DOMAĆI ZADACI

https://petlja.org/biblioteka/r/Problems/maxStack

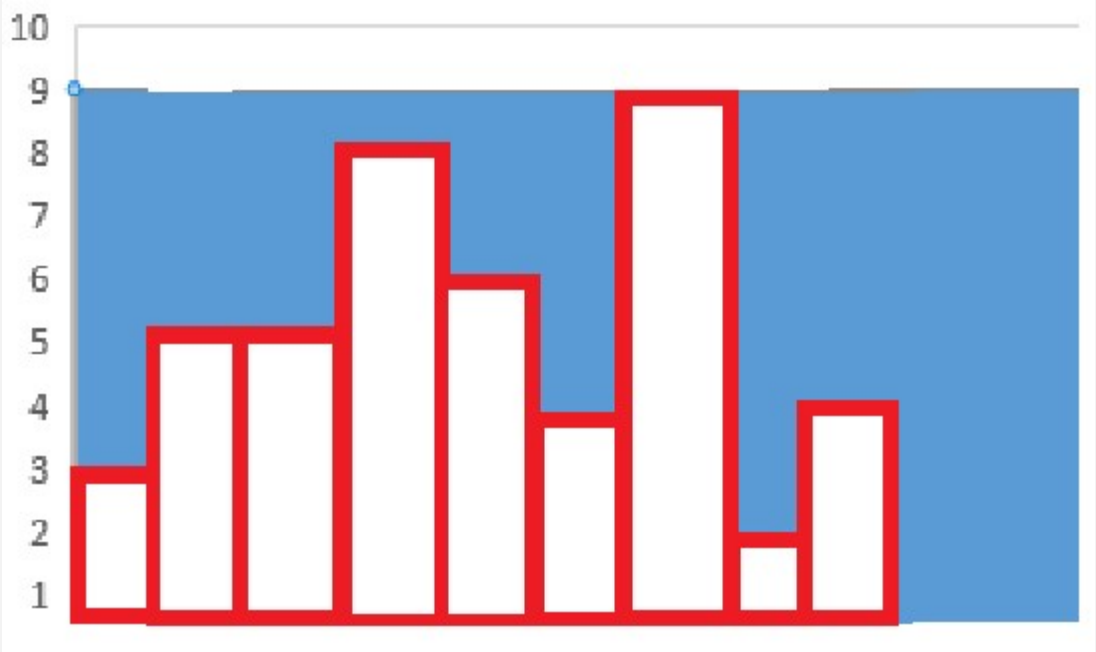
https://petlja.org/biblioteka/r/Problems/BrzaHrana

Najveći pravougaonik u histogramu

Problem: Niz brojeva predstavlja visine stubića u histogramu (svaki stubić je jedinične sirine). Odredi površinu najvećeg pravougaonika u tom histogramu.

ULAZ
9
3 5 5 8 6 4 9 2 4

IZLAZ
24



Za svaki stubić u histogramu potrebno je da odredimo
1. poziciju prvog stubića levo od njega koji je strogo manji od njega (ili poziciju -1 neposredno ispred početka, ako takav stubić ne postoji) i

2. poziciju prvog stubića desno od njega koji je strogo manji od njega (ili poziciju n, neposredno iza kraja niza, ako takav stubić ne postoji).

Ako te pozicije obeležimo sa l i d, onda možemo zaključiti da je najveća površina pravougaonika koji sadrži tekući stubić h[i]$(d-l-1)\cdot h[i]$
(taj pravougaonik je visine $h[i]$, a pod pretpostavkom da je svaki stubić jedinične širine, ukupna širina mu je $d-l-1$).

Pošto su svi stubići od pozicije $l+1$ do $d-1$ visine veće ili jednake od h[i], takav pravougaonik je moguće upisati histogram.
Jasno je i da ne može da postoji pravougaonik koji bi bio viši od ovoga (jer je i-ti stubić visine $h[i]$ i viši pravougaonik bi njega prevazišao) niti da postoji pravougaonik koji bi
bio širi od ovoga
(jer čak i da postoje stubići na pozicijama l i d, oni su niži od h[i] i pravougaonik visine h[i] ne bi mogao da se proširi i upiše u njih).

Naivan način da se za svako i odrede l i d je da za svaki stubić i iznova puštamo petlje koje nalaze odgovarajuće ivične stubiće.

```
#include <iostream>
#include <vector>
using namespace std;

int main() {

  int n;
  cin >> n;

  vector<int> h(n);

  for (int i = 0; i < n; i++)
    cin >> h[i];

  int max = 0;
  for (int i = 0; i < n; i++) {
    int l = i;
    while (l >= 0 && h[l] >= h[i])
      l--;

    int d = i;
    while (d < n && h[d] >= h[i])
      d++;

    int P = (d - l - 1) * h[i];
```

```
    if (P > max)
      max = P;
  }

  cout << max << endl;
  return 0;

}
```

Ovaj algoritam je prilično neefikasan (složenost najgoreg slučaja mu je očigledno $O(n^2)$ i ona nastupa, na primer, kada su svi stubići jednake visine).

Već smo razmatrali probleme određivanja najbližeg manjeg prethodnika i najbližeg manjeg sledbenika za svaki element niza i videli smo da oba problema možemo rešiti u vremenu $O(n)$.
Jedno moguće rešenje je da u jednom prolazu odredimo pozicije najbližih manjih prethodnika svakog elementa, u drugom pozicije najbližih većih sledbenika za svaki element niza i u trećem da na osnovu tih pozicija izračunamo maksimalne površine pravougaonika za svaki stubić.
Međutim, pokazaćemo da se zahvaljujući sličnosti algoritama za određivanje najbližeg manjeg prethodnika i najbližeg manjeg sledbenika sve može uraditi samo u jednom prolazu.

Na stek ćemo postavljati one stubiće za koje još ne znamo poziciju najbližeg manjeg sledbenika (i za koje još nismo odredili površinu maksimalnog pravougaonika koji određuju).
Stubiće obilazimo s leva na desno i za svaki stubić h_d na koji naiđemo sa vrha steka redom skidamo i obrađujemo jedan po jedan stubić h koji je strogo veći od
njega. Svakom od tih stubića h stubić h_d je najbliži manji sledbenik.

Ako ispod h na steku postoji neki stubić on je najbliži manji ili jednak prethodnik stubiću h.
Da bismo našli najbližeg strogo manjeg prethodnika potrebno je da nastavljamo da sa steak skidamo sve stubiće čija je visina jednaka h.
Stubić h_l koji se nalazi na vrhu steka nakon toga je najbliži strogo manji prethodnik stubiću h. Ako takvog stubića nema, stubić h nema manjih prethodnika i histogram koji mu odgovara se može širiti skroz do levog kraja histograma (tada je $l=-1$).

Površina maksimalnog pravougaonika koji uključuje i stubić h je $h\cdot(d-l-1)$.

Primetimo da se za stubiće jedanke visine stubiću h koji su skinuti sa steka ne računa površina pravougaonika - za tim nema potrebe jer je ona jednaka površini pravougaonika određenog stubićem h.

Recimo i da smo umesto potrage za najmanjim strogo manjim prethodnikom mogli da površine računamo na osnovu najbližih manje-jednakih prethodnika. U tom slučaju ne bismo imali petlju u kojoj se sa steka skidaju svi stubići čija je visina jednaka *h*. Tada bi se za svaki stubić računala površina, ali ne bi površine pravougaonika svih bile maksimalne. Algoritam bi i u tom slučaju bio korektan, jer bi se maksimalna površina pravougaonika određenog susednim stubićima iste visine korektno izračunala prilikom skidanja poslednjeg od njih sa steka. Kada se stubići sa vrha steak obrade (kada se stek isprazni ili kada se na njegovom vrhu nađe stubić h_l koji je manji ili jednak od h_d), na vrh steka se stavlja h_d.

Naglasimo i da je po dolasku do kraja niza, potrebno još obraditi sve stubiće koji su ostali na vrhu steka (to će biti stubići za koje ne postoji strogo manji sledbenik). Njihova obrada teče po istom prinicipu kao i ranije, osim što se za vrednost pozicije *d* uzima *n*. Zato je u implementaciji poželjno objediniti tu završnu obradu sa prvom fazom algoritma.

```
#include <iostream>
#include <vector>
#include <stack>

using namespace std;


int main() {
  int n;
  cin >> n;
  vector<int> a(n);
  for (int i = 0; i < n; i++)
    cin >> a[i];


  int max = 0;
  stack<int> s;


  for (int d = 0; d < n || !s.empty(); d++) {
    while (!s.empty() && (d == n || a[s.top()] > a[d])) {
      int h = a[s.top()];
```

```
      s.pop();
      while (!s.empty() && a[s.top()] == h)
         s.pop();
      int l = s.empty() ? -1 : s.top();
      int P = (d - l - 1) * h;
      if (P > max)
    max = P;
    }
    if (d < n)
      s.push(d);
  }
  cout << max << endl;
  return 0;
}
```

Prikažimo rad algoritma na jednom primeru.

3 5 5 8 6 4 9 2 4

Stek: 0 3

Stek: 0 1 3 5

Stek: 0 1 2 3 5 5

Stek: 0 1 2 3 5 5 8

6 na poziciji 4 je najbliži strogo manji sledbenik za 8

Stek: 0 1 2 3 5 5

5 na poziciji 2 je najbliži strogo manji prethodnik za 8

P = 8

Stek: 0 1 2 4 3 5 5 6

4 na poziciji 5 je najbliži strogo manji sledbenik za 6

Stek: 0 1 2 3 5 5

5 na poziciji 2 je najbliži strogo manji prethodnik za 6

P = 12

4 na poziciji 5 je najbliži strogo manji sledbenik za 5

Stek: 0 1 3 5

Stek: 0 3

3 na poziciji 0 je najbliži strogo manji pretodnik za 5

P = 20

Stek: 0 5 3 4

Stek: 0 5 6 3 4 9

2 na poziciji 7 je najbliži strogo manji sledbenik za 9

Stek: 0 5 3 4

4 na poziciji 5 je najbliži strogo manji prethodnik za 9

P = 9

2 na poziciji 7 je najbliži strogo manji prethodnik za 4

Stek: 0 3

3 na poziciji 0 je najbliži strogo manji prethodnik za 4

P = 24

2 na poziciji 7 je najbliži strogo manji sledbenik za 3

Stek:

3 nema strogo manjeg prethodnika

P = 21

Stek: 7 2

Stek: 7 8 2 4

4 nema najbližeg strogo manjeg sledbenika

Stek: 7 2

2 na poziciji 7 je najbliži strogo manji prethodnik za 4

P = 4

2 nema najbližeg strogo manjeg sledbenika

Stek:

2 nema najbližeg strogo manjeg prethodnika

P = 18

maxP = 24