Metoda Greedy

propune o strategie de rezolvare a problemelor de optim în care se poate obține optimul global prin alegeri succesive ale optimului local, ceea ce permite rezolvarea problemei fără nici o revenire la deciziile luate deja.

Descrierea metodei

Se dă o mulțime A cu n elemente şi se cere să se determine o submulțime a sa care să satisfacă anumite restricții. Această submulțime se numeşte *soluție posibilă*. Se cere să se determine o soluție posibilă care să maximizeze sau să minimizeze o anumită funcție obiectiv dată. Această soluție se numeşte *soluție optimă*.

Metoda Greedy lucrează în paşi astfel:

1. se consideră la fiecare pas câte un element $x \in A$;
2. la fiecare pas se ia o decizie dacă x face parte din soluția optimă, acest lucru realizându-se fie prin luarea elementelor din A într-o ordine determinată care depinde de problemă, fie folosind o funcție care le selectează într-un anumit mod;
3. dacă elementul x considerat împreună cu elementele ce fac parte din soluția optimă parțial construită nu dă o soluție posibilă, atunci x nu este adăugat la soluția optimă.

Forma generală a procedurii, într-un limbaj algoritmic, este următoarea:

```
Subprogram GREEDY (A,n)
Array a[1:n]; integer n,i;
Solutie:=0
For i:=1 to n do
        X=select(a)
        If posibil(solutie,x) then solutie=solutie∪ {x}
Return solutie;
End GREEDY
```

În acest context:

a) funcția *select(A)* selectează un element din A, notat cu x, şi-l şterge din A;

b) funcția *posibil* determină dacă x poate fi introdus sau nu în vectorul soluție. Dacă da, atunci el este reunit soluției.

Observație: Metoda Greedy nu caută să determine toate soluțiile posibile (care ar fi poate prea numeroase) şi apoi să aleagă din ele pe cea optimă, ci caută să introducă direct (să înghită) un element x din soluția posibilă;

Problemele de optimizare pot fi programate pe această cale. De exemplu, dacă trebuie determinat maximul sau minimul unei funcții de cost depinzând de elementele a_1, a_2, ...,a_n , atunci metoda Greedy alege la fiecare pas acel element care creşte sau scade cel mai mult valoarea acestei funcții.

Problema monedelor

Să se determine numărul minim de monede de 100, 50, 20, 10, 5, 1 cu care se poate achita suma S.

Pentru achitarea sumei S=276, probabil, se gândeşte astfel:

- cea mai mare monedă ce intră în 276 este 100 şi rămâne restul r=276-100=176;
- cea mai mare monedă ce intră în acest rest este 100 şi rămâne restul r=176-100=76;
- cea mai mare monedă ce intră în 76 este 50 şi rămâne rest r=76-50=26;
- cea mai mare monedă ce intră în 26 este 20 şi rămâne rest r=26-20=6;
- cea mai mare monedă ce intră în 6 este 5 şi rămâne rest r=6-5=1;
- cea mai mare monedă ce intră în 1 este 1 şi rămâne rest r=1-1=0;

În acest caz, 276=2x100+1x50+1x20+1x5+1x1, adică numărul minim de monede este 2+1+1+1+1=6. Orice altă combinație va da un număr mai mare de monede decât 6. Se observă că schema caută să "înghită" o parte cât mai mare din sumă sau resturi. În aplicarea metodei se vor folosi de la început monedele în ordinea descrescătoare a valorii lor.

Rezolvarea problemei în programul MONEDE.CPP.

Problema rucsacului

Se dă un rucsac de capacitate C şi n obiecte. Fiecare obiect i are o greutate g_i şi poate fi fracționat, fracțiunile sale fiind notate cu x_i, $0<=x_i<=1$. Fiecare obiect dacă este plasat în rucsac aduce un profit (pondere) p_i. Să se plaseze obiectele în rucsac aşa fel încât profitul să fie maxim.

Rezolvare:

În mod formal, problema cere de fapt să se maximizeze $\sum_{i=1}^{n} p_i x_i$ supusă la restricțiile $\sum_{i=1}^{n} g_i x_i$ <=M, unde $0<=x_i<=1$ şi $1<=i<=n$. Profiturile şi greutățile sunt numere reale pozitive.

O soluție posibilă (de încărcare) este orice mulțime $(x_1, x_2,\ldots,x_n)$ care care satisface condițiile $\sum_{i=1}^{n} g_i x_i$ <=M, $0<=x_i<=1$, $1<=i<=n$, iar o soluție optimă este o soluție posibilă pentru care $\sum_{i=1}^{n} p_i x_i$ este maxim.

Rezolvarea problemei în programul RUCSAC.CPP.