Laborator 11 - Tehnici de programare. Greedy

Algoritmii de tip Greedy pot fi utilizaţi atunci când se doreşte rezolvarea unei probleme de optim, a cărei soluţii se poate construi în etape. Aceşti algoritmi se caracterizează prin faptul că la fiecare pas se alege mereu cel mai bun candidat posibil, i.e., optimul local. În general, algoritmii de tip Greedy sunt eficienţi din punct de vedere al complexităţii de timp. Însă nu întotdeauna, soluţia construită este cea optimă.

În cele ce urmează vom studia câteva probleme în rezolvarea cărora vom aplica tehnica Greedy. Pentru unele dintre acestea, soluţia obţinută nu este optimă.

11.1 Problema fracţionară a rucsacului

Se consideră un rucsac având capacitatea (în kilograme) M şi n obiecte definite prin greutate şi valoare (valori strict pozitive). Să se determine o modalitate de umplere a rucsacului astfel încât valoarea totală a obiectelor introduse să fie maximală. Obiectele se pot alege şi parţial.

Pentru a rezolva această problemă se pot sorta obiectele descrescător după raportul valoare/greutate. Obiectele se vor introduce în rucsac pe rând, în ordinea obţinută. Dacă încap toate obiectele în rucsac, se vor adăuga toate.

În caz contrar, se vor adăuga obiecte întregi pe rând, astfel încât să nu se depăşească M. Ultimul obiect se poate introduce parţial, pentru a umple spaţiul rămas în rucsac.

```
struct obiect {
    float greutate;
    float valoare;
    int index;
} ob[100] ;
```

Cu ajutorul câmpului index obiectele pot fi numerotate atunci când sunt citite, deoarece, în urma sortării, se pierde ordinea iniţială a acestora.

Codul sursă al soluţiei în pseudocod este prezentată mai jos, unde sort_desc() este o funcţie care realizează sortarea descrescătoare a vectorului ob după raportul valoare/greutate:

```
rucsac(ob[]; n;M)
    sort_desc(ob, n)
```

```
        s ← 0
        i ← 0
        while s + ob[i].greutate <= M AND i < n do
            s ← s + ob[i].greutate
            write "Obiectul ", ob[i].index, " a fost introdus complet - greutatea ", ob[i].greutate, " - valoarea ", ob[i].valoare
            i ← i + 1
        end while
        if s < M AND i < n then
            write "Obiectul ", ob[i].index, " a fost introdus partial - greutatea ", M - s, " - valoarea ", ob[i].valoare*(M - s)/ob[i].greutate
        end if
    end rucsac
```

Observaţie: Este uşor de demonstrat faptul că în această implementare soluţia calculată este optimă.

Abordarea greedy prezentată mai sus nu mai garantează obţinerea unei soluţii optime în cazul în care problema se modifică astfel încât obiectele să nu poată fi introduse şi parţial în rucsac.

De realizat: descarcati codul sursa de aici si completati corespunzator codul sursa pentru determinarea solutiei, pentru urmatoarele date: numar de obiecte =7, capacitatea rucsacului M=100, iar pentru fiecare obiect:

valoare	greutate	obiectul
70	30	1
40	30.5	2
20	10	3
90	10.5	4
40	25	5
50	15	6
100	35	7

11.2 Colorarea unei hărţi (zone)

Se consideră o hartă cu n ţări, numerotate de la 0 la n - 1. Pentru oricare două ţări se înregistrează relaţia de vecinătate. Acest lucru se realizează cu ajutorul unei matrice t, astfel: dacă ţările i şi j sunt vecine, atunci t[i][j] = 1;

în caz contrar, t[i][j] = 0.

Găsiţi o modalitate de a colora harta, utilizând cât mai puţine culori, astfel încât două ţări vecine să nu aibă aceeaşi culoare.

Dispunem de un număr nelimitat de culori c0; c1; c2; ... .

Aplicând strategia Greedy, vom colora prima ţară utilizând culoarea c0.

La fiecare pas vom încerca să colorăm următoarea ţară folosind c0; dacă ţara are un vecin colorat cu c0, testăm posibilitatea de a folosi culoarea c1; continuăm până când găsim o culoare validă.

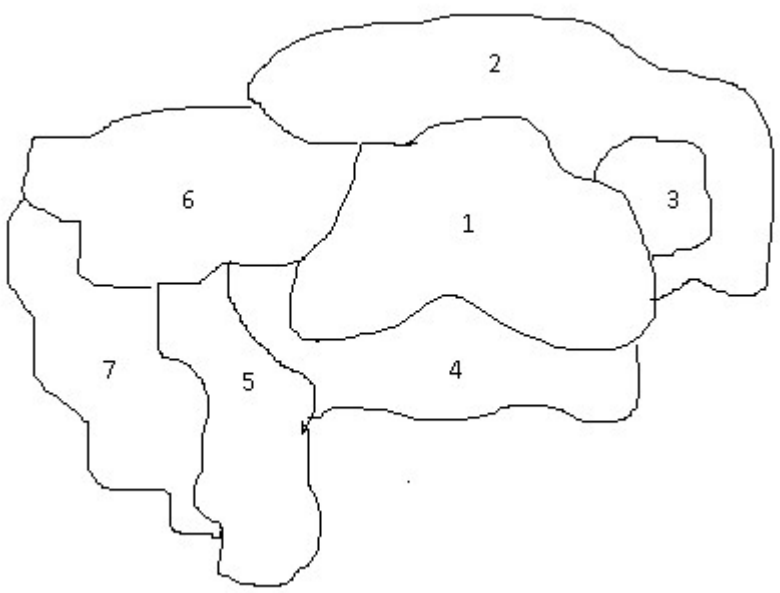
Codul sursă pseudocod al problemei de colorare hărţi este redat mai jos, în care în vectorul v se va memora culoarea corespunzătoare ţării i :

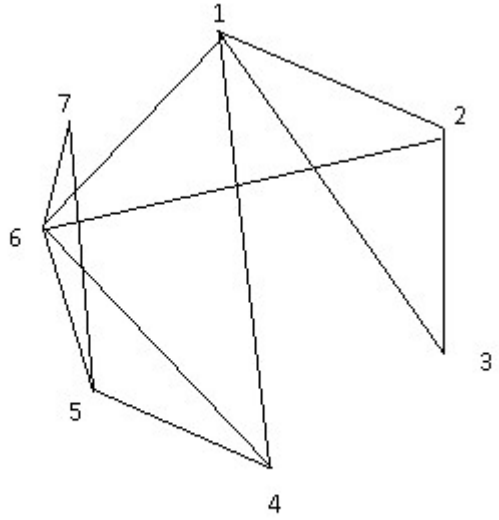
```
colorare( n; v[]; t[][])
    c ← 1
    cmax ← 1
    v[1] ← 1
    for i ← 2; n do
        c ← 1
        do
            ok ← 1
            for j ← 1; i-1 do
                if t[i][j] = 1 AND v[j]=c then
                    ok ← 0
                endif
            endfor
            if ok = 1 then
                v[i] ← c
            else
```

```
                    c ← c+1

                endif
            while ok =0
            if c > cmax then
                cmax ← c
            endif

        endfor

    end colorare
```

Exemplu: Considerăm o hartă cu 7 zone(ţări), pentru care relaţiile de vecinătate sunt reprezentate în graficul din imagine. Dacă zona i şi zona j sunt vecine, acest lucru se marchează printr-o legătură (muchie, arc) în graf.

Pentru a reprezenta graful in cod C, se ataseaza acestuia o matrice de adiacenta, de ordinul n (numar de zone), in care valoarea acesteia in pozitia (i,j) va fi 1 daca exista muchie de la zona i la j, si 0 in caz contrar.





De realizat: descarcati codul sursa de aici si completati corespunzator programul pentru determinarea solutiei.

Cu metoda Greedy descrisă anterior se obţine următoarea colorare:

Nod (zona)	1	2	3	4	5	6	7
Culoare	c_1	c_2	c_3	c_2	c_1	c_3	c_2

11.3 Pentru un vector unidimensional, determinati suma a[i]*i, astfel incat aceasta sa fie maxima. Folositi un algoritm Greedy.

Exemplu:

Pentru [10,2,13,40,5] suma maxima este 224.

Last modified: Wednesday, 18 January 2023, 1:04 PM

« PREVIOUS ACTIVITY

NEXT ACTIVITY »