**Binarno traženje elementa u skupu**

Da bi se pronašao neki elemenat u skupu od N elemenata Prvo što pada na pamet je pročayak kroz skup I uporedjivanje svakog lementa s traženim. Algoritam je složenosti O(N).

**cin >> x;** // x je elemenat kog trazimo u nizu N

**int nadjen = 0;**

**for (int i = 1; i<=N; i++)**

**if (niz[i]==x) {nadjen=i; break;}**

// promenljiva nadjen nam govori na kom mestu u nizu je

// trazeni elemenat ili 0 ako ga nema.

Ali to nije najbolje, tehnikom podeli i pokori, moguće je izvesti traženje u vremenu O(log2 N):

Uslov da je sledeći algoritam moguće izvesti je da skup koji posmatramo bude sortiran.

**Int BT(int x) {** // x je elemenat kog trazimo u nizu N

**int left=1;** // ili mesto prvog clan intervala u kom trazimo

**int right=N;** // ili mesto poslednj. clana interv. u kom trazimo

**while (left<=right) {**

**int srednji=(left+right)/2;**

**if (niz[srednji]==x) return srednji;** // nadjen je x

**else if (niz[srednji]<x) left = srednji + 1;** // idemo nadesno

**else right = srednji - 1;** // idemo nalevo

**}**

**return 0; // nije nadjen**

**}**

Ako uporedimo vremenski ova dva algoritma, na 1 000 000 članova, prvi je oko 25000 puta sporiji.

o

**Binarno traženje rešenja u skupu svih mogućih**

Ako rešenje linearno zavisi od ulaznih podataka, tada korišćenjem binarnog traženja pos vim mogućim rešenjima, možemo naći najpovoljnije. Algoritam je složenosti O(log2 N) .

**int left=??;** // najmanje moguće pretpostavljeno rešenje

**int right=??;** // najveće moguće pretpostavljeno rešenje

**while (left<=right) {**

**int srednji=(left+right)/2;**

**if (valja(srednji)) {** // funkcija valja() proverava da li je za

**resenje=srednji;** // ulaz srednji rešenje valjano

**left = srednji + 1;** // idemo nadesno, trazimo najveće resenje

**}**

**else right = srednji - 1;** // idemo nalevo

**}**

Zadaci:

* [Petlja](https://petlja.org/biblioteka/r/Kamenje)
* <https://www.codechef.com/OCT18B/problems/HMAPPY>
* <https://petlja.org/biblioteka/r/Problems/2009-okruzno-ss-burici>
* <https://petlja.org/biblioteka/r/Problems/FastPizza>