Problema inversiuni – descrierea soluției

Autor prof. Dan Pracsiu

Soluție O(n*n)

În timpul sortării prin inserare binară a permutării a[1],...,a[n]
Pentru fiecare i=2...n, se află (prin cautare binară) poziția p unde se va insera a[i] în șirul ordonat a[1],...,a[i-1] și rezultatul va crește cu min(p-1 , i-(p-1), apoi se va insera a[i] la pozitia p

Soluție O(n*sqrt(n))

x[1]...x[N] este permutarea data si r=[sqrt(n)]
Definim tabloul a[] prin
- a[c]=1 daca valoarea c este in secventa x[1]...x[i]
- a[c]=0 daca valoarea c nu este in secventa x[1]...x[i]

si tabloul b[] prin
- b[p]=numarul valorilor din secventa x[1], ..., x[i-1] care se afla in sectiunea [p*r ; (p+1)*r)

La fiecare pas vom citi x[i] si vom marca a[x[i]]=1. Pentru a afla cate din valorile x[1],...,x[i-1] sunt mai mici decat x[i] ar trebui sa calculam suma
a[1]+...+a[x[i]-1]. Gestionarea acestui calcul o putem face in O(r) astfel:
s=b[0]+b[1]+ ...+ b[j]+ ...+ b[p-1] unde p=[c/r],
t=a[p*r]+...+a[c-1]
s+t reprezinta numarul valorilor mai mici decat c si care se afla in secventa x[1]...x[i-1]
Daca s+t<i-1-(s+t)
    atunci x[i] se va adauga la stanga sirului construit,
        rezultat=rezultat+s+t
    altfel x[i] se va adauga la dreapta sirului construit
        rezultat=rezultat+i-1-(s+t);
Valoarea rezultat este numarul minim de inversiuni cautat.

Soluție O(n*log(n))

In timpul algoritmului de sortare, cu algoritmul merge-sort, a vectorului permutare dat x[1],...,x[n] se construieste vectorul low[] definit prin
low[c]=numarul valorilor din x[1]...x[i-1] care sunt mai mici decat c
Atunci cand avem de interclasat secventele ordonate x[p]...x[r] si respectiv x[r+1]...x[q] observam ca
    daca (i<=q && j<=r && x[i] >= x[j]), atunci low[x[j]] = low[x[j]] + (i - p);
    daca (i>q && j<=r), atunci low[x[j]] = low[x[j]] + (r+1 - p);
La final numarul minim de inversiuni se calculeaza insumand min(low[i],i-low[i]) , pentru i=1,...,n

Soluție O(n*log(n))

Presupunem că deja au fost depuse pe masă deja valorile p_1, p_2, ..., p_i și dorim să aflăm unde vom depune valoarea p_{i+1} pentru a obține cât mai puține inversiuni. Avem două cazuri:
1. depunem această valoare la început; în acest caz numărul de inversiuni care e adaugă este dat de numărul de valori din șirul p_1, p_2, ..., p_i care sunt mai mici decât p_{i+1}.
2. depunem această valoare la sfârșit; în acest caz numărul de inversiuni care e adaugă este dat de numărul de valori din șirul p_1, p_2, ..., p_i care sunt mai mari decât p_{i+1}.

Deci la fiecare pas vom calcula două valori:
- `mic` = numărul de valori din șirul p_1, p_2, ..., p_i care sunt mai mici decât p_{i+1}
- `mare` = numărul de valori din șirul p_1, p_2, ..., p_i care sunt mai mari decât p_{i+1}

Alegem să punem pe p_{i+1} la început dacă `mic < mare` sau la sfârșit dacă `mic >= mare`
Pentru a determina la fiecare pas valorile mic și mare putem utiliza arbori indexați binar.