2 ამოცანათა რეკურსიული აღწერა

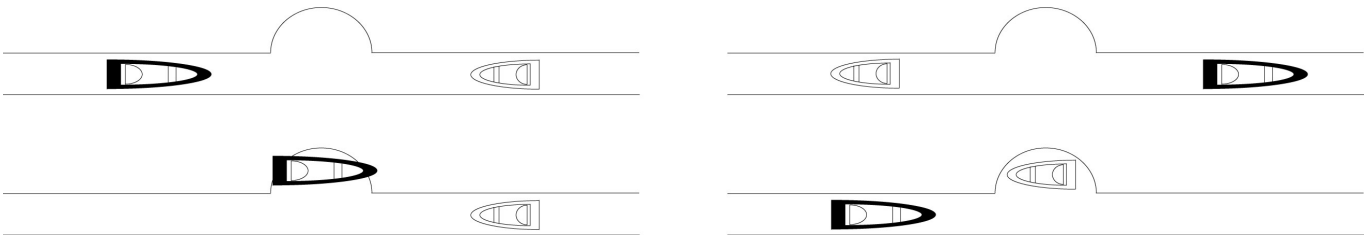
2.1 ამოცანა ნავების შესახებ

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 თეტლმა ნავმა შავს გვერდი აუქციოს, საჭიროა შემდეგი ალგორითმის ჩატარება:

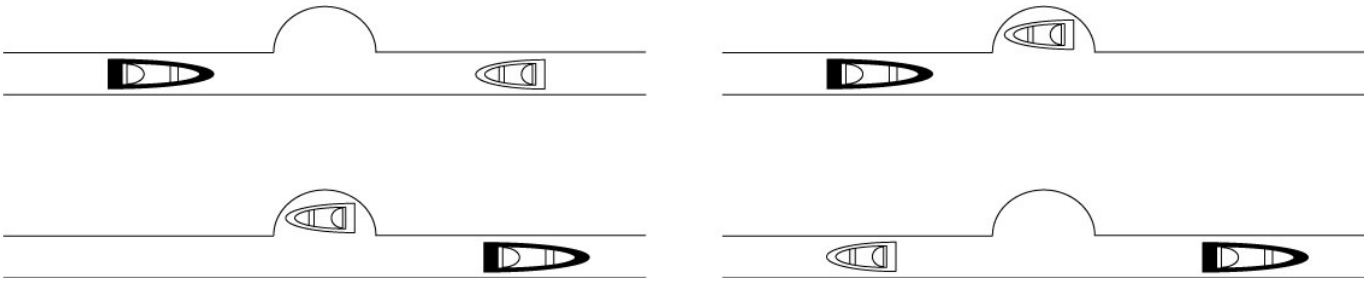
ალგორითმი „ერთი ნავის გაყვანა”

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.

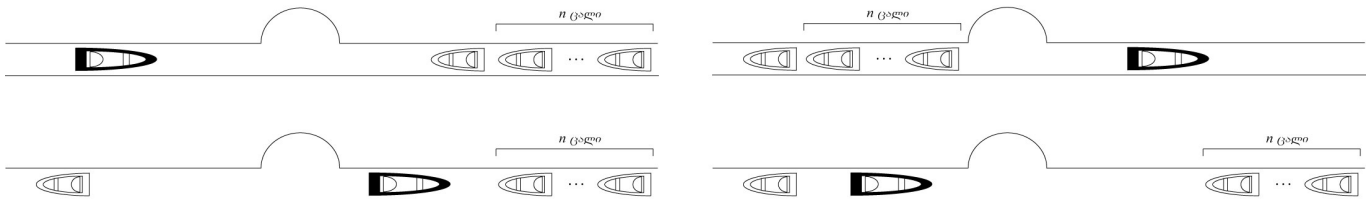
ახლა კი დავუშვათ, რომ ალგორითმი A_n n თეთრი ნავის გვერდის აქცევას ახერხებს (ნახ. 10). აქამდე ჩვენ განვიხილეთ, თუ როგორია A_n, თუ $n = 1$, ან $n = 2$.

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).

სავარჯიშო 2.1: 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).

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

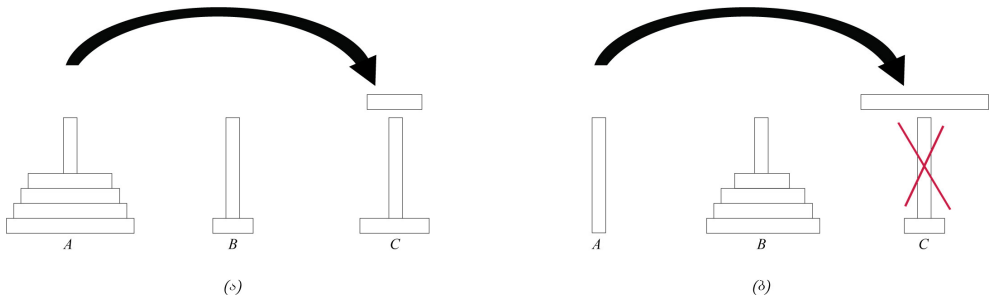
შედეგი: A ძელზე აგებული პირამიდა C ძელზე (ნახ. 12 (ბ)).

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 $A_1^{A,C}$.

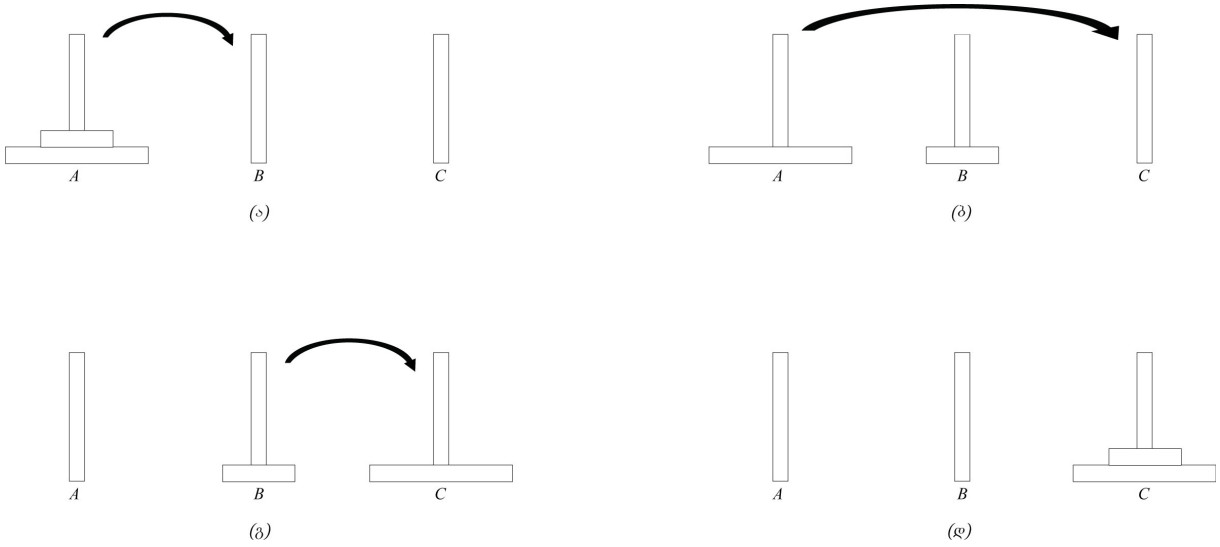
იმისათვის, რომ ორ რგოლიანი პირამიდა A ძელიდან C ძელზე გადავიტანოთ, საჭიროა შემდეგი მოქმედებების ჩატარება:

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



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

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

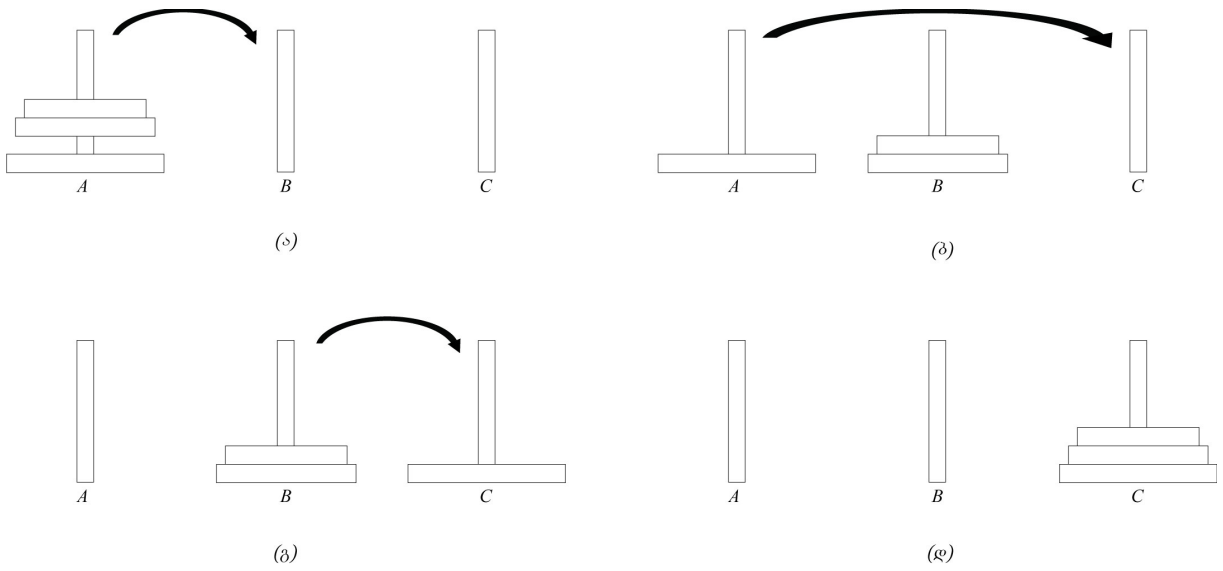
ორ რგოლიანი პირამიდის A ძელიდან C ძელზე გადატანის ალგორითმი (ანუ ზემოთ მოყვანილი სამ ბიჯიანი პროცესი) აღვნიშნოთ როგორც $A_2^{A,C}$.

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 2.2: 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 2.3: 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 $A_3^{A,C}$).

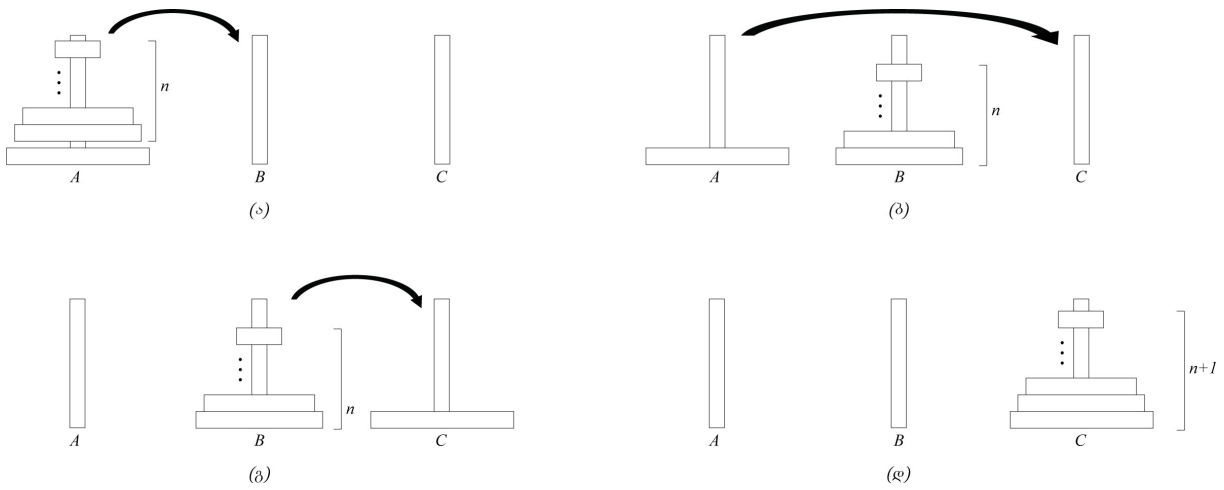
თუ ვიცით, როგორია n რგოლიანი პირამიდის ერთი ძელიდან მეორეზე გადატანის ალგორითმი $A_n^{X_1,X_2}$, ადვილად შევადგენთ $n+1$ რგოლიანი ალგორითმის გადატანის ალგორითმს $A_{n+1}^{X_1,X_2}$ (შესაბამისი მოქმედებები ნაჩვენებია ნახ. 26 -ში):

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

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

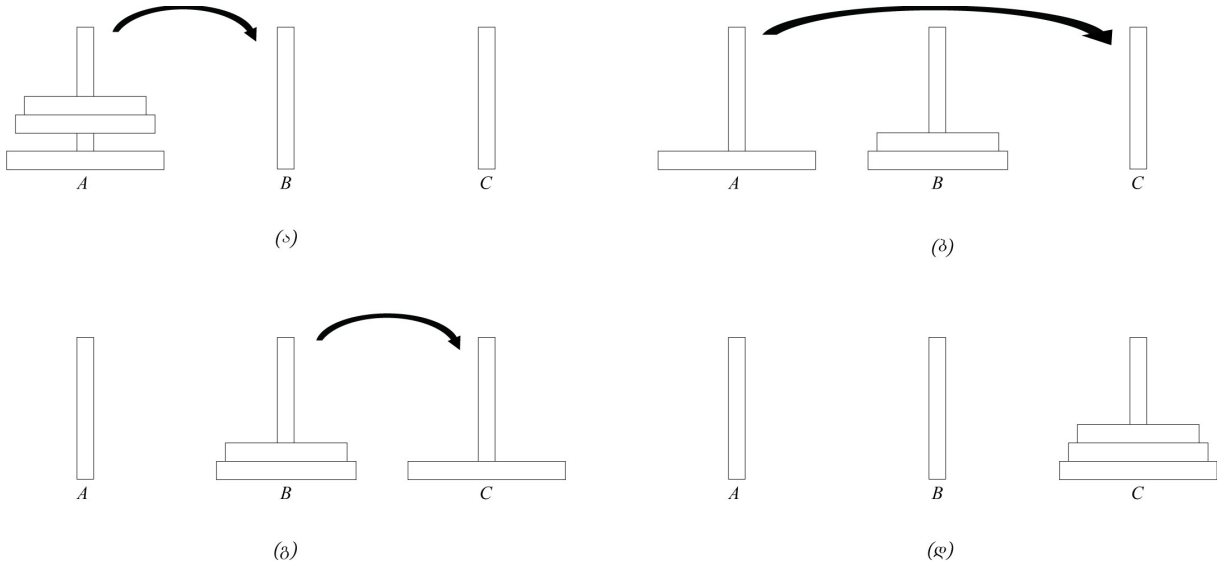
სავარჯიშო 2.4: რას აღნიშნავს შემდეგი ჩანაწერები: $A_7^{B,C}$, $A_{12}^{C,B}$, $A_4^{B,C}$?

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

<u>სავარჯიშო 2.5</u>: 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?