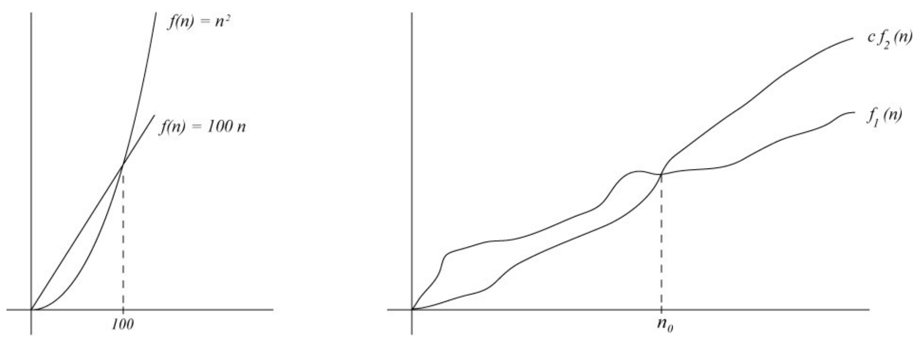
6 ალგორითმების სისწრაფის შეფასება

6.1 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. 30 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 $\log n = \log_2 n, \ln n = \log_e n$ და $\lg n = \log_{10} n$).

შენიშვნა: აქ და შემდგომში განხილული ფუნქციები დადებითია.



ნახ. 30: ორი ფუნქციის გრაფიკი

სავარჯიშო 6.1: $f_1(n)$ 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 6.2: 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 6.3: დაამტკიცეთ, რომ $f_1(n) = 10n^2$ და $f_2(n) = 10^{-6} \cdot n^2$ ფუნქციათა ასიმპტოტური ზრდის რიგი ტოლია.

სავარჯიშო 6.4: 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. $f_1(n) = n,\ f_2(n) = \log \log^7 n$.

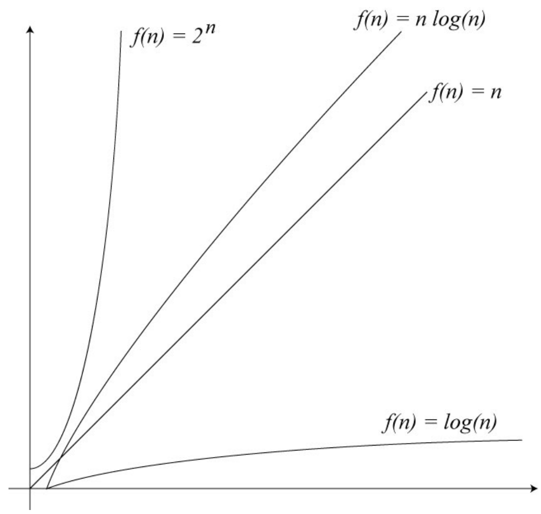
ნახ. 31 გვიჩვენებს რამოდენიმე ფუნქციის ზრდის სისწრაფეს, საიდანაც შეიძლება მათი ასიმპტოტური ზრდის რიგის დანახვა. ყველაზე ნელა იზრდება ლოგარითმული ფუნქცია $f(n) = \log n$; შემდეგია წრფივი ფუნქცია $f(n) = n$. მასზე სწრაფად იზრდება ფუნქცია $n \cdot \log n$ და ყველაზე დიდი ზრდის რიგი აქვს $f(n) = 2^n$ ფუნქციას.

სავარჯიშო 6.5: $f_1(n)$ 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. 31: 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 6.6: დაამტკიცეთ, რომ $100n \in O(n \cdot \log n)$, $700n \in O(n^2)$, $200n^2 \in O(2^n)$.

სავარჯიშო 6.7: მოიყვანეთ $O(n \log n)$ სიმრავლის 5 ელემენტი.

ლემა 6.1: 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 6.8: 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 6.2: $O(f(n))$ 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 $O(f_1(n)) \subset O(f_2(n))$, 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 6.9: 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 6.10: დაამტკიცეთ, რომ $f_1(n) = 10n^2$ და $f_2(n) = 10^{-6}n^2$ ფუნქციათა ზრდის რიგის ქვედა ზღვარი ტოლია.

სავარჯიშო 6.11: 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 6.12: დაამტკიცეთ, რომ $n \cdot \log n \in \Omega(100n)$, $n^2 \in \Omega(100n)$, $2^n \in \Omega(100n)$.

სავარჯიშო 6.13: მოიყვანეთ $\Omega(\log n)$ სიმრავლის 5 ელემენტი.

სავარჯიშო 6.14: 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 6.15: მოიყვანეთ $f_1(n)$ და $f_2(n)$ ფუნქციების მაგალითები, რომელთათვისაც $\Omega(f_1(n)) \subset \Omega(f_2(n))$ და, ამავდროულად, $\Omega(f_1(n)) \neq \Omega(f_2(n))$.

6.2 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 $\exists i \in \mathbb{N},\ 1 \le i \le n,\ \ a_i = b$.

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 $K(3,7,0,8,3)$ (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 6.16:</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 საუკეთესო 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$\times 10^{15}$ 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$\times 10^{13}$ 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.