Solutii pentru subiectele din ziua 1, clasele XI si XII

asmin - solutie

Exista o solutie simpla, cu complexitatea **`O(N²)`**. Aceasta solutie incearca sa aleaga fiecare nod drept radacina si pt. radacina astfel fixata, calculeaza in **`O(N)`** valorile asociate nodurilor. Daca radacina este nodul **`i`**, atunci valoarea radacinii este R_i. Pentru fiecare alt nod, valoarea asociata lui este cea mai mica valoare astfel incat suma valorilor pana la tatal lui in arborele cu radacina fixata plus aceasta valoare sa dea restul cerut, la impartirea cu numarul **`K`**. Valorile sunt atribuite nodurilor de la radacina catre frunze. Astfel, o simpla parcurgere **`DF`** este de ajuns, o data ce radacina a fost fixata.

Solutia optima are complexitatea **`O(N)`**. Se fixeaza intai ca radacina nodul **`1`** si se calculeaza valorile asociate nodurilor arborelui avand radacina in nodul **`1`**, conform algoritmului descris mai sus. Astfel se obtine **`C[1]`** = suma valorilor nodurilor arborelui cand nodul **`1`** este radacina. Pentru a calcula **`C[i]`** (**`i`** diferit de **`1`**) nu este nevoie sa repetam parcurgerea **`DF`** pentru nodul **`i`** considerat radacina. In schimb, inca o parcurgere **`DF`** din nodul **`1`** este suficienta pentru a calcula costurile fiecarui nod. Sa presupunem ca vrem sa calculam **`C[i]`** si am calculat deja **`C[j]`**, unde **`j`** este tatal lui **`i`**, in arborele avand radacina fixata in nodul **`1`**.

Atunci, **`C[i] = C[j] - R[j] - minim(R[j],R[i]) + R[i] + minim(R[i],R[j])`**,

unde **`minim(a,b)`** (cu **`0<=a,b<=K-1`**) calculeaza valoarea **`x`** minima (**`x>=0`**), astfel incat **`a + x = b (mod K)`**. Se alege minimul din vectorul **`C`** si se tiparesc nodurile **`i`** avand valoarea **`C[i]`** egala cu minimul.

cautare - solutie

Se citesc nodurile, se sorteaza.

Se citesc interogarile si (prin cautari binare) se adauga la nodul respectiv sau la intervalul dintre nodurile respective. Astfel se obtin niste ponderi a nodurilor si intervalelor dintre ele.

Dinamica: se foloseste o matrice **`a[i,j]`** = costul minim pentru a forma un arbore de cautare cu nodurile de la i la j. Pentru aceasta se incearca fixarea radacinii in punctul **k** (intre **i** si **j**), iar costul arborelui se determina pe baza lui **`a[i,k]`** si **`a[k+1,j]`**.

Se afla matricea, si se afiseaza **`a[1,n]`**, dar **`O(n³)`**.

Pentru a reduce complexitatea la **`O(n²)`** trebuie sa ne dam seama ca :

radacina arborelui de la **i** la **j-1** <= radacina arborelui de la **i** la **j** <= radacina arborelui de la **i+1** la **j**.

a007 - solutie

Se observă că problema se reduce la găsirea unui centru de simetrie într-o mulţime de **n** puncte, cu coordonate întregi.

Punctul de simetrie pentru muţimea de puncte are proproietatea că simetricul fiecărui punct din mulţime faţă de centrul de simetrie este tot un punct din mulţimea de puncte. De aici obţinem că pentru a exista centru de simetrie este necesar ca **n** să fie par.

Acum dacă **n** este **par**, observăm că centrul de simetrie este mijlocul a **n/2** segmente cu capete în aceste puncte şi fiecare punct din mulţime este capăt pentru un singur segment.

Notăm cu x şi y vectorii ce reţin coordonatele celor **n** puncte.

Ordonăm punctele crescător după abscise punctele, la abscise egale vom ordona crescător dupa ordonată.

Pentru un segment cu capetele de coordonate **(x1,y1)** şi **(x2,y2)** mijlocul segmentului are coordonatele **x=(x1+x2)/2**, **y=(y1+y2)/2**.

Dacă există centru de simetrie, atunci acesta este cu centrul în mijlocul segmentul cu capetele în primul punct şi celălalt în ultimul punct (adică cu un capăt în cel mai din stânga punct şi celălalt în cel mai din stânga). Notăm acest punct cu **M**.

Pentru ca să existe centru de simetrie trebuie ca segmentele cu coordonatele (x_i,y_i) şi (x_{n-i+1},y_{n-i+1}) să aibă centrul în **M**, i=1,2, ..., **n** div 2.

Complexitate algoritm: **n**log**n**.