Broj podnizova parne sume

Počećemo sa jednim naizgled jednostavnim primerom. Videćemo da se neka očigledna rešenja mogu dosta ubrzati.

Problem Dat je niz a prirodnih brojeva dužine n. Koliko ima podnizova, sastavljenih od uzastopnih elemenata, čija je suma paran broj?

Drugim rečima, treba naći broj uređenih parova (i, j) , gde je $1 \leq i \leq \mathrm{j} \leq \mathrm{n}$, za koje je

$$a_i + a_{i+1} + \cdots + a_j \text{ paran broj}$$

Primera radi, niz $a = (1,2,3,4)$ ima četiri uzastopna podniza parne sume - $(1,2,3)$, $(1,2,3,4)$, (2) i (4).

Nakon čitanja problema, većina odmah nastupi sa očiglednom idejom: za svaku vrednost promenjivih i i j, sumiraću elemente sa indeksima između njih i proveriti da li je to paran broj. Posmatrajmo kod za ovu ideju prikazan u *Algoritmu 3*.

```
=============================================================
01 brojParnih = 0;
02 for i = 1 to n
03    for j = i to n
04       suma = 0;
05       for k = i to j do
06          suma = suma + a [k];
07       if (suma mod 2 == 0) then
08          brojParnih = brojParnih + 1;
=============================================================
```

Algoritam 3. Prvo rešenje problema parnih podnizova

Složenost *Algoritma 3* je $O(n^3)$, međutim pokušajmo ovu implementaciju da ubrzamo. Linije koda 04 – 06 sumiraju vrednosti elemenata od i do j. Kada fiksiramo levu granicu tog segmenta po kome sumiramo, promenjivu i, posmatrajmo kako se menja suma kada se promenjiva j povećava za 1. Primećujemo da se suma poveća za vrednost $a\,[j]$. Drugim rečima, ne moramo uvek sumirati sve elemente počev od indeksa i. Implementacija ovde ideje je prikazana u *Algoritmu 4*. Složenost novog algoritma je $O(n^2)$.

```
=============================================================
01   brojParnih = 0;
02   for i = 1 to n
03      suma = 0;
04      for j = i to n
05         suma = suma + a [j];
06         if (suma mod 2 == 0) then
07            brojParnih = brojParnih + 1;
=============================================================
```

Algoritam 4

Iz gornjeg primera vidimo da istu ideju možemo implementirati na različite načine. Samom razmatranju implementacije treba posvetiti dosta vremena, nekada čak i više od ideje – **dizajna algoritma**. Naravno najbolji recept je razmatrati ih parelelno. Sa druge strane, od ideje ne treba odustati ukoliko niste dovoljno razmotrili implementaciju.

Međutim, ovaj problem možemo rešiti i u linearnom vremenu. Definišimo niz $suma$ kao

$$suma[k] = a[1] + \cdots + a[k]$$

Sumu elemenata $a[i] + \cdots + a[j]$ možemo preko niza $suma$ izračunati kao $suma[j] - suma[i-1]$, gde uzimamo da je $suma[0] = 0$.[1] Ovaj podniz će imati parnu sumu jedino u slučaju da su vrednosti $suma[j]$ i $suma[i-1]$ iste parnosti. Označimo sa m broj parnih elemenata niza $suma$, računajući i element sa indeksom 0. Ukoliko posmatramo dva parna elementa, podniz koji oni definišu (za koji su oni leva i desna granica) je parne sume. Analogno imamo i sa neparnim elementima. Kako je razlika dva broja parna ako i samo ako su članovi iste parnosti, ovo su jedini mogući slučajevi parnih podnizova. Krajnji rezultat dobijamo kao

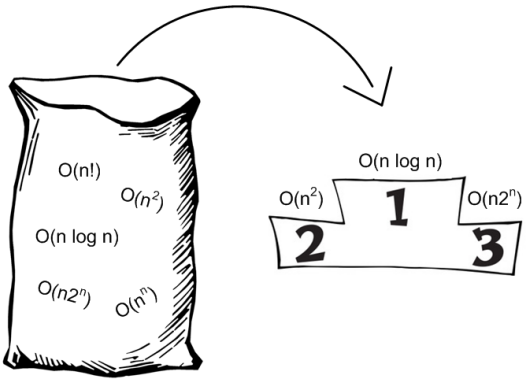
$$\binom{m}{2} + \binom{n+1-m}{2}$$

Složenost algoritma je linearna. Niz $suma$ možemo da inicijalizujemo jednim prolaskom kroz niz vezom:

$$suma[0] = 0$$

$$suma[k] = suma[k-1] + a[k], \quad k \geq 1$$

Nakon toga jednostavnim prebrojavanjem parnih odnosno neparnih elemenata niza, rešenje dobijamo gornjoj formulom.



Slika 2. Iz „vreće” mogućih algoritama treba izabrati onaj najefikasniji

[1] Ovakva konstukcija je jako česta i može se uopštiti za matrice.

Intervali

Problem Dat je niz a, dužine n, celobrojnih intervala na realnoj pravoj. Koordinate intervala su iz segmenta $[-M, M]$. Naći celobrojnu tačku sa realne prave koja pripada najvećem broju intervala. Tačka pripada intervalu i kada se nalazi na njegovom rubu.

Ovo je jedan poznatiji primer. Ovde ćemo izneti tri rešenja i upoređivati njihove složenosti.

- *Algoritam A*: Tražena tačka je celobrojna i pripada intervalu $[-M, M]$, jer za sve ostale tačke sigurno znamo da ne pripadaju ni jednom intervalu. Zato možemo proći kroz svaku tačku iz ovog segmenta i ispitati koliko je intervala sadrže. Algoritam možemo ubrzati ukoliko na početku izračunamo najmanju levu koordinatu intervala i najveću desnu koordinatu i ispitivanje svedemo samo na tačke iz ovog segmenta.

  Čak i kod ubrzane verzije segment koje treba ispitati može biti upravo $[-M, M]$ (najgori mogući slučaj). Za svaku tačku iz ovog segmenta obilazimo ceo niz intervala dužine n. Dobijamo da je ukupna složenost ovog algoritma jednaka $O(nM)$. Kako M može biti veliki broj, vidimo da ovaj pristup nije baš efikasan.

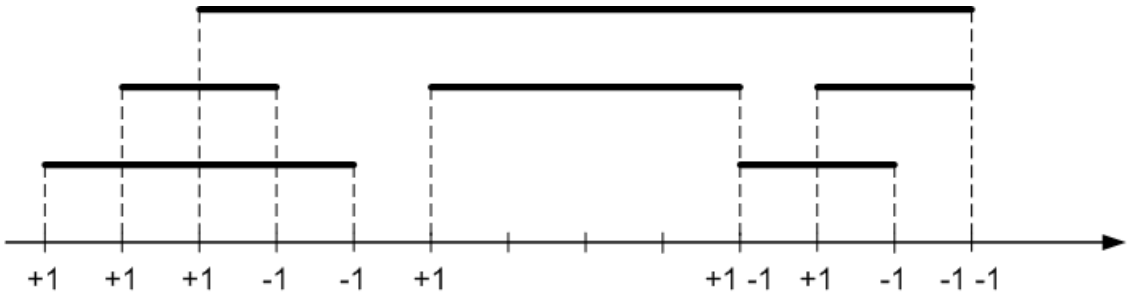
- *Algoritam B:* Označimo sa A traženu tačku (ona ne mora biti jedinstvena). Pokazaćemo da postoji tačka B koja pripada istom broju interval (dakle i ona može biti rešenje) a koja je kraj nekog od datih intervala, konkretno levog. Pretpostavimo suprotno, da tačka A nije levi kraj intervala. Posmatrajmo tačku $A - 1$. Ova tačka pripada istom broju intervala, jer smo pretpostavili da A nije levi kraj (u suprotnom bi ona ispala iz nekog intervala a kako su oni celobrojni to je nemoguće). Ukoliko je $A - 1$ leva granica nekog intervala, nju uzimamo za tačku B. U suprotnom slično rezonovanje primenjujemo na tačku $A - 2$. Kako su intervali iz konačnog segmenta u nekom trenutku moramo naići na kraj nekog od početnih segmenata koji sadrže A.

  Iz gornjeg razmatranja vidimo da ne moramo da obilazimo sve tačke iz segmenta u prvom algoritmu. Pretragu možemo vršiti samo po krajevima intervala. Krajeva ima ukupno $2n$, čime dobijamo da je novi algoritam složenosti $O(n^2)$.

- *Algoritam C:* U prethodnom algoritmu smo videli da se isptivanje može svesti na granice ulaznih intervala. Međutim, pri ispitivanju pripadnosti neke tačke ispitujemo sve intervale. Da li ovo možemo ubrzati?

  Razmatrajmo primer skupa intervala sa slike:

  $$[1, 5], [2, 4], [3, 13], [6, 10], [10,12], [11,13]$$



Slika 3. Primer intervala

Skenirajmo po koordinatama granica redom sa leva na desno i zapisujmo broj intervala kojima pripadaju:

$$1, 2, 3, 3, 2, 2, 3, 3, 3, 2$$

Primećujemo da kada naiđemo na "otvaranje" nekog intervala, njegovu levu granicu, broj pripadnosti intervalima se povećava za 1. Slično kada naiđemo na zatvaranje smanjuje se za 1. Specijalan slučaj je kada se u nekoj koordinati i otvaraju i zatvaraju neki intervali, a kako smo rekli da tačka sa granica pripada intervalima, onda prvo povećavamo broj preseka a tek kada napustimo tačku smanjujemo.

Kreirajmo dva nova niza $cord$ i $value$ dužine $2n$ i inicijalizujmo ih na sledeći način:

$$cord\,[2k-1] = l_k \quad \text{i} \quad value[2k-1] = +1$$
$$cord\,[2k] = d_k \quad \text{i} \quad value[2k] = -1$$

Dakle u niz $cord$ smo poređali vrednosti granica intervala, a paralelno u drugom nizu postavljali $+1$ za levu granicu intervala i -1 za desnu. Sortirajmo ove nizove po vrednostima niza $cord$, dok paralelno menjamo i elemente u nizu vrednosti. Ukoliko su koordinate jednake manja je ona kojom se segment otvara (dakle prednost imaju $value = 1$). Nakon sortiranja imamo sledeće stanje:

$$cord = (1, 2, 2, 3, 4, 5, 6, 10, 10, 11, 12, 13, 13)$$
$$value = (1, 1, 1, -1, -1, 1, 1, -1, 1, -1, 1, -1, -1, -1)$$

Skeniranje po koordinatama i računanje pripadnosti se sada svodi na jednostavno sabiranje elemenata niza $value$. Pravimo parcijalne sume niza $value$.

$$valueSuma = (1, 2, 3, 2, 1, 2, 3, 2, 3, 2, 3, 2, 1, 0)$$

Kako tražimo maksimalni broj intervala koji sadrži neku tačku, rezultat će biti maksimum parcijalnih suma. Složenost ovog algoritma je $O(n \log n + n) = O(n \log n)$.

Kao što vidimo za isti problem smo imali tri moguća rešenja. Sva tri algoritma su korektna što se rešenja tiče. Međutim njihove složenosti se drastično razlikuju (bar za red veličine). Gledano sa takmičarske strane, broj poena koji bi osvojili po gornjim algoritmima bi aproksimativno bio 10, 40 i 100, redom.