორითმებისთვის იყო ნაჩვენები.

სავარჯიშო 2.32: დაამტკიცეთ, რომ $|A_1, E| = \sqrt{2}$, თუ $|A_1, A_2| = 1$.

ამ ალგორითმს ვუწოდოთ S. ესე იგი, თუ მოცემულია ორი წერტილი A, B ისე, რომ $|A, B| = a$, $S(A, B) = (A, C)$, სადაც $|A, C| = \sqrt{a+1}$.

აქედან გამომდინარეობს, რომ შემდეგი რეკურსიული ალგორითმი $H(n)$ ორ წერტილს გვაძლევს, რომელთა შორის მანძილია $\sqrt{n}$:

ალგორითმი $H(n)$:

- თუ $n = 1$, გამოიტანე ორი წერტილი A, B, სადაც $|A, B| = 1$ და ალგორითმი დაამთავრე;
- თუ $n > 1$:

  გაუშვი ალგორითმი $H(n-1)$;

სავარჯიშო 2.33: მათემატიკური ინდუქციით დაამტკიცეთ $H(n)$ ალგორითმის სისწორე.

სავარჯიშო 2.34: რისი ტოლია $T(H(n))$?

სავარჯიშო 2.35: ზემოთ მოყვანილი ალგორითმების საფუძველზე შეადგინეთ ალგორითმი, რომელიც ფესვს ნებისმიერი რაციონალური რიცხვიდან გამოიანგარიშებს.

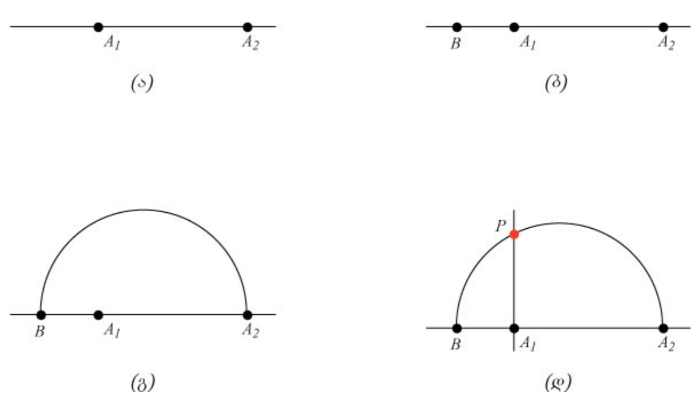
ახლა კი განვიხილოთ შემდეგი ალგორითმი:

მოცემულია: ორი წერტილი A_1, A_2, სადაც $|A_1, A_2| = \xi$ (ნახ. 25 *(ა)*).

- A_1 წერტილის მარცხნივ (A_1, A_2) წრფეზე გადაზომე ერთის ტოლი მონაკვეთი და მიღებული წერტილი იყოს B ($|B, A_1| = 1$) (ნახ. 25 *(ბ)*);
- შემოავლე წრეწირი დიამეტრით $[B, A_2]$ (ნახ. 25 *(გ)*);
- A_1 წერტილიდან აღმართე (A_1, A_2) წრფის პერპენდიკულარული წრფე (ნახ. 25 *(დ)*);

  შედეგი: ამ წრფისა და წრეწირის გადაკვეთის წერტილი P (ნახ. 25 *(დ)*).

- გამოიტანე პასუხი: ორი წერტილი A_1 და P.



ნახ. 25: $\sqrt{|A_1, A_2|}$ სიგრძის მონაკვეთის აგების პროცესი

სავარჯიშო 2.36: სამკუთხედების მსგავსებით დაამტკიცეთ, რომ $|A_1, P| = \sqrt{|A_1, A_2|} = \sqrt{\xi}$.

სავარჯიშო 2.37: მოცემულია ორი წერტილი A და B. რა ალგორითმით შეიძლება წრეწირის შემოვლება, რომლის დიამეტრია $[A, B]$?

სავარჯიშო 2.38: მოცემულია სამი წერტილი O, A და B. O წერტილიდან გამოდის ორი სხივი $[O, A[$ და $[O, B[$, რომელიც O წერტილში ქმნის კუთხეს α. დაწერეთ ალგორითმი, რომელიც O, A და B მონაცემზე პასუხად მოგვცემს სამ წერტილს O, A და C ისე, რომ $\angle AOC = \frac{\alpha}{2}$.

ანტიკური ამოცანები:

- წრის კვადრატურა: მოცემულია O წერტილი და მის გარშემო შემოვლებული წრეწირი რადიუსით 1. ამ წრის ფართობია π. შეიძლება თუ არა იგივე ფართობის კვადრატის აგება მხოლოდ ფარგლისა და სახაზავის გამოყენებით?
- მესამე ხარისხის ფესვი: მოცემულია ორი წერტილი, რომელთა შორის მანძილია a. შეიძლება თუ არა მხოლოდ ფარგლითა და სახაზავით ისეთი ორი წერტილის აგება, რომელთა შორის მანძილია $\sqrt[3]{a}$?
- სამმაგი ბისექტრისა: მოცემულია სამი წერტილი O, A და B. O წერტილიდან გამოდის ორი სხივი $[O, A[$ და $[O, B[$, რომელიც O წერტილში ქმნის კუთხეს α. შეიძლება თუ არა მხოლოდ ფარგლისა და სახაზავის გამოყენებით ავაგოთ ისეთი წერტილი C, რომ $\angle AOC = \frac{\alpha}{3}$?
- წესიერი მრავალკუთხედები: რამდენკუთხა წესიერი მრავალკუთხედის აგება შეიძლება მხოლოდ ფარგლისა და სახაზავის გამოყენებით? (ამოზნექილ მრავალკუთხედს ეწოდება წესიერი, თუ მისი ყველა გვერდი ერთმანეთის ტოლია.)

როგორც აღმოჩნდა, პირველი სამი ამოცანა ამოუხსნადია: არ არსებობს ისეთი ალგორითმი, რომელიც მხოლოდ ფარგლისა და სახაზავის მეშვეობით ააგებს ორ წერტილს, რომელთა შორის მანძილია π; ან ისეთი ალგორითმი, რომელიც ნებისმიერი a რიცხვიდან მესამე ხარისხის ფესვს ამოიღებს ან ისეთი ალგორითმი, რომელიც ნებისმიერ კუთხეს სამად გაყოფს (ისე, როგორც ის ალგორითმი, რომელიც ნებისმიერი რიცხვიდან კვადრატულ ფესვს ამოიღებს ან ნებისმიერ კუთხეს ორად გაჰყოფს).

ამის დამტკიცების იდეა შემდეგია:

ახალი წერტილის აგება შეიძლება მხოლოდ როგორც უკვე აგებულ წერტილებზე გავლებული ორი წრფის, ორი წრეწირისა ან ერთი წრეწირისა და ერთი წრფის გადაკვეთის წერტილისა. თუ ავაგებთ ორი გეომეტრიული ფიგურის გადაკვეთის წერტილს, მაშინ მისი დაშორება კოორდინატთა სათავიდან გამოითვლება შემდეგი პოლინომიური განტოლების ამონახსნით: $a_x^{2^n} + a_{2^n-1}x^{2^n-1} + \cdots + a_1x + a_0 = 0$, სადაც n რაღაცა ნატურალური რიცხვია.
რადგან $\sqrt[3]{a}$ არ არის ასეთი სახის პოლინომის (ანუ ორის ხარისხის რიგის პოლინომის) ამონახსნი, ამიტომ ამ რიცხვის ფარგლითა და სახაზავით აგება შეუძლებელია.
როგორც XIX საუკუნეში გერმანელმა მათემატიკოსმა ლინდემანმა დაამტკიცა, π ტრანსცენდენტული რიცხვია, ანუ იგი არ არის არანაირი პოლინომიური განტოლების ამონახსნი და მით უმეტეს ვერ იქნება ორის ხარისხის რიგის განტოლების ამონახსნი, რითაც მტკიცდება, რომ ფარგლითა და სახაზავით π რიცხვის აგება შეუძლებელია.

მაგრამ არსებობს ფორმულა, რომელიც გვეუბნება, თუ რამდენ კუთხა წესიერი მრავალკუთხედის აგება შეიძლება მხოლოდ ფარგლისა და სახაზავის გამოყენებით: n კუთხა წესიერი მრავალკუთხედის აგება შეიძლება მაშინ და მხოლოდ მაშინ, თუ $\exists m, q_1, ..., q_l \in \mathbb{N}_0$ ისე, რომ $n = 2^m \cdot (2^{2^{q_1}} + 1) \cdot (2^{2^{q_2}} + 1) \cdots (2^{2^{q_l}} + 1)$.
ამ ფორმულიდან გამომდინარე შეიძლება წესიერი ხუთკუთხედის, ცხრამეტკუთხედისა და 65537 კუთხედის აგება, მაგრამ არ შეიძლება წესიერი 7-კუთხედის აგება.

სავარჯიშო 2.39: შეადგინეთ ალგორითმი, რომლის მეშვეობითაც შეიძლება წესიერი ექვსკუთხედის აგება.

სავარჯიშო 2.40: შეადგინეთ ალგორითმი, რომლის მეშვეობითაც შეიძლება წესიერი რვაკუთხედის აგება.

სავარჯიშო 2.41: შეადგინეთ ალგორითმი, რომლის მეშვეობითაც შეიძლება წესიერი ხუთკუთხედის აგება.

შენიშვნა: ზემოთ მოყვანილი ამოცანებისათვის არ არსებობს ალგორითმი, რომელიც მხოლოდ ფარგლითა და სახაზავით აგვაგებინებდა საჭირო წერტილებსა და ფიგურებს. ეს კი იმას არ ნიშნავს, რომ არ არსებობს სხვა რაიმე მეთოდი (თუ არ შევიზღუდებით მხოლოდ ფარგლითა და სახაზავით), რითაც ამ ამოცანებს გადავჭრით.

ღია ამოცანა: წესიერი მრავალკუთხედის ზემოთ მოყვანილ ფორმულაში $2^{2^q}+1$ ე.წ. ფერმას მარტივი რიცხვია. დიდ ხანს ეგონათ, რომ ეს ფორმულა მხოლოდ მარტივ რიცხვებს იძლეოდა, მაგრამ აღმოჩნდა, რომ ეს ასე არაა. უფრო მეტიც: ეს ფორმულა ძირითადად შედგენილ რიცხვებს იძლევა. მაგრამ მნიშვნელოვანია შემდეგი საკითხი: სასრულია თუ არა ფერმას მარტივ რიცხვთა სიმრავლე? ან, სხვა სიტყვებით რომ ვთქვათ, შეგვხვდება თუ არა მიმდევრობაში $(2^{2^q}+1)_{q=0}^{\infty}$ უსასრულოდ ბევრი მარტივი რიცხვი? ამ შეკითხვაზე პასუხი ჯერ-ჯერობით უცნობია.

3 ანბანი და ენა

განვიხილოთ ქართული სიტყვები „ანბანი” და „ენა”. ეს ქართული ენის სიტყვებია, რომელთაც ენაში რაღაცა მნიშვნელობა (სემანტიკა) აქვს. სხვა საქმეა „გჰჭპყ” - ეს ქართული ენის სიტყვა არაა, თუმცა ქართული ანბანით კი არის ჩაწერილი. ამითი განსხვავდება ერთმანეთისაგან „ენის სიტყვა” და „ენის ანბანით ჩაწერილი სიტყვა”.

„ენის ანბანით ჩაწერილი სიტყვა” ამ ენის ანბანის ასოების მიმდევრობაა, რომელსაც რაღაცა სემანტიკური დატვირთვა (ანუ აზრი) შეიძლება ჰქონდეს, ან არ ჰქონდეს. რაიმე ანბანით ჩაწერილი სიტყვა შეიძლება იყოს სასრული, ან უსასრულო. როგორც წესი, ჩვენს ყოველდღიურობაში მხოლოდ სასრული სიტყვები გვხვდება. სასრული სიტყვა სასრული ზომისაა, რაც მასში შემავალი ასოების რაოდენობით განისაზღვრება.

მაგალითად, | ანბანი | = 6 და | ენა | = 3. თუ მოცემულია რაღაცა სიტყვა $w = w_1 w_2 ... w_n$, მისი სიგრძე (ანუ ასოების რაოდენობა) შემდეგნაირად აღინიშნება: $|w| = n$. $w(i)$ ამ სიტყვის i-ური ასოა. ასე, მაგალითად, „ანბანი”(4) = „ა” და „ელექტროფიკაცია”(7) = „ო”.

თუ მოცემულია ორი სიტყვა w_1 და w_2, მაშინ $w_1 \circ w_2 = w_1 w_2$ (ეს ორი სიტყვა ერთი მეორეს მიყოლებით). მაგალითად, „ფეხ”$\circ$„ბურთი”=„ფეხბურთი”. თუ რაღაცა სიტყვა $w = u \circ v$, მაშინ ამბობენ, რომ u სიტყვა w სიტყვის პრეფიქსია, ხოლო v სიტყვა w სიტყვის სუფიქსია: $u \prec w$ და $v \succ w$. w სიტყვის n ასოიანი პრეფიქსი აღინიშნება როგორც $w[n]$, ხოლო მისი m ასოიანი სუფიქსი კი აღინიშნება როგორც $w\{m\}$ (არ აგერიოთ $w(n)$ -ში!!!).

სავარჯიშო 3.1: რას ნიშნავს შემდეგი ჩანაწერები: $|w|$, $w[|w|]$, $w(|w|)$, $w[|w|-1]$, $w[0]$, $w\{|w|\}$, $w\{|w|\}$, $w\{|w|-1\}$, $w\{0\}$?

სავარჯიშო 3.2: რას ნიშნავს შემდეგი ჩანაწერები: $|w|$, $w[|w|]$, $w(|w|)$, $w[|w|-2]$, $w[0]$, $w\{|w|\}$, $w\{|w|\}$, $w\{|w|-3\}$, $w\{0\}$, თუ $w =$„ელექტროფიკაცია” ?

სავარჯიშო 3.3: მოცემულია რაღაცა ანბანი A და ორი სიტყვა $w_1 \in A^m$ და $w_2 \in A^n$. რისი ტოლია $|w_1 \circ w_2|$?

თუ $Q = \{$ა, ბ, გ, დ, ... , ჯ, ჰ$\}$ ქართული ანბანია, მაშინ Q^n ყველა იმ სიტყვის სიმრავლეა, რომელიც ქართულ ანბანზეა შედგენილი და რომელთა ასოების რაოდენობაა (ანუ სიგრძეა) n: $Q^n = \{w \mid w(i) \in Q, (1 \leq i \leq n), |w| = n\}$. Q^* ყველა იმ სასრული სიტყვის სიმრავლეა, რომელიც Q ანბანის ასოებითაა შედგენილი:

$$Q^* = \bigcup_{i=1}^{\infty} Q^i = Q^1 \cup Q^2 \cup \cdots \cup Q^n \cup \cdots.$$

სასრული და უსასრულო სიგრძის სიტყვების გარდა არსებობს კიდევ ე.წ. „ცარიელი სიტყვა” ϵ, ანუ ისეთი სიტყვა, რომელიც არც ერთი ასოსაგან არ შედგება (ცარიელია). ცხადია, რომ $|\epsilon| = 0$, $\epsilon \prec w$ და $w \circ \epsilon = \epsilon \circ w = w$ ნებისმიერი w სიტყვისათვის.

ყველაფერი ზემოთ თქმული შეგვიძლია ჩამოვაყალიბოთ ერთ განმარტებაში:

განმარტება 3.1: ნებისმიერი სასრული სიმრავლე A შეგვიძლია განვიხილოთ, როგორც ანბანი. ამ ანბანზე შექმნილი სიტყვაა ამ ანბანის ელემენტების (ანუ ასოების) მიმდევრობა. თუ w რაიმე A ანბანზე შექმნილი სიტყვაა, $|w|$ ამ სიტყვაში შემავალი ასოების რაოდენობაა. თუ $|w| = 0$, ასეთ სიტყვას ცარიელი ეწოდება და მას აღნიშნავენ სიმბოლოთი ϵ. თუ $|w| = \infty$, ასეთ სიტყვას ეწოდება უსასრულო. თუ მოცემულია ორი სიტყვა w და v, მაშინ $w \circ v = wv$ ამ ორი სიტყვის შერწყმაა. ამბობენ, რომ u სიტყვა w სიტყვის პრეფიქსია ($u \prec w$), თუ $\exists v$ სიტყვა ისეთი, რომ $w = u \circ v$. ანალოგიურად, u სიტყვა w სიტყვის სუფიქსია, ($u \succ w$), თუ $\exists v$ სიტყვა ისეთი, რომ $w = v \circ u$. თუ w რაიმე სიტყვაა, მაშინ $w(n)$ მისი მე-n-ე ასოა, $w[n|$ მისი n ასოსაგან შემდგარი პრეფიქსი, ხოლო wn კი - მისი n ასოსაგან შემდგარი სუფიქსი.
თუ მოცემულია A ანბანი, მაშინ $A^n = \{w \mid |w| = n\}$ და $A^* = \{w \mid |w| < \infty\}$

სავარჯიშო 3.4: მოცემულია ორი სიტყვა $w_1 \in A^*$ და $w_2 \in B^*$, სადაც A და B რაღაცა ანბანებია. რა ანბანის სიტყვაა $w_1 \circ w_2$?

სავარჯიშო 3.5: მოცემულია სიტყვები $w_1 = 00134$, $w_2 = 65430$, $w_3 = 001$, $w_4 = 346$. ჭეშმარიტია თუ არა შემდეგი გამონათქვამი: $w_3 \circ w_4 = w_1 \circ w_4[6]$? პასუხი დაამტკიცეთ.

ამბობენ, რომ $w \in A^*$ სიტყვა $v \in A^*$ სიტყვას შეიცავს, თუ $\exists w_1, w_2 \in A^*$ და $w = w_1 \circ v \circ w_2$ (w_1 ან w_2 ცარიელიც შეიძლება იყოს). ამ შემთხვევაში იტყვიან, რომ v სიტყვა w სიტყვის ქვესიტყვაა. მაგალითად, თუ გვაქვს სიტყვა

„მოდიფიკაცია", მაშინ მისი ქვესიტყვებია „დიფიკა", „კაცი", „მოდი", „კაცია". ამას გარდა, „მოდი" მისი პრეფიქსია, ხოლო „კაცია" კი - სუფიქსი. მაგრამ „მოდიკაცია" მისი ქვესიტყვა არაა, თუმცა შედგება ორი ქვესიტყვისაგან.

სავარჯიშო 3.6: ჭეშმარიტია თუ არა შემდეგი გამონათქვამები: $w \in A^{|w|}$, $w \in A^{|w|-1}$, $w[k] \in A^k$ თუ w სიტყვა A ანბანზეა შედგენილი და $k \in \mathbb{N}$? პასუხები დაამტკიცეთ.

ანალოგიურად სიტყვები შეიძლება შევადგინოთ ნებისმიერ სხვა ანბანზე, ანუ სასრულ სიმრავლეზე. მაგალითად, თუ მოცემულია ათობითი ანბანი $A = \{0,1,2,3,4,5,6,7,8,9\}$, მასში შეიძლება ყველა ნატურალური რიცხვი ჩაიწეროს. ასეთ ჩანაწერს „რიცხვის ათობითი ჩანაწერი" ეწოდება, რადგან მის გამოსახატავად (ჩასაწერად) მხოლოდ ეს 10 ასო, ანუ ციფრი გამოიყენება.

როგორც აღმოჩნდა, შეიძლება უსასრულოდ ბევრი ანბანის შექმნა. თუ M_1 და M_2 სხვადასხვა ანბანებია, არსებობს ბიექტიური ასახვა $f : M_1^* \to M_2^*$, რაც იმას ნიშნავს, რომ ანბანის შერჩევას მნიშვნელობა არ აქვს: რაც ერთი ანბანით ჩაიწერება, იგივე სხვა ნებისმიერი ანბანითაც შეიძლება ჩაიწეროს.

მაგალითად, ქართული ანბანის სიტყვები ათობითი ანბანით შემდეგნაირად შეიძლება ჩაიწეროს:

პირველ რიგში ქართული ანბანის თითო ასო ათობითი ანბანის სიტყვებად უნდა ჩავწეროთ:

ა $\to$ 00	ბ $\to$ 01	გ $\to$ 02	დ $\to$ 03	ე $\to$ 04	ვ $\to$ 05	ზ $\to$ 06	თ $\to$ 07	ი $\to$ 08	კ $\to$ 09
ლ $\to$ 10	მ $\to$ 11	ნ $\to$ 12	ო $\to$ 13	პ $\to$ 14	ჟ $\to$ 15	რ $\to$ 16	ს $\to$ 17	ტ $\to$ 18	უ $\to$ 19
ფ $\to$ 20	ქ $\to$ 21	ღ $\to$ 22	ყ $\to$ 23	შ $\to$ 24	ც $\to$ 25	ჩ $\to$ 26	ძ $\to$ 27	წ $\to$ 28	ჭ $\to$ 29
ხ $\to$ 30	ჯ $\to$ 31	ჰ $\to$ 32							

შემდეგ ქართული ანბანით ჩაწერილი ყოველი სიტყვის ასო შესაბამისი ორეულით უნდა შევცვალოთ. მაგალითად, „კონსპექტი" შემდეგნაირად ჩაიწერება: „091312171404211808".

სავარჯიშო 3.7: როგორ ჩაიწერება ამ მეთოდებით სიტყვა „ელექტროფიკაცია" ? რომელი ქართული სიტყვაა ჩაწერილი სიტყვით „02001113250300" ?

თუ ცნობილია, რომელ ასოს რომელი ციფრების წყვილი (ორეული) შეესაბამება, ადვილი გამოსაანგარიშებელია ქართული სიტყვა. მაგრამ თუ ათობითში ჩაწერილი ეს სიტყვა ვინმეს ჩაუვარდა ხელში, ვინც არ იცის, თუ რომელ რიცხვს რომელი ასო შეესაბამება, ქართული სიტყვის აღდგენა გაძნელდება. ძალიან ძნელი იქნება საწყისი სიტყვის აღდგენა, თუ ჩვენ ასოებს ორ ნიშნა რიცხვებს შევუსაბამებთ ისე, როგორც ამას ჩვენ მოვინდომებთ, მაგალითად:

ა $\to$ 39	ბ $\to$ 27	გ $\to$ 99	დ $\to$ 03	ე $\to$ 38	ვ $\to$ 21	ზ $\to$ 76	თ $\to$ 78	ი $\to$ 87	კ $\to$ 90
ლ $\to$ 10	მ $\to$ 11	ნ $\to$ 13	ო $\to$ 31	პ $\to$ 37	ჟ $\to$ 65	რ $\to$ 16	ს $\to$ 17	ტ $\to$ 18	უ $\to$ 19
ფ $\to$ 47	ქ $\to$ 51	ღ $\to$ 66	ყ $\to$ 08	შ $\to$ 24	ც $\to$ 25	ჩ $\to$ 26	ძ $\to$ 00	წ $\to$ 01	ჭ $\to$ 09
ხ $\to$ 81	ჯ $\to$ 06	ჰ $\to$ 32							

თუ ცნობილია, რომელ ასოს რა ორეული შეესაბამება (მაგალითად ისე, როგორც ზედა ცხრილშია მოყვანილი), შეგვიძლია რაღაცა $f : Q \to A \times A$ ფუნქციის შედგენა (აქ Q ქართული ანბანია და $A = \{0,1,2,...,9\}$). თუ ეს ფუნქცია შედგენილია ზედა ცხრილის საშუალებით, მაშინ $f(\text{ა}) = 39$, $f(\text{ბ}) = 27$, $f(\text{ლ}) = 10$, $f(\text{შ}) = 24$ და ა.შ.

რაღაცა სიტყვა $w \in Q^n$ კი შემდეგი რეკურსიული ალგორითმით შეიძლება ჩავწეროთ ათობითი ანბანის გამოყენებით:

ალგორითმი $P(w)$

მონაცემი: $w \in Q^{|w|}$.

- თუ $w = \epsilon$, ალგორითმი დაასრულე;
- პასუხად გამოიტანე სიტყვა „$P(w[|w|-1]) \circ f(w(|w|))$"

  (აქ f ფუნქცია ზემოთ მოყვანილი ცხრილითაა განსაზღვრული).

ამ ალგორითმში ორი სიახლეა შემოტანილი:

1. ეს ალგორითმი უფრო ფორმალურადაა ჩაწერილი, ვიდრე აქამდე მოყვანილ ყველა მაგალითში: ჩანაწერი „ალგორითმი $P(w)$" ნიშნავს, რომ ამ ალგორითმს სახელად ეწოდება P, ხოლო მონაცემად (ან, სამეცნიერო ტერმინოლოგია რომ ვიხმაროთ, არგუმენტად) მოცემული აქვს სიტყვა w.

2. იმის მაგივრად, რომ სიტყვიერად ვწეროთ: „ჩაატარე იგივე ოპერაციები $w[|w|-1]$ მონაცემისათვის, ჩვენ ვწერთ $P(w[|w|-1])$ (რადგან ამ ალგორითმს ეწოდება P, ამიტომ $P(w[|w|-1])$ ნიშნავს: ჩაატარე ალგორითმი P მონაცემით $w[|w|-1]$.

სავარჯიშო 3.8: დაწვრილებით აღწერეთ P(„ხელი") ალგორითმის მსვლელობა (რას აკეთებს ყოველ ბიჯში).

თუ ქართულ სიტყვებს ბოლოს მოყვანილი ცხრილის მიხედვით ჩავწერთ ათობით ანბანში, მაშინ საწყისი ტექსტის აღდგენა საკმაოდ გაძნელდება, თუ ასოებსა და ციფრთა ორეულებს შორის შესაბამისობები ცნობილი არ არის.

სავარჯიშო 3.9: რომელი ქართული სიტყვაა კოდირებული ზემოთ მოყვანილი ცხრილის მიხედვით ათობით ანბანზე შედგენილ სიტყვაში „99001131250300" ? როგორ შეიძლება ჩავწეროთ სიტყვა „წყალი" ?

სავარჯიშო 3.10: მოცემულია ქართული ანბანი Q და ათობითი ანბანი A. თუ $w \in Q^n$ და $v \in A^*$ w სიტყვის შესაბამისი ჩანაწერია ათობით ანბანში

აღსანიშნავია, რომ არსებობს მეთოდები, რომელთა საშუალებითაც შეიძლება ზედა ცხრილის მიხედვით კოდირებული ტექსტის გახსნა იმის და მიუხედავად, თუ ცხრილი ცნობილი არ არის: თუ ვიცით, რომ კოდირებულია ქართული ტექსტი, მოვძებნით იმ ორეულს, რომელიც ყველაზე ხშირად გვხვდება. რადგან ქართულ ენაში ყველაზე ხშირია ასო „ე", ამიტომ სავარაუდოა, რომ ის ორეულიც „ე" ასოს შესაბამისი იქნება. შემდეგ დავითვლით იმ ოთხეულების რაოდენობას, რომელიც „ე" ასოს შესაბამისი ორეულით იწყება. ქართულ ენაში გამოკვლეულია, თუ რომელი ასო გვხვდება ყველაზე ხშირად „ე" ასოს შემდეგ. ანალოგიურად და რამოდენიმე ექსპერიმენტის ჩატარების შედეგად ტექსტის გაშიფვრა შესაძლებელია.

მონაცემთა ანდაგვარი კოდირებითა და გახსნით დაკავებულია ინფორმატიკისა და მათემატიკის ერთ-ერთი განხრა - კრიპტოგრაფია.

თუ მოცემულია რაიმე ანბანი M, მაშინ $L \subset M^*$ ამ ანბანზე შედგენილი ენა ეწოდება.

ანბანსა და ენას ცენტრალური როლი ენიჭება ინფორმატიკაში, რადგან დამტკიცდა, რომ ნებისმიერი ამოცანა შეიძლება რაღაცა ენაში ჩაიწეროს და მისი ამოხსნის ძიება ამ ენაში გარკვეული სიტყვების ძიების ტოლფასია.

ინფორმატიკაში ძალიან მნიშვნელოვანია ე.წ. „ორობითი ანბანი" $\mathbb{B} = \{0, 1\}$. ნებისმიერი ინფორმაცია შეიძლება ჩაიწეროს ამ ანბანის სიტყვებით, ანუ ორობით კოდში.

მაგალითად, თუ მოცემულია რაიმე ნატურალური რიცხვი $n \in \mathbb{N}$, მისი ჩაწერა ორობით კოდში შემდეგი ალგორითმით შეიძლება:

მოცემულია: $n \in \mathbb{N}$.

- თუ $n = 0$, ალგორითმი დაასრულე.
- ამობეჭდე $\frac{n}{2}$ გაყოფისას მიღებული ნაშთი

  (თუ n კენტია, ამობეჭდე „1") ;

  (თუ n ლუწია, ამობეჭდე „0");
- ეს პროცედურა გაიმეორე $\lfloor \frac{n}{2} \rfloor$ მონაცემისათვის .

მაგალითად, თუ $n = 5$, ალგორითმი შემდეგნაირად იმუშავებს:

მოცემულია: $n = 5$.

- თუ $n = 0$, ალგორითმი დაასრულე - (ამ შემთხვევაში ეს არ სრულდება)
- თუ n კენტია, ამობეჭდე „1" - (ამ შემთხვევაში ეს სრულდება: 5 კენტია) ;

  ამობეჭდილი რიცხვი: "1"
- თუ n ლუწია, ამობეჭდე „0" - (ამ შემთხვევაში ეს არ სრულდება) ;

- ეს პროცედურა გაიმეორე $\lfloor\frac{5}{2}\rfloor = 2$ მონაცემისათვის :

  მოცემულია: $n = 2$.

- თუ $n = 1$, ამობეჭდე „1" და ალგორითმი დაასრულე - (ამ შემთხვევაში ეს არ სრულდება)

- თუ n კენტია, ამობეჭდე „1" - (ამ შემთხვევაში ეს არ სრულდება: 2 ლუწია) ;

  ამობეჭდილი რიცხვი: "01"

- თუ n ლუწია, ამობეჭდე „0" - (ამ შემთხვევაში ეს სრულდება: 2 ლუწია) ;

- ეს პროცედურა გაიმეორე $\lfloor\frac{2}{2}\rfloor = 1$ მონაცემისათვის :

  მოცემულია: $n = 1$.

- თუ $n = 0$, ალგორითმი დაასრულე - (ამ შემთხვევაში ეს არ სრულდება)

- თუ n კენტია, ამობეჭდე „1" - (ამ შემთხვევაში ეს სრულდება: 1 კენტია) ;

  ამობეჭდილი რიცხვი: "101"

- ეს პროცედურა გაიმეორე $\lfloor\frac{1}{2}\rfloor = 0$ მონაცემისათვის :

  მოცემულია: $n = 1$.

- თუ $n = 0$, ალგორითმი დაასრულე - (ამ შემთხვევაში ეს სრულდება).

  ალგორითმი დასრულდა.

სავარჯიშო 3.11: გადაიყვანეთ ორობით კოდში შემდეგი რიცხვები: 13, 127, 17, 8, 16, 0.

ანალოგიურად შეიძლება ნებისმიერი რიცხვის ნებისმიერი ანბანით ჩაწერა. თუ მოცემულია k ასოიანი ანბანი, მაშინ იტყვიან, რომ მისი სიტყვები ჩაწერილია k ბაზით:

მოცემულია: $n \in \mathbb{N}$ (ჩასაწერი რიცხვი) და $k \in \mathbb{N}$ (ბაზა).

- თუ $n = 0$, ალგორითმი დაასრულე.

- ამობეჭდე $\frac{n}{k}$ გაყოფისას მიღებული ნაშთი

- ეს პროცედურა გაიმეორე $\lfloor\frac{n}{k}\rfloor$ მონაცემისათვის .

სავარჯიშო 3.12: წინა სავარჯიშოში მოყვანილი რიცხვები ჩაწერეთ რვაობით, თექვსმეტობით და ორობით კოდებში.

სავარჯიშო 3.13: დაწერეთ ალგორითმი, რომელიც ორობით კოდში ჩაწერილ რიცხვს ათობით კოდში გადაიყვანს.

3.1 მომგებიანი სტრატეგია თამაშებში

3.1.1 თამაში ასანთებით:

მოცემულია ასანთების სამი გროვა. პირველ გროვაშია x_1 ასანთი, მეორეში x_2 და მესამეში x_3.
ორი მოთამაშე რიგ-რიგობით იღებს რამოდენიმე ასანთს ერთი და მხოლოდ ერთი გროვიდან. მოგებულია ის მოთამაშე, რომელიც ბოლოს აიღებს ასანთს და მოწინააღმდეგეს არაფერი აღარ დარჩება.

მაგალითად, პირველ გროვაშია 3 ასანთი, მეორეში 9 და მესამეში 6.

მოცემულია: $x_1 = 3$, $x_2 = 9$, $x_3 = 6$.

- პირველი მოთამაშე იღებს 3 ასანთს მეორე კონიდან: $x_2 = x_2 - 3$. დარჩება: $x_1 = 3$, $x_2 = 9 - 3 = 6$, $x_3 = 6$.
- მეორე მოთამშე ისევე მეორე კონიდან იღებს 2 ასანთს: $x_2 = x_2 - 2$. დარჩება: $x_1 = 3$, $x_2 = 6 - 2 = 4$, $x_3 = 6$.
- პირველი მოთამაშე იღებს 1 ასანთს მესამე კონიდან: $x_3 = x_3 - 1$. დარჩება: $x_1 = 3$, $x_2 = 4$, $x_3 = 6 - 1 = 5$.
- მეორე მოთამშე პირველი კონიდან იღებს 3 ასანთს: $x_1 = x_1 - 3$. დარჩება: $x_1 = 3 - 3 = 0$, $x_2 = 4$, $x_3 = 5$.
- პირველი მოთამაშე იღებს 1 ასანთს მესამე კონიდან: $x_3 = x_3 - 1$. დარჩება: $x_1 = 0$, $x_2 = 4$, $x_3 = 5 - 1 = 4$.
- მეორე მოთამშე მეორე კონიდან იღებს 3 ასანთს: $x_2 = x_2 - 3$. დარჩება: $x_1 = 0$, $x_2 = 4 - 3 = 1$, $x_3 = 4$.
- პირველი მოთამაშე იღებს 3 ასანთს მესამე კონიდან: $x_3 = x_3 - 3$. დარჩება: $x_1 = 0$, $x_2 = 1$, $x_3 = 4 - 3 = 1$.
- მეორე მოთამშე მეორე კონიდან იღებს 1 ასანთს: $x_2 = x_2 - 1$. დარჩება: $x_1 = 0$, $x_2 = 1 - 1 = 0$, $x_3 = 1$.
- პირველი მოთამაშე იღებს 1 ასანთს მესამე კონიდან: $x_3 = x_3 - 1$. დარჩება: $x_1 = 0$, $x_2 = 0$, $x_3 = 1 - 1 = 0$.

პირველმა მოთამაშემ მოიგო, რადგან მოწინააღმდეგეს სვლა აღარ დარჩა.

ამ თამაშში მომგებიანი სტრატეგია არსებობს, ანუ ისეთი ალგორითმი, რომლითაც ერთ-ერთი მოთამაშე ყოველთვის მოიგებს.
თითო კონაში ასანთების რაოდენობა x_1, x_2 და x_3 ორობით კოდში ჩავწეროთ: $x_1 = a_1a_2...a_n$, $x_2 = b_1b_2...b_n$, $x_3 = c_1c_2...c_n$. ჩვენს მაგალითში მივიღებთ:
$x_1 = 0011$, $x_2 = 1001$, $x_3 = 0101$.
შენიშვნა: x_2 ოთხი ასოსგან (ბიტისგან) შედგება, x_1 და x_3 რიცხვების ჩასაწერად კი საკმარისია 2 და შესაბამისად 3 ბიტი, მაგრამ ჩვენ სამივე რიცხვს ერთსა და იმავე სიგრძის სიტყვებად ვწერთ: თუ რაიმე ორობითი რიცხვი მოკლეა, მარცხენა მხარეს ნულების დამატებით მათი მნიშვნელობა არ იცვლება.

სამივე რიცხვს ვწერთ ერთმანეთის ქვემოთ და თითოეულ სვეტში ერთიანების რაოდენობას ვითვლით:

$$\begin{array}{cccc} a_1 & a_2 & ... & a_n \\ b_1 & b_2 & ... & b_n \\ c_1 & c_2 & ... & c_n \end{array}$$

თუ ყველა სვეტში ერთიანების რაოდენობა ლუწია, მაშინ პირველი სვლა მოწინააღმდეგეს უნდა დავუთმოთ. ამ შემთხვევაში, თუ მოწინააღმდეგე ერთი კონიდან რამოდენიმე ასანთს აიღებს, ერთიანების რაოდენობე ერთ სვეტში მაინც კენტი იქნება.
თუ ერთ-ერთ სვეტში მაინც ერთიანების რაოდენობა კენტია, ჩვენ ერთ-ერთი კონიდან იმდენი ასანთი უნდა ავიღოთ, რომ ერთიანების რაოდენობა ყველა სვეტში ლუწი გახდეს.
ჩვენს მაგალითში:

$$\begin{array}{cccc} 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{array}$$

რადგან ერთი სვეტი მაინც არსებობს ისეთი, სადაც ერთიანების რაოდენობა კენტია, პირველი სვლა ჩვენი უნდა იყოს.
თუ მეორე კონაში (სტრიქონში) დავტოვებთ რიცხვს 0110, მაშინ ყველა სვეტში ერთიანების რაოდენობა ლუწი გახდება. ამიტომ მეორე კონაში უნდა დავტოვოთ 6 ასანთი (ანუ უნდა ავიღოთ 3).
დაგერჩება:

$$\begin{matrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{matrix}$$

რამდენი ასანთიც არ უნდა აიღოს მოწინააღმდეგემ, აუცილებლად აღმოჩნდება ისეთი სვეტი, სადაც ერთიანების რაოდენობა კენტია. ვთქვათ, პირველი კონიდან მოაკლეს 2 და დაგვრჩა:

$$\begin{matrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{matrix}$$

ერთიანების რაოდენობა მარჯვნიდან მეორე სვეტშია კენტი. ამრიგად, თუ მეორე კონაში დავტოვებთ ოთხ ასანთს, დაგვრჩება:

$$\begin{matrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \end{matrix}$$

მოწინააღმდეგის მიერ რამოდენიმე ასანთის აღება ისევ იგივე ეფექტს გამოიწვევს: ერთ-ერთ სვეტში მაინც გაჩნდება კენტი რაოდენობის ერთიანი. ვთქვათ, მან აიღო მეორე კონიდან ყველა ასანთი:

$$\begin{matrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \end{matrix}$$

თუ ჩვენ მესამე კონაში დავტოვებთ ერთ ასანთს, ერთიანების რაოდენობა კვლავ ყველგან გალუწდება:

$$\begin{matrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$

მოწინააღმდეგე იძულებულია, ერთ-ერთი კონიდან დარჩენილი ერთი ასანთი აიღოს:

$$\begin{matrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{matrix}$$

ბოლო სვლით ჩვენ ვიგებთ.

ამრიგად, გარკვეულ ვითარებებში სასურველია მონაცემთა ორობით კოდში ჩაწერა და შემდეგ ორობით ანბანზე შედგენილი სიტყვებით სტრატეგიის შემუშავება.

სავარჯიშო 3.14: დაწერეთ ალგორითმი, რომელიც ამ თამაშში მომგებიანი სტრატეგიით იმოქმედებს, ანუ მოცემული სამი რიცხვისათვის განსაზღვრავს, თვითონ დაიწყოს თუ არა და შემდეგ ყოველთვის მოიგებს.

3.2 მომგებიანი სტრატეგია კაზინოში:

ვირჩევთ რომელიმე ფერს (მაგალითად, შავს) და ყოველ ჯერზე ვდებთ რაღაცა თანხას. თუ ეს ფერი მოვიდა, ვიგებთ დადებული თანხის ორმაგ რაოდენობას. თუ ჩვენი ფერი არ მოვიდა, დადებული თანხა იკარგება. იმისათვის, რომ ამ თამაშისათვის შევიმუშავოთ მომგებიანი სტრატეგია, უნდა გავითვალისწინოთ რამოდენიმე ზოგადი წესი:

1. პირველ ჯერზე ჩვენს ფერზე ვდებთ a_1 ოდენობის თანხას. ჯამში დახარჯული თანხაა a_1.
2. თუ ჩვენი ფერი მოვიდა, ვიღებთ მოგებულ თანხას $2a_1$ და ყველაფერს ვიწყებთ თავიდან.
3. თუ ჩვენი ფერი არ მოვიდა, მეორე ჯერზე ვდებთ a_2 ოდენობის თანხას. ჯამში ჩადებული თანხა იქნება a_1+a_2.
4. თუ ჩვენი ფერი მოვიდა, ვიღებთ მოგებულ თანხას $2a_2$ და ყველაფერს ვიწყებთ თავიდან.

5. თუ ჩვენი ფერი არ მოვიდა, მესამე ჯერზე ვდებთ a_3 ოდენობის თანხას. ჯამში ჩადებული თანხა იქნება $a_1 + a_2 + a_3$.

   და ასე ვაგრძელებთ მანამ, სანამ არ მოვა ჩვენი ფერი:

6. მე-n-ე ჯერზე ვდებთ a_n ოდენობის თანხას. ჯამში ჩადებული თანხა იქნება $a_1 + a_2 + \cdots + a_{n-1} + a_n$. თუ ჩვენი ფერი მოვიდა, მოგებული თანხა იქნება $2a_n$.

7. რადგან აქამდე ჩადებული თანხა იყო $a_1 + a_2 + \cdots + a_{n-1} + a_n$, სულ მოგებული გვექნება $2a_n - (a_1 + a_2 + \cdots + a_{n-1} + a_n) = a_n - (a_1 + a_2 + \cdots + a_{n-1})$ ოდენობის თანხა.

თუ $2a_n - (a_1 + a_2 + \cdots + a_{n-1} + a_n) = a_n - (a_1 + a_2 + \cdots + a_{n-1}) < 0$, მაშინ დახარჯული თანხა მოგებულზე მეტი იქნება, ანუ თამაშს წავაგებთ. ჩვენი ამოცანაა $a_1, a_2, ..., a_n, ...$ მიმდევრობა ისე შევარჩიოთ, რომ $a_n - (a_1 + a_2 + \cdots + a_{n-1}) > 0$.

ერთი შესაძლებლობაა $a_i = 2^i$. ამ შემთხვევაში $a_n = 2^n$ და $(a_1 + a_2 + \cdots + a_{n-1}) = 2^n - 1$. აქედან გამომდინარე, $a_n - (a_1 + a_2 + \cdots + a_{n-1}) = 2^n - (2^n - 1) = 1$. ესე იგი, ამ სტრატეგიით (ყოველ ჯერზე დადებული თანხის გაორმაგებით) 1 ერთეულს მოვიგებთ.

თუ $(a_i)_{i=1}^{\infty}$ მიმდევრობას ისე შევარჩევთ, რომ $a_n - (a_1 + a_2 + \cdots + a_{n-1}) = n$, მაშინ ჩვენი ფერის მოსვლაზე ვიგებთ იმდენ თანხას, რამდენჯერაც მოგვიწია თანხის დადება.

ახლა გამოვიანგარიშოთ, თუ რა უნდა იყოს $(a_i)_{i=1}^{\infty}$ მიმდევრობა. $a_1 = 1$. a_n მოცემულია რეკურსიული ფორმულით: $a_n - (a_1 + a_2 + \cdots + a_{n-1}) = n$.

სავარჯიშო 3.15: მათემატიკური ინდუქციით დაამტკიცეთ, რომ $a_n = 2^n - 1$ და $a_n = 2 \cdot a_{n-1} + 1$.

ამრიგად, ამ თამაშის მომგებიანი სტრატეგია შემდეგია:

- არჩეულ ფერზე დადე 2·[წინა ჯერზე დადებული თანხა] +1 თანხა
- თუ ეს ფერი მოვიდა, აიღე მოგებული თანხა და თამაში შეწყვიტე.
- თუ ჩვენი ფერი არ მოვიდა, ალგორითმი თავიდან გაიმეორე.

სავარჯიშო 3.16: დაამტკიცეთ ამ სტრატეგიის სისწორე.

4 მიმართებები და დალაგება

განვიხილოთ საქართველოს მოქალაქეთა სიმრავლე $A = \{w \,|\, w$ საქართველოს მოქალაქეა$\}$. რა თქმა უნდა, ამ სიმრავლეში ისეთი ადამიანების ქვესიმრავლეები იქნება, რომლებიც ერთმანეთთან მეგობრობენ. თუ $a, b \in A$ და a მეგობრობს b-თან, მაშინ b მეგობრობს a-თან. იმის აღსანიშნავად, რომ a და b მეგობრობენ, შეგვიძლია დავწეროთ: (a, b). თუ ჩამოვწერთ ყველა ასეთი მეგობრების წყვილებს, მივიღებთ რაღაცა სიმრავლეს $R = \{(a, b) \,|\, a, b \in A,\ a$ და b ერთმანეთთან მეგობრობენ$\}$. ადვილი შესამჩნევია, რომ $(a, b) \in R \Leftrightarrow (b, a) \in R$.
ახლა კი განვიხილოთ იგივე სიმრავლე A და მასზე განსაზღვრული $R_1 = \{(a, b) \,|\, a, b \in A,\ a$ არის b-ს წინაპარი$\}$. ცხადია, რომ თუ $(a, b) \in R_1 \Rightarrow (b, a) \notin R_1$.

თუ მოცემულია ნებისმიერი ორი სიმრავლე A და B, მაშინ $A \times B = \{(a, b) \,|\, a \in A, b \in B\}$ A და B სიმრავლეების **„დეკარტული ნამრავლი"** ეწოდება.

მაგალითად, თუ $A = \{1, 2, 3, 4\}$ და $B = \{a, b, c\}$, მაშინ

$$A \times B = \{(1, a), (1, b), (1, c), (2, a), (2, b), (2, c), (3, a), (3, b), (3, c), (4, a), (4, b), (4, c)\}.$$

აღსანიშნავია, რომ აქ მნიშვნელობა აქვს ელემენტების თანმიმდევრობას: პირველ ადგილზეა A სიმრავლის ელემენტი, ხოლო მეორეზე კი - B სიმრავლისა.

ცხადია, რომ $A \times A$ სიმრავლეც A სიმრავლის თავის თავთან დეკარტული ნამრავლია.

მაგალითად, თუ $A = \{a_1, a_2, a_3\}$, $A \times A = \{(a_1, a_1), (a_1, a_2), (a_1, a_3), (a_2, a_1), (a_2, a_2), (a_2, a_3), (a_3, a_1), (a_3, a_2), (a_3, a_3)\}$.

<u>სავარჯიშო 4.1:</u> განვიხილოთ შემდეგი ამოცანა: მოცემულია $n \in \mathbb{N}$. შეადგინეთ $A \times A$, სადაც $A = \{a_1, a_2, ..., a_n\}$. რა არის ამ ამოცანის მონაცემი? რა უნდა იყოს მისი შედეგი? დაწერეთ ალგორითმი, რომელიც ამ ამოცანას გადაჭრის.

თუ მოცემულია რაიმე სიმრავლე A, მაშინ $R \subset A \times A$ მასზე განსაზღვრული მიმართება ეწოდება.
მაგალითად, ზემოთ განსაზღვრული R (მეგობრობის აღმნიშვნელი) და R_1 (წინაპრების აღმნიშვნელი) სიმრავლეები შესაბამისი A სიმრავლის სხვადასხვა დამოკიდებულებების განმსაზღვრელია.

ორ სხვადასხვა სიმრავლეზე განსაზღვრული მიმართების მაგალითიად შეგვიძლია მოვიყვანოთ საქართველოს რეგიონებისა და ქალაქების სიმრავლეები:
$A = \{$ქართლი, კახეთი, რაჭა, იმერეთი, სამეგრელო$\}$ და $B = \{$ოზურგეთი, ონი, ფოთი, აგარა, ზუგდიდი, ვანი, თელავი, გურჯაანი, ქუთაისი$\}$.
მიართება, რომელიც თითოეულ რეგიონს მასში არსებულ ქალაქს დაუკავშირებს, იქნება:

$$\begin{aligned} R_2 \;=\{ &\quad (\text{ქართლი, აგარა}), (\text{კახეთი, თელავი}), (\text{კახეთი, გურჯაანი}), (\text{რაჭა, ონი}), (\text{იმერეთი, ვანი}), \\ &\quad (\text{იმერეთი, ქუთაისი}), (\text{სამეგრელო, ფოთი}), (\text{სამეგრელო, ზუგდიდი}) \;\; \}. \end{aligned}$$

ამრიგად, გვაქვს შემდეგი განსაზღვრება:

> ნებისმიერი A და B სიმრავლისათვის $R \subset A \times B$ ამ სიმრავლეზე განსაზღვრული მიმართებაა (არაა გამორიცხული, რომ $A = B$).

- თუ $\forall a_1, a_2 \in R$, $(a_1, a_2) \in R$ ან $(a_2, a_1) \in R$, მაშინ R მიმართებას სრული ეწოდება;
- თუ $\forall a \in A$, $(a, a) \in R \subset A \times A$, მაშინ R მიმართებას რეფლექსური ეწოდება;
- თუ $(a_1, a_2) \in R \Leftrightarrow (a_2, a_1) \in R$, მაშინ R მიმართებას სიმეტრიული ეწოდება;
- თუ $(a_1, a_2) \in R \Rightarrow (a_2, a_1) \notin R$, მაშინ R მიმართებას ანტისიმეტრიული ეწოდება;
- თუ $\forall a_1, a_2, a_3 \in R$, $(\,(a_1, a_2) \in R,\ (a_2, a_3) \in R\,) \Rightarrow (a_1, a_3) \in R$, მაშინ R მიმართებას ტრანსიტული ეწოდება.

მაგალითად, ზემოთ განსაზღვრული R (მეგობრობის აღმნიშვნელი) მიმართება სიმეტრიულია, მაგრამ არ არის სრული, რადგან შეიძლება მოიძებნოს ორი ისეთი ადამიანი $a, b \in A$, რომელიც ერთმანეთთან არ მეგობრობს და ამიტომ $(a, b) \notin R$.

მეორე მიმართება R_1 (წინაპრების განმსაზღვრელი) ტრანზიტულია: თუ a-ს წინაპარია b ($(a,b) \in R_1$) და b-ს წინაპარია c ($(b,c) \in R_1$), a-ს წინაპარია c ანუ $(a,c) \in R_1$.

მესამე მიმართება R_2 ანტისიმეტრიული და არასრულია: R_2 არ შეიცავს არც ერთ წყვილს, რომელშიც შედის ოზურგეთი.

სავარჯიშო 4.2: დაამტკიცეთ, რომ მიმართება $R_3 \subset \mathbb{N} \times \mathbb{N}$, $R_3 = \{(a,b) | a \leq b\}$ რეფლექსური და სრულია.

სავარჯიშო 4.3: დაწერეთ, რისი ტოლია შემდეგი სიმრავლეები:

(ა) $\{1\} \times \{1,2\} \times \{1,2,3\}$;

(ბ) $\emptyset \times \{1,2,3\}$;

(გ) $2^{\{1,2\}}$, რაც არის $\{1,2\}$ სიმრავლის ყველა შესაძლო ქვესიმრავლის სიმრავლე;

(დ) $2^{\{1,2\}} \times \{1,2\}$.

თვალსაჩინოებისათვის პატარა სიმრავლეებზე მიმართებები გრაფიკულად შეიძლება გამოვსახოთ: თუ A სიმრავლეზე განსაზღვრულია რაიმე მიმართება R და $(a,b) \in R$, მაშინ a და b ელემენტები გამოისახება რგოლებად, ხოლო $(a,b) \in R$ კი a ელემენტიდან b ელემენტში მიმართული ისრით (ნახ. 26 (ა)). თუ $(b,b) \in R$, ეს გრაფიკულად b ელემენტის შესაბამისი რგოლიდან გამომავალი და იგივე რგოლში შემავალი ისრით გამოიხატება (ნახ. 26 (ბ)).

თუ $A = \{a,b,c,d,e\}$, მაშინ $R = \{(a,a),(a,b),(b,b),(b,a)(c,a),(c,d),(d,e),(e,b),(e,c),(e,d)\}$ ისე შეიძლება წარმოვადგინოთ, როგორც ნახ. 26 (გ)) -ში.

ნახ. 26: მიმართებათა გრაფიკული წარმოდგენა

რეფლექსურ, სიმეტრიულ და ტრანზიტულ მიმართებას „ტოლობის მიმართება" ან „ექვივალენტურობის მიმართება" ეწოდება.

მაგალითად, თუ მოცემულია მსოფლიოს ხალხთა სიმრავლე A, მაშინ $R' = \{(a,b) | a \text{ და } b \text{ ერთი ეროვნების არიან}\}$ ექვივალენტურობის მიმართებაა, რადგან იგი რეფლექსური, სიმეტრიული და ტრანზიტულია.

სავარჯიშო 4.4: დაამტკიცეთ, რომ ზემოთ მოყვანილი მიმართება R' მართლაც რეფლექსური, სიმეტრიული და ტრანზიტულია.

ექვივალენტურობის მიმართება სიმრავლეს ე.წ. „ექვივალენტურობის კლასებად" ჰყოფს, ანუ ისეთ ქვესიმრავლეებად, სადაც ერთმანეთის ექვივალენტური (ანუ გარკვეული თვალსაზრისით მსგავსი) ელემენტები შედის. სხვა სიტყვებით რომ ვთქვათ, თუ რაიმე $A \neq \emptyset$ სიმრავლეზე განსაზღვრულია ექვივალენტურობის მიმართება R, იგი განსაზღვრავს A სიმრავლის ისეთ ქვესიმრავლეებს $B \subset A$, რომ $B = \{a, b \in A \,|\, (a,b) \in R\}$ (ამ ქვესიმრავლეებში მხოლოდ ისეთი ელემენტები შედის, რომლებიც R მიმართების განსაზღვრებით ერთმანეთის „ექვივალენტურია").

მაგალითად, თუ მოცემულია მიმართება $R = \{(a,b) | \, a \text{ და } b \text{ ორივე ლუწია ან } a \text{ და } b \text{ ორივე კენტია}\}$, იგი ნატურალურ რიცხვთა $\mathbb{N}$ სიმრავლეში ორ ქვესიმრავლეს გამოჰყოფს - ლუწ და კენტ რიცხვთა ქვესიმრავლეებს (კლასებს): $N_1 = \{a_i | (a_k, a_l) \in R \text{ და ორივე ლუწია}\}$, $N_2 = \{a_i | (a_k, a_l) \in R \text{ და ორივე კენტია}\}$

თუ მოცემულია $A = \{0,1,2,3,4,5,6,7\}$, მაშინ მიმართება $R = \{(a,b) \in A \times A \,|\, a$ და b ორივე ლუწია ან a და b ორივე კენტია$\}$ გრაფიკულად შემდეგნაირად შეიძლება გამოისახოს:

ნახ. 27: სიმრავლის ორ დამოუკიდებელ კლასად დაყოფის მაგალითი

ადვილი დასანახია, რომ R მიმართება A სიმრავლეში ორ დამოუკიდებელ კლასს (ქვესიმრავლეს) გამოჰყოფს.

აღსანიშნავია, რომ ეს ერთმანეთის ექვივალენტური ანუ ტოლი ელემენტები მოცემული მიმართებითაა განსაზღვრული. სხვა მიმართებას შეიძლება სხვა ექვივალენტური ელემენტები გამოეყო. ამის მაგალითია იგივე A სიმრავლეზე განსაზღვრული $R' = \{(a,b) \,|\, a$ და b ორივე იყოფა 3-ზე ან a და b ორივე არ იყოფა 3-ზე$\}$.

<u>სავარჯიშო 4.5:</u> გრაფიკულად გამოხატეთ ბოლოს მოცემული მიმართება R' ისე, როგორც ეს წინა მაგალითში მოხდა.

რაიმე A სიმრავლის **ექვივალენტურობის კლასები** შემდეგნაირად აღინიშნება: $[a] = \{b \,|\, (a,b) \in R\}$, სადაც R არის A სიმრავლის ექვივალენტურობის მიმართება.

მაგალითად, თუ $A = \{0,1,2,3,4,5,6,7,8,9\}$ და $R' = \{(a,b) \,|\, a$ და b ორივე იყოფა 3-ზე ან a და b ორივე არ იყოფა 3-ზე$\}$, $[6] = \{0,3,6,9\}$ და $[2] = \{1,2,4,5,7,8\}$.

<u>სავარჯიშო 4.6:</u> მოიყვანეთ ნატურალურ რიცხვთა სიმრავლეზე განსაზღვრული ექვივალენტურობის მიმართების მაგალითი, რომელიც სამ ქვესიმრავლეს გამოჰყოფს. თითოეულ ასეთ კლასში ერთმანეთის ექვივალენტური ელემენტებია.

<u>სავარჯიშო 4.7:</u> მოცემულია $A = \{0,1,2,3,4,5,6,7,8,9\}$ და მასზე განსაზღვრული ექვივალენტურობის მიმართება

$R = \{(a,b) \,|\, (a,b) \in R$, თუ:
a და b ორივე იყოფა 2-ზე, მაგრამ არ იყოფა 3-ზე და არ იყოფა 5-ზე და არ იყოფა 7-ზე;
ან
a და b ორივე იყოფა 3-ზე, მაგრამ არ იყოფა 2-ზე და არ იყოფა 5-ზე და არ იყოფა 7-ზე;
ან
a და b ორივე იყოფა 5-ზე, მაგრამ არ იყოფა 2-ზე და არ იყოფა 3-ზე და არ იყოფა 7-ზე;
ან
a და b ორივე იყოფა 7-ზე, მაგრამ არ იყოფა 2-ზე და არ იყოფა 3-ზე და არ იყოფა 5-ზე
$\}$

ამოწერეთ ამ მიმართების ყველა ელემენტი და შემდეგ წარმოარგინეთ იგი გრაფიკულად.

ახლა კი განვიხილოთ ორი ნატურალური რიცხვი, რომელიც ათობით ანბანშია ჩაწერილი: 307 და 509. ჩვენ ვიცით, რომ $307 < 509$. ამ ორი რიცხვის ასეთი მიმართება სადღაც უნდა იყოს განსაზღვრული (ანალოგიურად ჩვენ შეგვეძლო განგვესაზღვრა $509 < 307$. ჩვენ ვიცით, რომ $0 < 1 < 2 < 3 < 4 < 5 < 6 < 7 < 8 < 9$. მაგრამ ამ ციფრების ასეთი მიმართება ცხადი არაა, ესეც ვიღაცის მიერაა დადგენილი და შენდეგ საყოველთაოდ მიღებული.

ამრიგად, გვაქვს შემდეგი მიმართება $A = \{0,1,2,3,4,5,6,7,8,9\}$ სიმრავლეზე:

$$
\begin{aligned}
R \;=\{&\\
&(0,1),(0,2),(0,3),(0,4),(0,5),(0,6),(0,7),(0,8),(0,9)\\
&(1,2),(1,3),(1,4),(1,5),(1,6),(1,7),(1,8),(1,9)\\
&(2,3),(2,4),(2,5),(2,6),(2,7),(2,8),(2,9)\\
&(3,4),(3,5),(3,6),(3,7),(3,8),(3,9)\\
&(4,5),(4,6),(4,7),(4,8),(4,9)\\
&(5,6),(5,7),(5,8),(5,9)\\
&(6,7),(6,8),(6,9)\\
&(7,8),(7,9)\\
&(8,9)\\
\}&
\end{aligned}
$$

ეს მიმართება ე.წ. „დალაგებას" განსაზღვრავს, ანუ გვაძლევს იმის წესს, თუ როგორ შეიძლება დავალაგოთ სიმრავლის ელემენტები ზრდადობის მიხედვით.

განმარტება 4.1: სრულ, ანტისიმეტრიულ და ტრანზიტულ მიმართებას დალაგება ეწოდება. არასრულ, ანტისიმეტრიულ და ტრანზიტულ მიმართებას ნაწილობრივი დალაგება ეწოდება.

სავარჯიშო 4.8: დაამტკიცეთ, რომ ბოლოს მოყვანილი მიმართება R დალაგებაა.

სავარჯიშო 4.9: მოიყვანეთ ზემოთ განსაზღვრულ A სიმრავლეზე ნაწილობრივი დალაგების მაგალითი.

თუ $(a,b)\in R$ და R დალაგებაა, მაშინ ვწერთ: $a<b$.

თუ გვაქვს მოცემული დალაგება ზემოთ მოყვანილ ანბანზე A, ადვილად შეიძლება A^* სიმრავლის სიტყვების დალაგებაც შემდეგი ალგორითმით:

ალგორითმი $C(w,v)$

მოცემულია: $w=(w_n,w_{n-1},...,w_1), v=(v_n,v_{n-1},...,v_1)\in A^*$

- თუ $|w|<|v|$, მაშინ $(w,v)\in R$ (ან, რაც იგივეა, $w<v$) და ალგორითმი დაასრულე.
- თუ $|v|<|w|$, მაშინ $(v,w)\in R$ (ან, რაც იგივეა, $v<w$) და ალგორითმი დაასრულე.
- თუ $|w|=|v|=0$, მაშინ $w=v$ და ალგორითმი დაასრულე.
- თუ $w(|w|)<v(|v|)$, მაშინ $(w,v)\in R$ (ან, რაც იგივეა, $w<v$) და ალგორითმი დაასრულე.
- თუ $w(|v|)<v(|w|)$, მაშინ $(v,w)\in R$ (ან, რაც იგივეა, $v<w$) და ალგორითმი დაასრულე.
- თუ $w(|w|)=v(|v|)$, მაშინ ჩაატარე $C(w\{|w|-1\},v\{|v|-1\})$ (იგივე ალგორითმი w და v სიტყვების სუფიქსებისათვის).

სავარჯიშო 4.10: სიტყვიერად ახსენით, თუ რას ნიშნავს ზედა ალგორითმში მოყვანილი მათემატიკური ჩანაწერები „თუ $w(|w|)<v(|v|)$, მაშინ..." და „$C(w\{|w|-1\},v\{|v|-1\})$".

სავარჯიშო 4.11: დაამტკიცეთ ამ ალგორითმის სისწორე. გამოითვალეთ მისი ბიჯების რაოდენობა, თუ $|w|=|v|$ და შემდეგ თუ $|w|\neq|v|$.

სავარჯიშო 4.12: დაწერეთ, თუ რისი ტოლია ზემოთ მოყვანილ A სიმრავლეზე განსაზღვრული დალაგების სიმრავლე, რომლის მიხედვითაც $1\leq 3\leq 2\leq 5\leq 8\leq 4\leq 0\leq 9\leq 7\leq 6$.

სავარჯიშო 4.13: მოიყვანეთ A სიმრავლეზე განსაზღვრული ორი სხვადასხვა ნაწილობრივი დალაგების მაგალითი. არის თუ არა $R=\emptyset$ ამ სიმრავლის ნაწილობრივი დალაგება?

სავარჯიშო 4.14: დაამტკიცეთ, რომ თუ R_1 და R_2 რაღაცა სიმრავლეზე განსაზღვრული ნაწილობრივი დალაგებებია, მაშინ $R_1\cap R_2$ იგივე სიმრავლეზე განსაზღვრული ნაწილობრივი დალაგებაა.

სავარჯიშო 4.15: მოცემულია ნებისმიერი სიმრავლე S, რომელიც თავის მხრივ რაღაცა სიმრავლეებისაგან შედგება. დაამტკიცეთ, რომ $R_S = \{(A,B) \,|\, A, B \in S, A \subseteq B\}$ ნაწილობრივი დალაგებაა.

სავარჯიშო 4.16: დავუშვათ, $S = 2^{\{1,2,3\}}$, რაც არის $\{1,2,3\}$ სიმრავლის ყველა შესაძლო ქვესიმრავლის სიმრავლე. ამოწერეთ ამ სიმრავლის ყველა ელემენტი და დიაგრამის სახით გამოსახეთ წინა სავარჯიშოში განსაზღვრული მიმართება R_S, რომელიც ამ სიმრავლეზეა განსაზღვრული. ცალკე ამოწერეთ S სიმრავლის მინიმალური ელემენტები, ანუ ისეთი ელემენტები a_i, რომელთათვისაც $(a_i, b) \in R_S, \forall b \in S$.

სავარჯიშო 4.17: როგორ განისაზღვრება ნებისმიერი A სიმრავლის რაღაცა R დალაგების შედეგად მიღებული მაქსიმალური ელემენტები?

დალაგება და ნაწილობრივი დალაგება ცენტრალურ როლს თამაშობს ინფორმატიკაში, რადგან ამოცანათა უდიდესი ნაწილი მონაცემთა რაღაცა წესის მიხედვით დალაგების შედეგად საკმაოდ მარტივდება.

ამის მაგალითია ქართულ ანბანზე Q შემოტანილი დალაგება ა $<$ ბ $<$ გ $<$ დ $< \dots <$ ჯ $<$ ჰ. თუ ჩვენ ამის საფუძველზე ქართულ სიტყვებსაც დავალაგებთ (ანუ შემოვიტანთ დალაგების წესს Q^* სიმრავლეზე), ქართულ ლექსიკონში რაიმე მოცემული w სიტყვის მოძებნა გაადვილდება: ლექსიკონს გადავშლით შუაში და ამოვიკითხავთ პირველივე სიტყვას v. თუ $w = v$, სიტყვა მოძებნილია. თუ ჩვენი საძებნი სიტყვა ამ სიტყვის წინაა (ანუ $w < v$), მაშინ იგივე ოპერაციას გავიმეორებთ ლექსიკონის პირველ ნახევარში (თუ $v < w$, ვიღებთ მეორე ნაწილს): გადავშლით ამ ნაწილის შუაში და ანალოგიურ პროცედურას გავიმეორებთ.

სავარჯიშო 4.18: დაწერეთ ალგორითმი, რომელიც ქართულ ანბანზე განსაზღვრული ორი სიტყვისათვის w და v განსაზღვრავს, $w = v$ თუ $w < v$ თუ $v < w$.
შენიშვნა: ეს ალგორითმი ათობითში ჩაწერილი რიცხვების შედარების ალგორითმის მსგავსია.

სავარჯიშო 4.19: დაამტკიცეთ წინა სავარჯიშოში მოყვანილი ალგორითმის სისწორე და გამოითვალეთ მისი ბიჯების რაოდენობა, თუ $|w| = n$ და $|v| = m$.

ზოგადად, თუ მოცემულია რაიმე A ანბანი და $S = \{u_1, u_2, ..., u_n \in A^*\}$ სიტყვათა სიმრავლე, მოცემული w სიტყვის მოძებნა ამ სიმრავლეში შეიძლება შემდეგი ალგორითმით:

ალგორითმი $L(S, w)$

მოცემულია: $S = \{u_1, u_2, ..., u_n \in A^*\}$ სიტყვათა სიმრავლე და რაღაცა სიტყვა w.

შედეგი: ვიპოვნოთ ისეთი $u_i \in S$, რომ $u_i = w$.

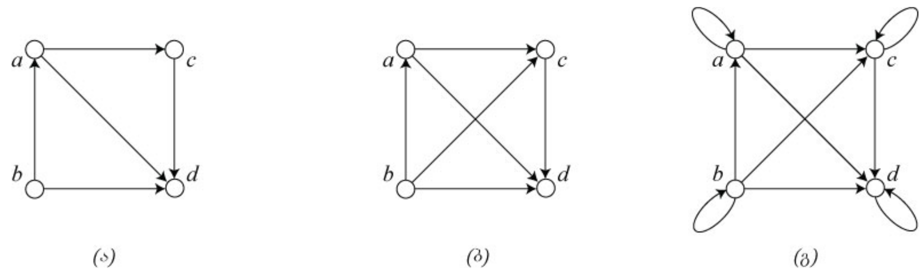
- თუ $S = \emptyset$, მაშინ დაბეჭდე: „სიტყვა სიმრავლეში არ მოიძებნა" და ალგორითმი დაასრულე.
- თუ $u_{\lfloor \frac{|S|}{2} \rfloor} = w$, მაშინ დაბეჭდე: „$i$-ური ელემენტია w" და ალგორითმი დაასრულე.
- $u_{\lfloor \frac{|S|}{2} \rfloor} < w$, მაშინ ჩაატარე $L(\{u_{\lfloor \frac{|S|}{2} \rfloor + 1}, ..., u_n\}, w)$.
- $u_{\lfloor \frac{|S|}{2} \rfloor} > w$, მაშინ ჩაატარე $L(\{u_{\lfloor \frac{|S|}{2} \rfloor}, ..., u_n\}, w)$.

სავარჯიშო 4.20: ინდუქციის გამოყენებით დაამტკიცეთ ამ ალგორითმის სისწორე. გამოითვალეთ მისი ბიჯების რაოდენობა, თუ $|S| = n$.

ახლა კი განვიხილოთ ნახ. 28 (ა) -ში მოყვანილი მიმართება. ადვილი საჩვენებელია, რომ იგი არც რეფლექსური და არც ტრანზიტულია.

სავარჯიშო 4.21: აჩვენეთ, რომ ნახ. 28 (ა) -ში მოყვანილი მიმართება არც რეფლექსური და არც ტრანზიტულია.

ამ მიმართების სიმრავლისათვის რამოდენიმე ახალი წყვილის (ან გრაფიკულად ისრის) ჩამატებით შეიძლება მივიღოთ ტრანზიტული მიმართება (ნახ. 28 (ბ). დამატებით ყველა a ელემენტისათვის (a, a) წყვილის დამატებით კი ეს მიმართება რეფლექსურიც ხდება (ნახ. 28 (გ)).



ნახ. 28: მიმართების რეფლექსური და ტრანზიტული ჩაკეტვა

ანალოგიური პროცედურა - დამატებითი წყვილებით გაფართოვება ისე, რომ ნებისმიერი მიმართება ტრანზიტული და რეფლექსური გახდეს, შეიძლება ნებისმიერ მიმართებაზე ჩავატაროთ. მიღებულ მიმართებას საწყისი მიმართების რეფლექსური და ტრანზიტული ჩაკეტვა ეწოდება.

განმარტება 4.2: ნებისმიერი R მიმართების ტრანზიტული და რეფლექსური ჩაკეტვა R^* ეწოდება ისეთ რეფლექსურ და ტრანზიტულ მიმართებას, რომლისთვისაც $R \subset R^*$ და R^* სიმრავლის ელემენტების რაოდენობა მინიმალურია იმ სიმრავლეების ელემენტების რაოდენობათა შორის, რომლებიც R მიმართებას ქვესიმრავლედ შეიცავენ, ანუ R^* სიმრავლე R სიმრავლიდან რაც შეიძლება ცოტა წყვილის დამატებით უნდა მიიღებოდეს.

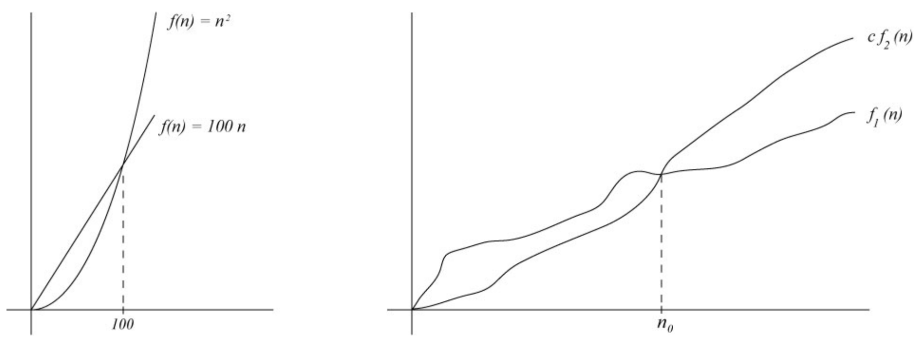
სავარჯიშო 4.22: დაწერეთ ალგორითმი, რომელიც ნებისმიერი A სასრული სიმრავლის რაიმე R მიმართებისათვის მის რეფლექსურ და ტრანზიტულ ჩაკეტვას გამოიანგარიშებს (ანუ შეადგენს შესაბამის სიმრავლეს). დაამტკიცეთ მისი სისწორე და გამოიანგარიშეთ ბიჯების რაოდენობა, თუ $|A| = n$.

5 ალგორითმების სისწრაფის შეფასება

5.1 ფუნქციათა ზრდის რიგი

განვიხილოთ ორი ფუნქცია: $f_1(n) = n^2$ და $f_2(n) = 100\cdot n, n > 0$. ცხადია, რომ $f_2(n) > f_1(n)$, თუ $0 < n < 100$. მაგრამ თუ $n > 100$, მაშინ $f_1(n) > f_2(n)$. ესე იგი, დაწყებული რაღაცა ადგილიდან, $f_1(n) > f_2(n)$ (ნახ. 29 მარცხნივ). ასეთ შემთხვევებში - როდესაც დაწყებული რაღაცა ადგილიდან ერთი ფუნქციის მნიშვნელობა ყოველთვის აჭარბებს მეორე ფუნქციის შესაბამის მნიშვნელობას - ამბობენ, რომ f_1 ფუნქცია უფრო სწრაფად იზრდება, ვიდრე f_2. მაგალითად, $f_1(n) = n$ უფრო სწრაფად იზრდება, ვიდრე $f_2(n) = \log n$ (აქ და შემდგომში $\log n = \log n$, $\ln n = \log_e n$ და $\lg n = \log_{10} n$).

შენიშვნა: აქ და შემდგომში განხილული ფუნქციები დადებითია.



ნახ. 29: ორი ფუნქციის გრაფიკი

სავარჯიშო 5.1: $f_1(n)$ და $f_2(n)$ ფუნქციებს შორის რომელი იზრდება უფრო სწრაფად? (პასუხი დაამტკიცეთ)

1. $f_1(n) = 10 \cdot n^2$, თუ $f_2(n) = 15 \cdot n^2$; 2. $f_1(n) = 0.1 \cdot n^2$, თუ $f_2(n) = n$; 3. $f_1(n) = 10^6 \cdot \log n$, თუ $f_2(n) = n$; 4. $f_1(n) = 10 \cdot \log n^2$, თუ $f_2(n) = 20 \cdot \log n$; 5. $f_1(n) = 2^n$, თუ $f_2(n) = 15^{10} \cdot n^7$.

სავარჯიშო 5.2: დაამტკიცეთ, რომ $f_1(n)$ ფუნქცია უფრო სწრაფად იზრდება, ვიდრე $f_2(n)$, თუ:

1. $f_1(n) = n^2$, $f_2(n) = 15 \cdot n \cdot \log n$; 2. $f_1(n) = n^3$, $f_2(n) = 1983 \cdot n$; 3. $f_1(n) = \log n$, $f_2(n) = 10 \log\log n$; 4. $f_1(n) = \log n^2$, $f_2(n) = 100\sqrt{\log n}$; 5. $f_1(n) = n$, $f_2(n) = \log^7 n$.

გამონათქვამი „დაწყებული რაღაცა ადგილიდან f_1 ფუნქციის მნიშვნელობა ყოველთვის აჭარბებს f_2 ფუნქციის შესაბამის მნიშვნელობას” მათემატიკურად შემდეგნაირად ჩაიწერება: $\exists n_0 \in \mathbb{N},\ \ \forall n > n_0,\ \ f_1(n) > f_2(n)$.

თუ მოცემულია ორი ფუნქცია $f_1(n), f_2(n)$ და $\exists c \in \mathbb{N}$ ისეთი, რომ დაწყებული რაღაცა ადგილიდან $f_1(n) < c \cdot f_2(n)$, მაშინ ამბობენ, რომ $f_1(n)$ ფუნქციის ასიმპტოტური ზრდის რიგი არ აღემატება $f_2(n)$ ფუნქციის ასიმპტოტური ზრდის რიგს.

ამ შემთხვევაში აგრეთვე ამბობენ, რომ f_1 ფუნქციის ზრდის რიგი ზემოდანაა შემოსაზღვრული f_2 ფუნქციის ზრდის რიგით, ანუ f_2 ფუნქციის ზრდის რიგი f_1 ფუნქციის ზრდის რიგის ზედა ზღვარია.

მაგალითად, თუ $f_1(n) = 10 \cdot n$ და $f_2(n) = n$, $f_1(n)$ ფუნქციის ასიმპტოტური ზრდის რიგი არ აღემატება $f_2(n)$ ფუნქციის ასიმპტოტური ზრდის რიგს, რადგან $\exists c = 11$ და $f_1(n) = 10 \cdot n < c \cdot f_2(n) = 11 \cdot n$.
ასიმპტოტური ზრდის რიგი გვიჩვენებს, „დაახლოებით რა სისწრაფით” იზრდება მოცემული ფუნქცია. ზედა მაგალითში შეგვეძლო აგრეთვე დაგვეწერა: $\exists c = 1$ და $c \cdot f_1(n) = 10 \cdot n > f_2(n) = n$. ასე რომ, ერთ შემთხვევაში $f_1(n)$ ფუნქციის ასიმპტოტური ზრდის რიგი არ აღემატება $f_2(n)$ ფუნქციის ასიმპტოტური ზრდის რიგს, მეორე შემთხვევაში კი პირიქით. ასეთ დროს იტყვიან, რომ ამ ორი ფუნქციის ასიმპტოტური ზრდის რიგი ტოლია, ანუ ორივე „დაახლოებით ერთი სისწრაფით იზრდება”. თუ მოცემულია ორი ფუნქცია $f_1(n), f_2(n)$ და $\exists c \in \mathbb{N}$ ისეთი,

რომ დაწყებული რაღაცა ადგილიდან $f_1(n) < c \cdot f_2(n)$, მაგრამ $\nexists d \in \mathbb{N}$ ისეთი, რომ დაწყებული რაღაცა ადგილიდან $f_2(n) < d \cdot f_1(n)$, მაშინ ამბობენ, რომ $f_2(n)$ ფუნქციის ასიმპტოტური ზრდის რიგი უფრო მაღალია, ვიდრე $f_1(n)$ ფუნქციის ასიმპტოტური ზრდის რიგი. ცხადია, რომ თუ $f_1(n)$ და $f_2(n)$ ფუნქციების ასიმპტოტური ზრდის რიგი ტოლია, შეიძლება ასევე ითქვას, რომ $f_1(n)$ ფუნქციის ასიმპტოტური ზრდის რიგი არ აღემატება $f_2(n)$ ფუნქციის ასიმპტოტური ზრდის რიგს (და პირიქით).

ქვემოთ მოყვანილია ცხრილი, რომელიც რამოდენიმე ფუნქციის ზრდის რიგს გვიჩვენებს.

n	$\log n$	n	$n \cdot \log n$	n^2	2^n	$n!$
10	3	10	30	100	1.024	3.628.800
20	4	20	80	400	1.048.576	$\gg 10^{15}$
30	5	30	150	900	1.073.741.824	
40	5	40	200	1600	1.099.511.627.776	
50	6	50	300	2500	$>10^{15}$	
100	7	100	700	10^4	$>10^{30}$	
1.000	10	1.000	10.000	10^6		
10.000	13	10.000	130.000	10^8		
100.000	17	100.000	1.700.000	10^{10}		
1.000.000	20	1.000.000	20.000.000	10^{12}		
10.000.000	23	10.000.000	230.000.000	10^{14}		
100.000.000	27	100.000.000	2.700.000.000	10^{16}		
1.000.000.000	30	1.000.000.000	30.000.000.000	10^{18}		

როგორც ვხედავთ, ამ ფუნქციათა შორის ყველაზე ნელა $f(n) = \log n$ ფუნქცია იზრდება, ყველაზე სწრაფად კი $f(n) = n!$. ამ ბოლო ფუნქციის მნიშვნელობა $n = 20$ -თვის უკვე ძალიან დიდია - როგორც ვარაუდობენ, $2^{100} = 10^{30}$ ჩვენს სამყაროში არსებული ატომების რაოდენობას აღემატება და, აქედან გამომდინარე, 10^{15} ძალიან დიდი რიცხვია.

სავარჯიშო 5.3: დაამტკიცეთ, რომ $f_1(n) = 10n^2$ და $f_2(n) = 10^{-6} \cdot n^2$ ფუნქციათა ასიმპტოტური ზრდის რიგი ტოლია.

სავარჯიშო 5.4: ტოლია თუ არა შემდეგი ორი ფუნქციის ასიმპტოტური ზრდის რიგი (პასუხები დაამტკიცეთ):

1. $f_1(n) = n^2,\ f_2(n) = 15 \cdot n^2 \cdot \log\log n$; 2. $f_1(n) = \log n^3,\ f_2(n) = 1983 \cdot n$; 3. $f_1(n) = \log^2 n,\ f_2(n) = 10 \log n$; 4. $f_1(n) = \log n^2,\ f_2(n) = 100\sqrt{\log n}$; 5. $f_1(n) = n,\ f_2(n) = \log\log^7 n$.

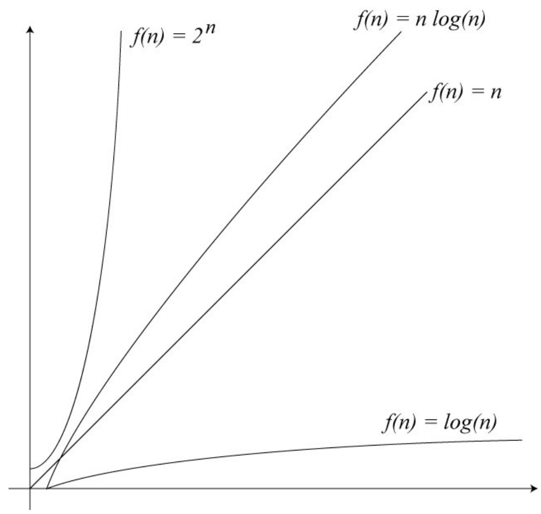
ნახ. 30 გვიჩვენებს რამოდენიმე ფუნქციის ზრდის სისწრაფეს, საიდანაც შეიძლება მათი ასიმპტოტური ზრდის რიგის დანახვა. ყველაზე ნელა იზრდება ლოგარითმული ფუნქცია $f(n) = \log n$; შემდეგია წრფივი ფუნქცია $f(n) = n$. მასზე სწრაფად იზრდება ფუნქცია $n \cdot \log n$ და ყველაზე დიდი ზრდის რიგი აქვს $f(n) = 2^n$ ფუნქციას.

სავარჯიშო 5.5: $f_1(n)$ და $f_2(n)$ ფუნქციებს შორის რომლის ასიმპტოტური ზრდის რიგია უფრო მაღალი?
1. $f_1(n) = \log^2 n,\ f_2(n) = \sqrt{n}$; 2. $f_1(n) = n^3,\ f_2(n) = 1983 \cdot n^2$; 3. $f_1(n) = n \cdot \log n,\ f_2(n) = 2^{\log n}$; 4. $f_1(n) = n^2 \cdot \log n,\ f_2(n) = n^2$; 5. $f_1(n) = \sqrt[3]{n},\ f_2(n) = (\log\log n)^7$.

თუ მოცემულია რაიმე ფუნქცია $f(n)$, შეგვიძლია გამოვყოთ ყველა იმ ფუნქციათა სიმრავლე $O(f(n))$ (იკითხება: ო-დიდი $f(n)$), რომელთა ასიმპტოტური ზრდის რიგი ამ $f(n)$ ფუნქციის ზრდის რიგს არ აღემატება (ანუ ამ სიმრავლეში შემავალი ყველა ფუნქცია ამ ფუნქციის „ქვედა ზღვარია" - დაწყებული რაღაცა ადგილიდან ყოველთვის უფრო ნაკლები იქნება):

$$O(f(n)) = \{g(n) | \exists n_0, c \in \mathbb{N}, \forall n > n_0,\ c \cdot f(n) > g(n)\}.$$

ცხადია, რომ $O(f(n))$ სიმრავლე უსასრულოა, ამიტომ მასში შემავალი ყველა ფუნქციის ამოწერა შეუძლებელია. მაგრამ შესაძლებელია ამ სიმრავლეში შემავალი რამოდენიმე ფუნქციის მაგალითის მოყვანა:



ნახ. 30: რამოდენიმე ფუნქციის გრაფიკი

თუ $f(n) = n$, მაშინ $g(n) = 100 \cdot n \in O(f(n))$, რადგან $\exists c = 101$ და $c \cdot f(n) = 101 \cdot n > 100 \cdot n = g(n)$.
ანალოგიურად შეგვიძლია დავამტკიცოთ: $100n \in O(n \cdot \log n)$, $700n \in O(n^2)$, $200n^2 \in O(2^n)$.

სავარჯიშო 5.6: დაამტკიცეთ, რომ $100n \in O(n \cdot \log n)$, $700n \in O(n^2)$, $200n^2 \in O(2^n)$.

სავარჯიშო 5.7: მოიყვანეთ $O(n \log n)$ სიმრავლის 5 ელემენტი.

ლემა 5.1: თუ $f_1(n)$ ფუნქცია არ იზრდება უფრო სწრაფად, ვიდრე $f_2(n)$ ფუნქცია, მაშინ $O(f_1(n)) \subset O(f_2(n))$.

დამტკიცება: რადგან $f_1(n)$ ფუნქცია არ იზრდება უფრო სწრაფად, ვიდრე $f_2(n)$ ფუნქცია, ამიტომ $\exists d \in \mathbb{N}, f_1(n) < d \cdot f_2(n)$. ახლა განვიხილოთ ნებისმიერი $g(n) \in O(f_1(n))$. განმარტების თანახმად $\exists c \in \mathbb{N}, g(n) < c \cdot f_1(n)$. ზემოთ მოყვანილი უტოლობის თანახმად, $g(n) < c \cdot d \cdot f_2(n)$. ესე იგი, $\exists d \cdot c \in \mathbb{N}$ ისეთი, რომ $g(n) < c \cdot d \cdot f_2(n)$, რაც განმარტების თანახმად ნიშნავს, რომ $g(n) \in O(f_2(n))$.

Q.E.D.

აქედან გამომდინარე, გამონათქვამი „f_1 ფუნქცია არ იზრდება უფრო სწრაფად, ვიდრე f_2 ფუნქცია" შემდეგი მათემატიკური ჩანაწერის ტოლფასია: $O(f_1(n)) \subset O(f_2(n))$.

სავარჯიშო 5.8: მოიყვანეთ $f_1(n)$ და $f_2(n)$ ფუნქციების მაგალითები, რომელთათვისაც $O(f_1(n)) \subset O(f_2(n))$ და, ამავდროულად, $O(f_1(n)) \neq O(f_2(n))$.

ლემა 5.2: $O(f(n))$ სიმრავლეებისათვის ჭეშმარიტია:

1. $O(k \cdot f(n)) = O(f(n))$ ($k \in \mathbb{N}$);
2. $O(f(n) + k) = O(f(n))$ ($k \in \mathbb{N}$);
3. თუ $O(f_2(n)) \subset O(f_1(n))$, მაშინ $O(f_1(n) + f_2(n)) = O(f_2(n))$.

დამტკიცება:

1. თუ ვაჩვენებთ, რომ $O(k \cdot f(n)) \subset O(f(n))$ და $O(k \cdot f(n)) \supset O(f(n))$, ტოლობა დამტკიცდება.
განვიხილოთ ნებისმიერი $g(n) \in O(k \cdot f(n))$. განმარტების თანახმად $\exists c \in \mathbb{N}$ ისეთი, რომ $g(n) < c \cdot k \cdot f(n)$ (რადგან k ნატურალურია). ეს კი განმარტების თანახმად იმას ნიშნავს, რომ $g(n) \in O(f(n))$: $\exists c \cdot k \in \mathbb{N}$ ისეთი, რომ $g(n) < c \cdot k \cdot f(n)$.
ახლა კი განვიხილოთ ნებისმიერი $g(n) \in O(f(n))$. განმარტების თანახმად $\exists d \in \mathbb{N}$ ისეთი, რომ $d \cdot f(n) > g(n)$. თუ უტოლობის ორივე მხარეს გავამრავლებთ k რიცხვზე, მივიღებთ: $k \cdot d \cdot f(n) > k \cdot g(n) > g(n)$ (რადგან $k \in \mathbb{N}$).

აქედან გამომდინარე, $\exists d \in \mathbb{N}$ ისეთი, რომ $d \cdot (k \cdot f(n)) > g(n)$. ესე იგი, $g(n) \in O(k \cdot f(n))$ ($O(k \cdot f(n))$ სიმრავლის განმარტების თანახმად).

Q.E.D.

სავარჯიშო 5.9: დაამტკიცეთ ზემოთ მოყვანილი ლემას მე-2-ე და მე-3-ე პუნქტები.

ანალოგიურად შეიძლება ნებისმიერი $f(n)$ ფუნქციის ზრდის რიგის ქვედა ზღვარი (ომეგა-დიდი $f(n)$) განვმარტოთ:

$$\Omega(f(n)) = \{g(n)|f(n) \in O(g(n))\}.$$

ეს ყველა იმ ფუნქციათა სიმრავლეა, რომელთა ასიმპტოტური ზრდის რიგი $f(n)$ ფუნქციის ასიმპტოტური ზრდის რიგზე ნაკლები არაა.

სავარჯიშო 5.10: დაამტკიცეთ, რომ $f_1(n) = 10n^2$ და $f_2(n) = 10^{-6}n^2$ ფუნქციათა ზრდის რიგის ქვედა ზღვარი ტოლია.

სავარჯიშო 5.11: ტოლია თუ არა შემდეგი ორი ფუნქციის ასიმპტოტური ზრდის რიგის ქვედა ზღვარი? (პასუხები დაამტკიცეთ):

1. $f_1(n) = n^2,\ f_2(n) = 15 \cdot n^2 \cdot \log\log n$; 2. $f_1(n) = \log n^3,\ f_2(n) = 1983 \cdot n$; 3. $f_1(n) = \log^2 n,\ f_2(n) = 10\log n$; 4. $f_1(n) = \log n^2,\ f_2(n) = 100\sqrt{\log n}$; 5. $f_1(n) = n,\ f_2(n) = \log\log^7 n$.

სავარჯიშო 5.12: დაამტკიცეთ, რომ $n \cdot \log n \in \Omega(100n)$, $n^2 \in \Omega(100n)$, $2^n \in \Omega(100n)$.

სავარჯიშო 5.13: მოიყვანეთ $\Omega(\log n)$ სიმრავლის 5 ელემენტი.

სავარჯიშო 5.14: დაამტკიცეთ, რომ თუ $f_1(n)$ ფუნქცია არ იზრდება უფრო სწრაფად, ვიდრე $f_2(n)$ ფუნქცია, მაშინ $\Omega(f_2(n)) \subset \Omega(f_1(n))$.

სავარჯიშო 5.15: მოიყვანეთ $f_1(n)$ და $f_2(n)$ ფუნქციების მაგალითები, რომელთათვისაც $\Omega(f_1(n)) \subset \Omega(f_2(n))$ და, ამავდროულად, $\Omega(f_1(n)) \neq \Omega(f_2(n))$.

5.2 ალგორითმების ბიჯების რაოდენობის შეფასება

განვიხილოთ შემდეგი ამოცანა:

მოცემულია: რიცხვების მიმდევრობა $a_1, a_2, ..., a_n \in \mathbb{N}$ და დამატებით ერთი რიცხვი $b \in \mathbb{N}$.

შედეგი: „კი" ან „არა"

შეზღუდვა: „კი" მაშინ და მხოლოდ მაშინ, თუ $\exists i \in \mathbb{N},\ 1 \leq i \leq n,\ \ a_i = b$.

სხვა სიტყვებით რომ ვთქვათ, ალგორითმი ადგენს, მოიძებნება თუ არა $a_1, ..., a_n$ მიმდევრობაში ერთი მაინც რიცხვი $a_i = b$.

ქვემოთ მოყვანილია რეკურსიული ალგორითმი, რომელიც ამ ამოცანას ხსნის:

ალგორითმი $K(a_1, a_2, ..., a_n, b)$

1. თუ მიმდევრობა $a_1, a_2, ..., a_n$ შემოსული არაა, დაბეჭდე „არა" და ალგორითმი დაასრულე;
2. თუ $a_1 = b$ დაბეჭდე „კი" და ალგორითმი დაასრულე;
3. ჩაატარე ალგორითმი $K(a_2, ..., a_n, b)$.

განვიხილოთ ამ ალგორითმის ბიჯები საწყის მონაცემებზე $a_1 = 3, a_2 = 7, a_3 = 0, a_4 = 8, b = 2$.

ალგორითმი $K(3,7,0,8,2)$ (აქ $a_1=3, a_2=7, a_3=0, a_4=8, b=2$).

1. თუ მიმდევრობა $a_1, a_2, ..., a_n$ შემოსული არაა, დაბეჭდე „არა” და ალგორითმი დაასრულე; (ეს არ სრულდება)
2. თუ $a_1=b$ დაბეჭდე „კი” და ალგორითმი დაასრულე; (ეს არ სრულდება)
3. $K(7,0,8,2)$ (აქ $a_1=7, a_2=0, a_3=8, b=2$).
4. თუ მიმდევრობა $a_1, a_2, ..., a_n$ შემოსული არაა, დაბეჭდე „არა” და ალგორითმი დაასრულე; (ეს არ სრულდება)
5. თუ $a_1=b$ დაბეჭდე „კი” და ალგორითმი დაასრულე; (ეს არ სრულდება)
6. $K(0,8,2)$ (აქ $a_1=0, a_2=8, b=2$).
7. თუ მიმდევრობა $a_1, a_2, ..., a_n$ შემოსული არაა, დაბეჭდე „არა” და ალგორითმი დაასრულე; (ეს არ სრულდება)
8. თუ $a_1=b$ დაბეჭდე „კი” და ალგორითმი დაასრულე; (ეს არ სრულდება)
9. $K(8,2)$ (აქ $a_1=8, b=2$).
10. თუ მიმდევრობა $a_1, a_2, ..., a_n$ შემოსული არაა, დაბეჭდე „არა” და ალგორითმი დაასრულე; (ეს არ სრულდება)
11. თუ $a_1=b$ დაბეჭდე „კი” და ალგორითმი დაასრულე; (ეს არ სრულდება)
12. $K(2)$ (აქ $a_1, a_2, ..., a_n$ მიმდევრობა ცარიელია).
13. თუ მიმდევრობა $a_1, a_2, ..., a_n$ შემოსული არაა, დაბეჭდე „არა” და ალგორითმი დაასრულე; (ეს სრულდება)

მაგრამ თუ ამოცანის საწყისი მონაცემებია $a_1=3, a_2=7, a_3=0, a_4=8, b=3$, მაშინ ალგორითმის მსვლელობა შემდეგნაირი იქნებოდა:

ალგორითმი $K(3,7,0,8,2)$ (აქ $a_1=3, a_2=7, a_3=0, a_4=8, b=3$).

1. თუ მიმდევრობა $a_1, a_2, ..., a_n$ შემოსული არაა, დაბეჭდე „არა” და ალგორითმი დაასრულე; (ეს არ სრულდება)
2. თუ $a_1=b$ დაბეჭდე „კი” და ალგორითმი დაასრულე; (ეს სრულდება)

ამ მაგალითიდან ჩანს, რომ ალგორითმების ბიჯების რაოდენობა დამოკიდებულია მის მონაცემთა რაოდენობასა და თვითონ მონაცემთა მნიშვნელობებზე.

განასხვავებენ ბიჯების შეფასების სამ შემთხვევას:

- უარესი შემთხვევის ანალიზს (worst-case), ანუ მაქსიმუმ რამდენი ბიჯი დაგვჭირდება ამ ამოცანის გადასაჭრელად, მაშინაც კი, როდესაც ყველაზე „ცუდი” მონაცემები შემოგვივა?
- საუკეთესო შემთხვევის ანალიზს (best-case), ანუ მინიმუმ რამდენი ბიჯი დაგვჭირდება ამ ამოცანის გადასაჭრელად, როდესაც ყველაზე „კარგი” მონაცემები შემოგვივა?
- საშუალო შემთხვევის ანალიზს (average-case), ანუ საშუალოდ რამდენი ბიჯი დაგვჭირდება ამ ამოცანის გადასაჭრელად?

ადვილი დასანახია, რომ ჩვენს ზედა ამოცანაში ალგორითმი $n+1$ მონაცემის დამუშავებას (n ელემენტიანი მასივში რაღაცა b რიცხვის პოვნას) მაქსიმუმ n ბიჯსა და მინიმუმ 1 ბიჯს მოანდომებს.

<u>სავარჯიშო 5.16:</u> დაამტკიცეთ ზემოთ მოყვანილი გამონათქვამი.

სხვა სიტყვებით რომ ვთქვათ, უარესი შემთხვევის ანალიზის შედეგად მიღებული ფუნქცია $f(n)$ გვეუბნება, რომ „n ცალი მონაცემისათვის მოცემული ალგორითმის ბიჯების რაოდენობა არასოდეს არ გადააჭარბებს $f(n)$ ფუნქციას”, ხოლო საუკეტესო შემთხვევის ანალიზის შედეგად მიღებული ფუნქცია კი გვეუბნება, რომ „მოცემული ალგორითმის ბიჯების რაოდენობა ვერასოდეს ვერ იქნება ამ ფუნქციაზე ნაკლები”.

რაც შეეხება გამოთვლის საშუალო დროის დადგენას, ეს პროცესი მათემატიკურ სტატისტიკას ემყარება და ამ კურსში მას არ განვიხილავთ.

ცხადია, რომ n მონაცემის დამუშავების მაქსიმალური და მინიმალური დრო მონაცემთა რაოდენობის ცვლილებასთან ერთად იცვლება, ანუ ეს არის ფუნქცია, რომელიც დამოკიდებულია $n \in \mathbb{N}$ ცვლადზე. ჩვენს ზედა მაგალითში ბიჯების მაქსიმალური რაოდენობაა $f_1(n) = n$, ხოლო მინიმალური კი $f_2(n) = 1$.

განვიხილოთ ალგორითმი, რომელიც n ცალი მონაცემის დამუშავებას მაქსიმუმ $f(n)$ ბიჯს ანდომებს, ხოლო 1 ბიჯის დამუშავებას კი 10^{-9} წამს ანდომებს.

ქვემოთ მოყვანილი ცხრილი, სადაც წარმოდგენილია $f(n)$ ფუნქციის რამოდეიმე მაგალითი. მასში ნაჩვენებია, მაქსიმუმ რამდენ ხანს მოანდომებს ეს ალგორითმი n მონაცემის დამუშავებას. ამ ცხრილში $1\mu s = 10^{-6}$ წმ, $1ms = 10^{-3}$ წმ ($1\mu s$ იკითხება: 1 მიკრო წამი, $1ms$ იკითხება: 1 მილი წამი.

n	$f(n) = \log n$	$f(n) = n$	$f(n) = n \cdot \log n$	$f(n) = n^2$	$f(n) = 2^n$	$n!$
10	0,003 μs	0,01 μs	0,033 μs	0,1 μs	1 μs	3,63 ms
20	0,004 μs	0,02 μs	0,086 μs	0,4 μs	1 ms	77,1 წელი
30	0,005 μs	0,03 μs	0,147 μs	0,9 μs	1 წმ	8,4×10^{15} წელი
40	0,005 μs	0,04 μs	0,213 μs	1,6 μs	18,3 წთ	
50	0,006 μs	0,05 μs	0,282 μs	2,5 μs	13 დღე	
100	0,007 μs	0,1 μs	0,644 μs	10 μs	4×10^{13} წელი	
1.000	0,010 μs	1 μs	9,966 μs	1 ms		
10.000	0,013 μs	10 μs	130 μs	100 ms		
100.000	0,017 μs	9,10 ms	1,67 ms	10 წმ		
1.000.000	0,020 μs	1 ms	19,93 ms	16.7 წთ		
10.000.000	0,023 μs	0,01 წმ	0,23 წმ	1,16 დღე		
100.000.000	0,027 μs	0,1 წმ	2,66 წმ	115,7 დღე		
1.000.000.000	0,03 μs	1 წმ	29,9 წმ	31,7 წელი		

ამ ცხრილიდან ჩანს, რომ თუ ალგორითმის ბიჯების რაოდენობაა $f(n) = n \cdot \log n$ ან უფრო ნელა ზრდადი ფუნქცია, მაშინ მისი გამოთვლები საკმაოდ სწრაფი იქნება. თუ $f(n) = n^2$, გამოთვლები სწრაფი იქნება დაახლოებით 50.000.000 ელემენტამდე. მაგრამ თუ $f(n) = 2^n$, ასეთი ალგორითმი პრაქტიკაში ვერ გამოიყენება 53-ზე მეტი მონაცემისათვის. თუ ალგორითმის ზედა ზღვარია $f(n) = n!$, მაშინ იგი პრაქტიკულად საერთოდ ვერ გამოიყენება.

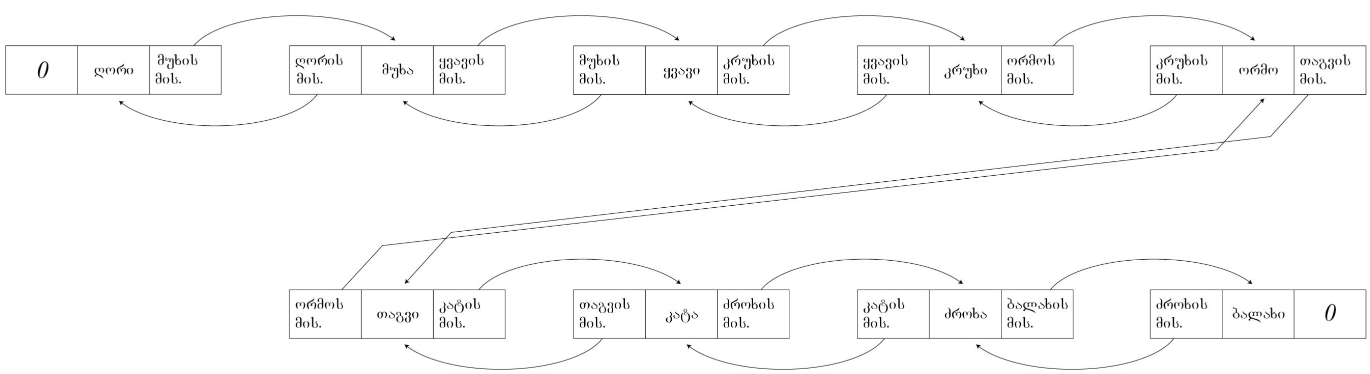
6 ბმული სიები

6.1 რწყილი და ჭიანჭველა

ცნობილ ქართულ ზღაპარში „რწყილი და ჭიანჭველა” რწყილი ჭიანჭველის გადასარჩენად ღორთან მიდის და თოკისთვის ჯაგარს თხოვს. ღორი მოსთხოვს რკოს და გააგზავნის მუხასთან. მუხა სთხოვს, რომ მოაშოროს ყვავი და გააგზავნის ყვავთან. ყვავი სთხოვს წიწილას და გააგზავნის კრუხთან. კრუხი სთხოვს ფეტვს და გააგზავნის ორმოსთან. ორმო სთხოვს თაგვის მოშორებას და გააგზავნის თაგვთან. თაგვი სთხოვს კატის მოშორებას და გააგზავნის კატასთან. კატა სთხოვს რძეს და გააგზავნის ძროხასთან. ძროხა სთხოვს ბალახს და გააგზავნის მინდორში, სადაც რწყილი მოკრეფს ბალახს, მიუტანს ძროხას, ის მისცემს რძეს და გააგზავნის უკან კატასთან. კატა რძეს რომ მიიღებს, მოეშვება თაგვს და ა.შ.: რწყილი განვლილ ჯაჭვს უკან გაჰყვება, ბოლოს მივა ღორთან, მიუტანს რკოს, მისგან მიიღებს ჯაგარს, დაწნის თოკს და თავის მეგობარს წყლიდან ამოიყვანს.

ამ ზღაპარში ჩვენ ერთი საინტერესო ფაქტი შეგვიძლია დავინახოთ, რომელიც ფართოდ გამოიყენება ინფორმატიკაში:

შექმნილია მონაცემთა ჯაჭვი, რომლის თავშიც დგას ღორი. ღორმა იცის, სად დგას მუხა, ანუ მონაცემთა ჯაჭვში შემდგომი ელემენტის მისამართი. მუხამ იცის ყვავის (ჯაჭვში მისი შემდგომი ელემენტის) მისამართი და ა.შ.: ჯაჭვის ყოველ ელემენტში ჩაწერილია სამი კომპონენტი: ინფორმაცია იმის შესახებ, თუ რა არის ეს კომპონენტი (ჩვენს მაგალითში რა ცხოველი ან მცენარეა), მისი წინა კომპონენტის მისამართი და მისი შემდგომი კომპონენტის მისამართი. გრაფიკულად ეს ნაჩვენებია ნახ. 31. მონაცემთა ასეთ ჯაჭვს ბმული სია ეწოდება. ბმული იმიტომ, რომ ამ ჯაჭვის ყოველი კომპონენტი მის წინა და მის შემდგომ კომპონენტზეა „გადაბმული”: მას თავის წინა და შემდგომი ელემენტის მისამართი აქვს დახსომებული.



ნახ. 31: ზღაპრის „რწყილი და ჭიანჭველა” ბმული სია

აღსანიშნავია, რომ რადგან ღორი ამ სიაში პირველია, მას წინა ელემენტი არ ჰყავს და, შესაბამისად, მისი მისამართიც ვერ ექნება. ამიტომაც წინა ელემენტის მისამართის მაგივრად უწერია 0. ანალოგიური სიტუაციაა ბალახთან: რადგან იგი ბოლო ელემენტია ჯაჭვში, მისი შემდგომი ელემენტის მისამართის ადგილას წერია 0.

აღსანიშნავია, რომ ეს 0 არაა რიცხვის მნიშვნელობის მატარებელი. ეს მხოლოდ იმის მანიშნებელია, რომ ამ ელემენტის შემდგომი (ან წინა) ელემენტი არ არსებობს (ამიტომაც ზოგგან წერენ კიდეც , ან ილ იმის აღსანიშნავად, რომ ცვლადი ცარიელია).

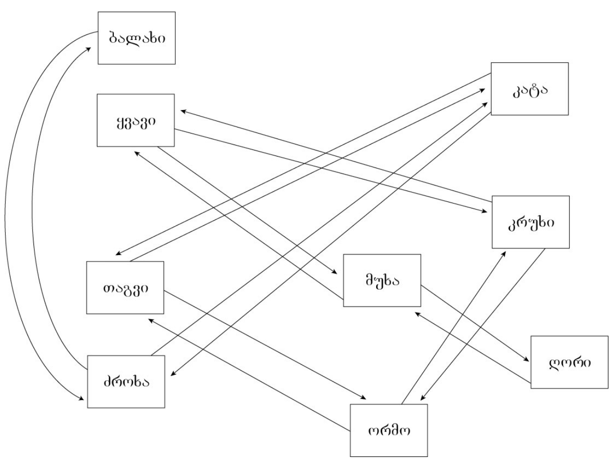
ბმულ სიის ერთ-ერთი უპირატესობაა ის, რომ მასში ელემენტის მოსაძებნად არაა საჭირო ყველა ელემენტის მისამართის ცოდნა, საკმარისია მხოლოდ მისი საწყისი ელემენტის მისამართი ვიცოდეთ: გადავალთ იმ ელემენტთან და თუ ეს არაა ის, რასაც ვეძებთ, გავიგებთ მისი შემდგომი ელემენტის მისამართს. შემდეგ გადავალთ ამ შემდგომ ელემენტზე (თუ ასეთი არსებობს) და ჩავატარებთ იგივე პროცედურას: ვნახავთ, არის თუ არა ეს ის ელემენტი, რომელსაც ვეძებთ. თუ არ არის, ვიგებთ მისი შემდგომი ელემენტის მისამართს (თუ არსებობს) და იგივეს ვიმეორებთ. თუ შემდგომი ელემენტი არ არსებობს, მაშინ საძებნი ელემენტი ამ სიაში არ ყოფილა.

ჩვენს კონკრეტულ მაგალითში, თუ გვაინტერესებს, არის თუ არა კრუხი ამ ბმულ სიაში, ვიწყებთ პირველი ელემენტით, რომლის მისამართი უნდა ვიცოდეთ (ამ შემთხვევაში ამბობენ, რომ საძიებელი ელემენტია „კრუხი”). ვადარებთ მნიშვნელობებს: კატას ვეძებთ, მაგრამ გვხვდება ღორი. ამიტომ ვამოწმებთ, არსებობს თუ არა შემდგომი ელემენტი (ანუ შესაბამის გრაფაში თუ წერია 0). რადგან ამ გრაფაში წერია მისამართი (და არა 0), გადავდივართ ამ მისამართზე. ვადარებთ აქტუალური ჩანაწერის მნიშვნელობას საძიებელი ელემენტის მნიშვნელობას.

აქტუალური ჩანაწერის მნიშვნელობაა „ყვავი". რადგან ჩვენ ვეძებთ კატას, უნდა გადავიდეთ შემდეგ ელემენტზე, მაგრამ ჯერ შევამოწმოთ, არსებობს თუ არა ასეთი ელემენტი. რადგან შესაბამის გრაფაში არ წერია 0, ამიტომ ასეთი ელემენტი არსებობს და გადავდივართ მის მისამართზე. ახლა აქტუალური ჩანაწერის მნიშვნელობაა „კატა". ვადარებთ საძებნ მნიშვნელობას. რადგან ემთხვევა, პროცედურა უნდა დასრულდეს: საძებნი ჩანაწერი ნაპოვნია.

თუ ჩვენს კონკრეტულ მაგალითში გვინდა გავიგოთ, არსებობს თუ არა ჩანაწერი „სპილო", ისევ ვიწყებთ თავიდან და თუ აქტუალური ჩანაწერის მნიშვნელობა არ ემთხვევა საძიებო ჩანაწერს, გადავდივართ შემდეგ ელემენტზე, თუ ასეთი არსებობს. როდესაც მივადგებით ჩანაწერს „ბალახი" და დავასკვნით, რომ იგი არ ემთხვევა საძიებო ჩანაწერის მნიშვნელობას, უნდა გადავიდეთ შემდეგზე, მაგრამ ჯერ შევამოწმოთ, არის თუ არა ეს აქტუალური ჩანაწერი ბოლო მოცემულ ბმულ სიაში. რადგან შემდეგი ჩანაწერის მისამართის შესაბამის გრაფაში წერია 0, შეგვიძლია დავასკვნათ, რომ ეს სიაში ბოლო ელემენტია და პროცედურა დავასრულოთ: ჩანაწერი „სპილო" ამ სიაში არ გვხვდება.

წინა ნახაზში ბმული სიის მეზობელი ელემენტები ერთი მეორეს მიყოლებით არიან წარმოდგენილნი. ეს არაა აუცილებელი: შესაძლოა, რომ ისინი არეულად იყვნენ განლაგებული (ნახ. 32). მთავარია, რომ ყოველმა ელემენტმა მისი წინა და მომდევნო ელემენტების მისამართები იცოდნენ.



ნახ. 32:

ბმული სიის ყოველი ელემენტი სამი კომპონენტისაგან შედგება: ესაა თვითონ ამ კომპონენტის მნიშვნელობა (ანუ გასაღები), მისი ცინა ელემენტის მისამართი და შემდგომი ელემენტის მისამართი. ნახ. 33-ში ზემოთ ნაჩვენებია ეს სამ კომპონენტიანი ელემენტი. თუ იგი განთავსებულია მისამართით x, მისი წინა ელემენტის მისამართი აღინიშნება ფუნქციით $L(x)$ (გრაფიკულად იწერება მარცხენა უჯრაში), ხოლო მისი მომდევნო ელემენტის მისამართი კი აღინიშნება ფუნქციით $R(x)$ (გრაფიკულად იწერება მარჯვენა უჯრაში). თვით მისი მნიშვნელობა აღინიშნება ფუნქციით $Key(x)$ (გრაფიკულად შუა უჯრაში). ამრიგად, $L($კრუხის მისამართი$) =$„ყვავი", $R($კატის მისამართი$) =$ „ძროხა", ხოლო $Key($მუხის მისამართი$) =$„მუხა".

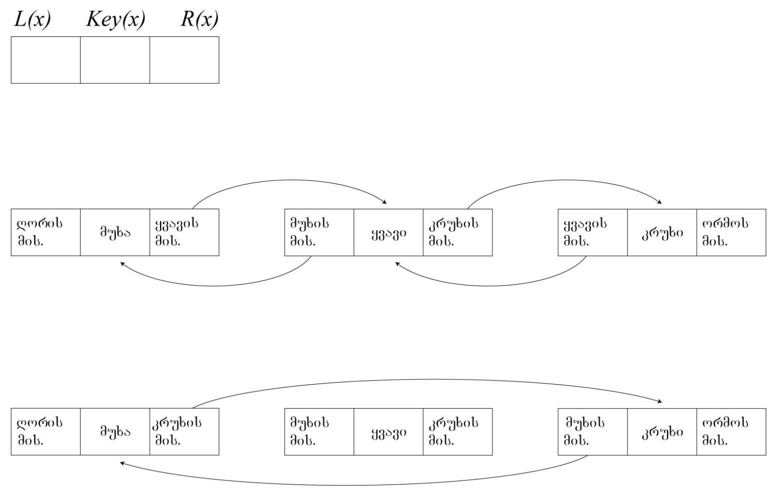
<u>სავარჯიშო 6.1:</u> რისი ტოლია $L(x)$, $R(x)$ და $Key(x)$, თუ $x =$„ძროხის მისამართი", $x =$„თაგვის მისამართი", $x =$„ბალახის მისამართი", $x =$„ორმოს მისამართი" ?

ზოგადად, თუ მოცემულია ბმული სიის რომელიმე ელემენტის მისამართი x, $Key(x)$ ამ ჩანაწერის მნიშვნელობაა, $L(x)$ მისი წინა ჩანაწერის მისამართი, ხოლო $R(x)$ კი - მისი მომდევნო ჩანაწერის მისამართი.

აქედან გამომდინარე, $R(L(x))$ არის x მისამართზე მყოფი ელემენტის წინა ჩანაწერის მარჯვენა გრაფას მნიშვნელობა

<u>სავარჯიშო 6.2:</u> რას ნიშნავს ჩანაწერები $R(R(x))$, $L(L(x))$, $L(R(x))$, $Key(L(x))$ და $Key(R(x))$?

როდესაც ვწერთ $L(x)$, $Key(x)$ ან $R(x)$, იმის და მიხედვით, თუ რისი ტოლია x, ეს ჩანაწერებიც სხვადასხვა იქნება. ამ შემთხვევაში იტყვიან, რომ x აქტუალურ ჩანაწერზე მიუთითებს. როდესაც x ბმული სიის რომელიმე ჩანაწერის მისამართია (ანუ ამ ჩანაწერზე მიუთითებს), მის შემდგომ ელემენტზე „გადასვლა" (ანუ მის შემდგომ



ნახ. 33:

ელემენტზე მითითება) შეიძლება ბრძანებით $x = R(x)$. აქ x ცვლადს მიენიჭება აქტუალური ჩანაწერის შემდეგი ელემენტის მისამართი და იგი ამ შემდეგ ელემენტზე მიუთითებს (შემდეგ ელემენტზე „გადავა”).

სავარჯიშო 6.3: რა ბრძანებით უნდა „გადავიდეთ” აქტუალური ჩანაწერის წინა ელემენტზე?

ცხადია, სანამ გადავალთ წინა ან მომდევნო ელემენტზე, უნდა შევამოწმოთ, არსებობს თუ არა ეს ელემენტი (ანუ აქტუალური ჩანაწერი ბოლო ან პირველი ხომ არაა).

სავარჯიშო 6.4: რა ბრძანებით შეიძლება შემოწმდეს, არის თუ არა x მისამართზე მყოფი ჩანაწერი ბმული სიის ბოლო ან პირველი ელემენტი?

ახლა წარმოვიდგინოთ, რომ გვინდა ზღაპრის გადაკეთება ისე, რომ ამ ჯაჭვიდან ამოვაგდოთ ყვავი: მუხა პირდაპირ აგზავნის კრუხთან (ანუ ჩანაწერი „ყვავი” ამ ბმული სიიდან უნდა ამოვარდეს). ესე იგი, თავიდან მოცემული გვაქვს სიტუაცია, რომელიც ნაჩვენებია ნახ. 33-ში შუაში და გვინდა მივიღოთ სიტუაცია, რომელიც ნაჩვენებია იგივე ნახაზში ქვემოთ. ჩანახერი „ყვავი„ ბმული სიიდან ამოვარდება იმ თვალსაზრისით, რომ ამ სიაში მოძრაობისას ამ ელემენტს ვეღარ წავაწყდებით. ჩანაწერი „ყვავი” სადღაც კი იარსებებს, მაგრამ იგი ამ სიის ელემენტი აღარ იქნება.

როგორც ზედა ნახაზიდან ჩანს, ყვავის წინა ელემენტი (ამ შემთხვევაში „მუხა”) უნდა მიუთითებდეს „ყვავის” შემდგომ ელემენტზე, ამ შემთხვევაში ჩანაწერზე „კრუხი” და პირიქით: ყვავის შემდგომი ელემენტი (ამ შემთხვევაში „კრუხი”) უნდა მიუთითებდეს „ყვავის” წინა ელემენტზე, ამ შემთხვევაში ჩანაწერზე „მუხა”. აქედან გამომდინარე, თუ გვინდა რაიმე ელემენტის ბმული სიიდან ამოშლა, მისი წინა ელემენტის მარჯვენა გრაფაში უნდა ჩაიწეროს ამ ელემენტის შემდგომი ჩანაწერის მისამართი, ხოლო ამ ელემენტის შემდგომი ელემენტის მარცხენა გრაფაში უნდა ჩაიწეროს ამ ელემენტის წინა ჩანაწერის მისამართი.

ბრძანებებით ეს ასე ჩაიწერება:

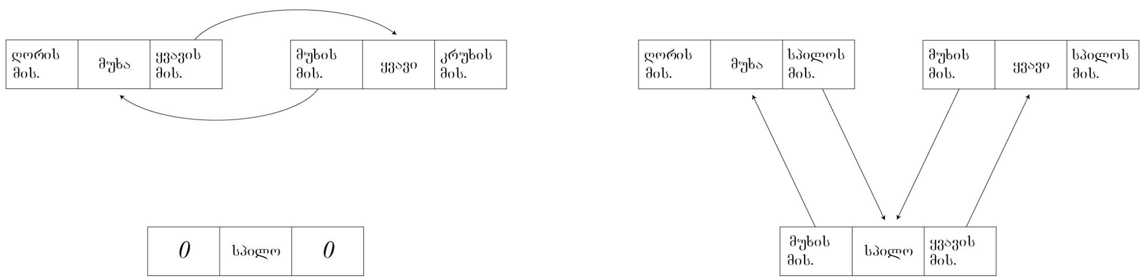
მოც.: x არის იმ ჩანაწერის მისამართი, რომელიც უნდა ამოვშალოთ.

- $R(L(x)) = R(x)$ (აქტუალური ელემენტის წინა ელემენტი უნდა მიუთითებდეს აქტუალური ელემენტის მომდევნო ელემენტზე);
- $L(R(x)) = L(x)$ (აქტუალური ელემენტის მომდევნო ელემენტი უნდა მიუთითებდეს აქტუალური ელემენტის წინა ელემენტზე);
- $x = L(x)$ აქტუალური ელემენტი სიიდან ამოგდებულია. ამიტომ გადავდივართ მის წინა ელემენტზე.

სავარჯიშო 6.5: დავუშვათ, x არის ბმული სიის პირველი ელემენტის მისამართი. ბრძანებებით ჩაწერეთ, როგორ შეიძლება ამ ელემენტის ბმული სიიდან წაშლა.

სავარჯიშო 6.6: დავუშვათ, x არის ბმული სიის ბოლო ელემენტის მისამართი. ბრძანებებით ჩაწერეთ, როგორ შეიძლება ამ ელემენტის ბმული სიიდან წაშლა.

ახლა კი წარმოვიდგინოთ, რომ ჩვენი მგალითის ბმულ სიაში ჩანაწერ „მუხასა" და „ყვავს" შორის უნდა ჩავსვათ ჩანაწერი „სპილო" (ნახ. 34 მარცხნივ). საწყის ჯაჭვში ჩანაწერი „მუხა" შემდგომ ელემენტად მიუთითებს ჩანაწერზე „ყვავი", ხოლო ჩანაწერი „ყვავი" წინა ელემენტად მიუთითებს ჩანაწერზე „მუხა". იმისათვის, რომ ჩავსვათ ჩანაწერი „სპილო", უნდა შევქმნათ ისეთი ბმულები, როგორიც ნაჩვენებია ნახ. 34 მარჯვნივ.



ნახ. 34:

ბრძანებებით ეს შემდეგნაირად ჩაიწერება:

მოც.: x არის იმ ჩანაწერის მისამართი, რომლის შემდეგაც უნდა ჩაჯდეს ახალი ჩანაწერი; S არის ახალი ჩანაწერის მისამართი.

- $R(S) = R(x)$ (ახალი ელემენტი უნდა მიუთითებდეს აქტუალური ელემენტის მომდევნო ელემენტზე);
- $L(S) = x$ (ახალი ელემენტი უნდა მიუთითებდეს აქტუალურ ელემენტზე);
- $L(R(x)) = S$ (აქტუალური ელემენტის მომდევნო ელემენტი უნდა იყოს ახალი ელემენტი);
- $R(x) = S$ (აქტუალური ელემენტის მომდევნო ელემენტი უნდა იყოს ახალი ელემენტი).

სავარჯიშო 6.7: მოცემულია ბმული სია და S მისამართზე განთავსებული ახალი ჩანაწერი. დაწერეთ ბრძანებათა მიმდევრობა, რომელთა საშუალებითაც შეიძლება ახალი ელემენტის სიის პირველ ელემენტად ჩამატება.

სავარჯიშო 6.8: მოცემულია ბმული სია და S მისამართზე განთავსებული ახალი ჩანაწერი. დაწერეთ ბრძანებათა მიმდევრობა, რომელთა საშუალებითაც შეიძლება ახალი ელემენტის სიის ბოლო ელემენტად ჩამატება.

იმისათვის, რომ ვიპოვნოთ ბმული სიის ბოლო ჩანაწერი, უნდა „ვიაროთ მარჯვნივ" მანამ, სანამ არ შეგვხვდება ბოლო ელემენტი. აქედან გამომდინარე, ეს შეიძლება მოხერხდეს შემდეგი ბრძანებების საშუალებით:

მოცემულია: ბმული სია და მისი ერთ-ერთი ელემენტის მისამართი x.

- while($R(x)$!= 0)

  $x = R(x)$

სავარჯიშო 6.9: დაამტკიცეთ, რომ ამ ციკლის დამთავრების შემდეგ x ცვლადში სიის ბოლო ელემენტის მისამართი ეწერება.

სავარჯიშო 6.10: დაწერეთ ბრძანებათა მიმდევრობა, რომლითაც ბმული სიის პირველი ელემენტის პოვნა შეიძლება.

სავარჯიშო 6.11: მოცემულია ბმული სია და მისი ერთ-ერთი ელემენტის მისამართი x. აგრეთვე მოცემულია რაღაცა მნიშვნელობა M. დაწერეთ ბრძანებათა მიმდევრობა, რომელთა მეშვეობითაც შეიძლება იმის დადგენა,

გვხვდება თუ არა ბმულ სიაში ელემენტის ჩანაწერი, რომლის მნიშვნელობაცაა M (ანუ, სხვა სიტყვებით რომ ვთქვათ, თუ x რაიმე ჩანაწერის მისამართია, $Key(x) = M$).

სავარჯიშო 6.12: მოცემულია n ელემენტიანი ბმული სია. გამოიანგარიშეთ ამ სიაში ელემენტის ჩამატებასა და წაშლისათვის საჭირო ოპერაციათა რაოდენობა და შეაფასეთ მისი ზედა ზღვარი O აღნიშვნით.

სავარჯიშო 6.13: მოცემულია ჩვეულებრივი მასივი, რომელიც შედგება n ელემენტისაგან. დაწერეთ ამ მასივში ელემენტის ჩამატების ალგორითმი. გამოიანგარიშეთ მისი ბიჯების რაოდენობა და შეაფასეთ მისი ზედა ძღვარი O აღნიშვნით.

სავარჯიშო 6.14: რა უპირატესობა აქვს ბმულ სიას მასივთან შედარებით?