PROIECT INDIVIDUAL

LA INFORMATICĂ

TEMA: TEHNICA GREEDY

Realizat de eleva clasei a XI-a „D”

Beț Daniela

Profesor: Guțu Maria

IPLT „Spiru Haret” Chișinău 2019

descrieREA METODEI **GREEDY**

Metoda de programare **Greedy** se aplică la rezolvarea **problemelor de optimizare**. Metoda dată constă în construirea soluției optime pas cu pas, iar la fiecare etappă se selectează în soluție elementul care pare a fi cel mai potrivit și optim global.

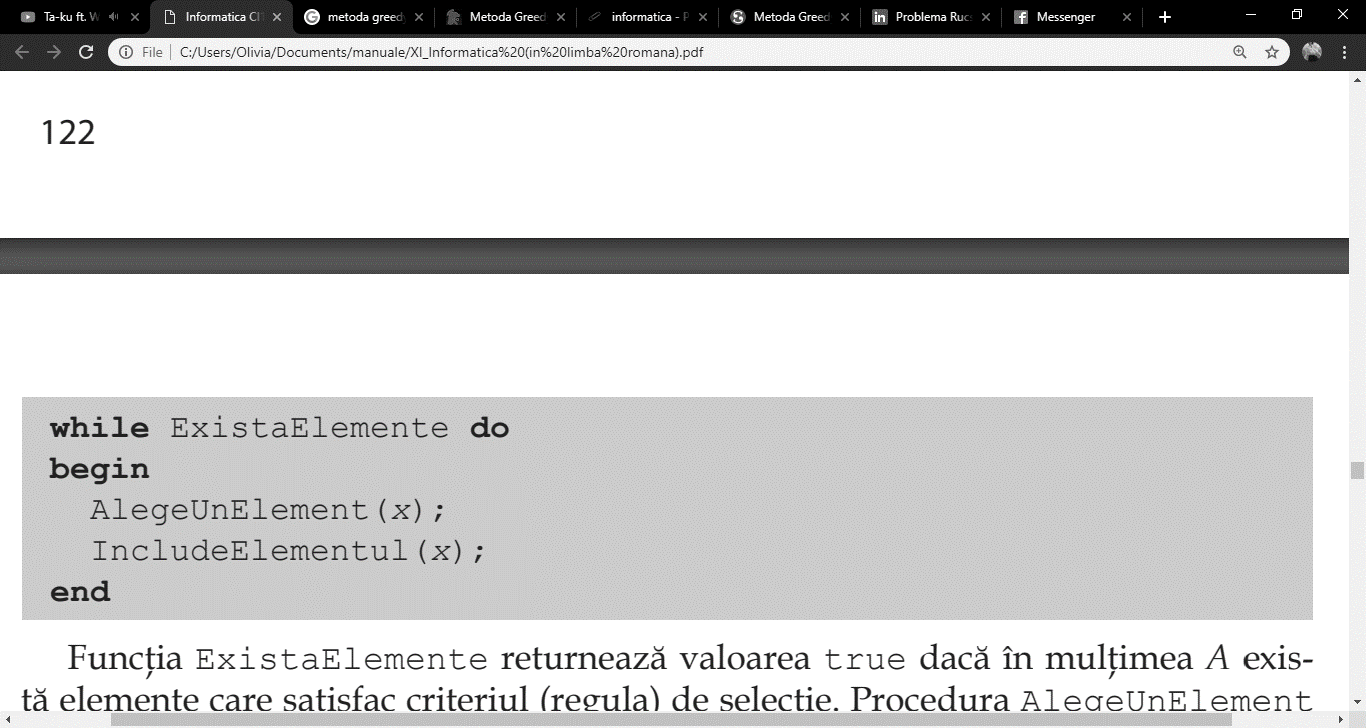
Algoritmii Greedy sunt foarte eficienți, însă nu întotdeauna conduc la soluția optimă. De asemenea nu este posibilă creearea unui criteriu general cu ajutorul căruia să verificăm corectitudinea acestei metode în rezolvarea unei anumite probleme de optimizare. Astfel, orice algoritm Greedy necesită să fie însoțit de o demonstrație a corectitudinii sale, de obicei folosindu-se prin inducție matematică.

Metoda Greedy se aplică problemelor pentru care se dă o mulțime **A** cu **n** elemente şi pentru care trebuie determinată o submulţime a sa, **S** cu **m** elemente, care îndeplinesc anumite condiţii, numite și conditii de optim.

Structura algoritmului în limbaj natural al metodei de

programare Greedy:

* se propune o mulţime A;
* se cere o submulţime S ce se include în mulțimea A care să îndeplinească anumite condiţii interne (să fie acceptabilă) și să fie optimală (să realizeze un maxim sau un minim).



**Principiul metodei Greedy:**

* Se iniţializează o mulţime a soluţiilor **S** cu mulţimea vidă ( **S=Ø )**;
* La fiecare etapă se alege un anumit element x∈A (cel mai potrivit element la momentul respectiv) care poate duce la o soluţie optimă;
* Se verifică dacă elementul ales poate fi adăugat la mulţimea soluţiilor:
  + **dacă da,** atunci elementul dat va fi adăugat, iar mulţimea soluţiilor devine **S=S∪{x}** - un element introdus în mulţimea S nu va mai putea fi eliminat;
* **dacă nu,** atunci elementul nu se mai testează ulterior.
* Procedeul de evaluare continuă, până când au fost determinate toate elementele din mulţimea soluţiilor S.

**Metoda Greedy are două componente:**

1. stabilirea soluţiei de început;
2. optimizarea acesteia.

**PROBLEME REZOLVATE:**

**1.*Suma maximă.***

Se dă o mulţime X={x1, x2, . . ., xn } cu elemente reale***.*** Să se determine o submulţime a lui X astfel încât suma elementelor submulţimii să fie maximă**.**

***Observaţie*** : Pentru rezolvarea problemei reprezentăm atât mulţimea **X,** cât şi mulţimea soluţiilor S sub forma a doi vectori de numere reale. Alegerea unui element din **X** se face in ordine, de la **1** **la n**. Funcţia **POSIBIL(B, x)** se reduce la comparaţia **x[i]>0**, iar procedura **ADAUG(B, x)** va consta din adăugarea unui element **x[i]>0** la vectorul **S** în funcţie de contorul **k**.

**Program** suma\_maxima;

**var** s,x:**array**[1..20] **of** real;

i,k,n:integer;

**begin**

**write**('Numarul de elemente n = ');

**readln**(n);

**for** i:=1 **to** n **do**

**begin**

**write**('x[',i,']= ');

**readln**(x[i]);

**end**;

k:=0;

**for** i:=1 **to** n **do**

**if** x[i]>0 **then**

**begin**

k:=k+1;

s[k]:=x[i]

**end**;

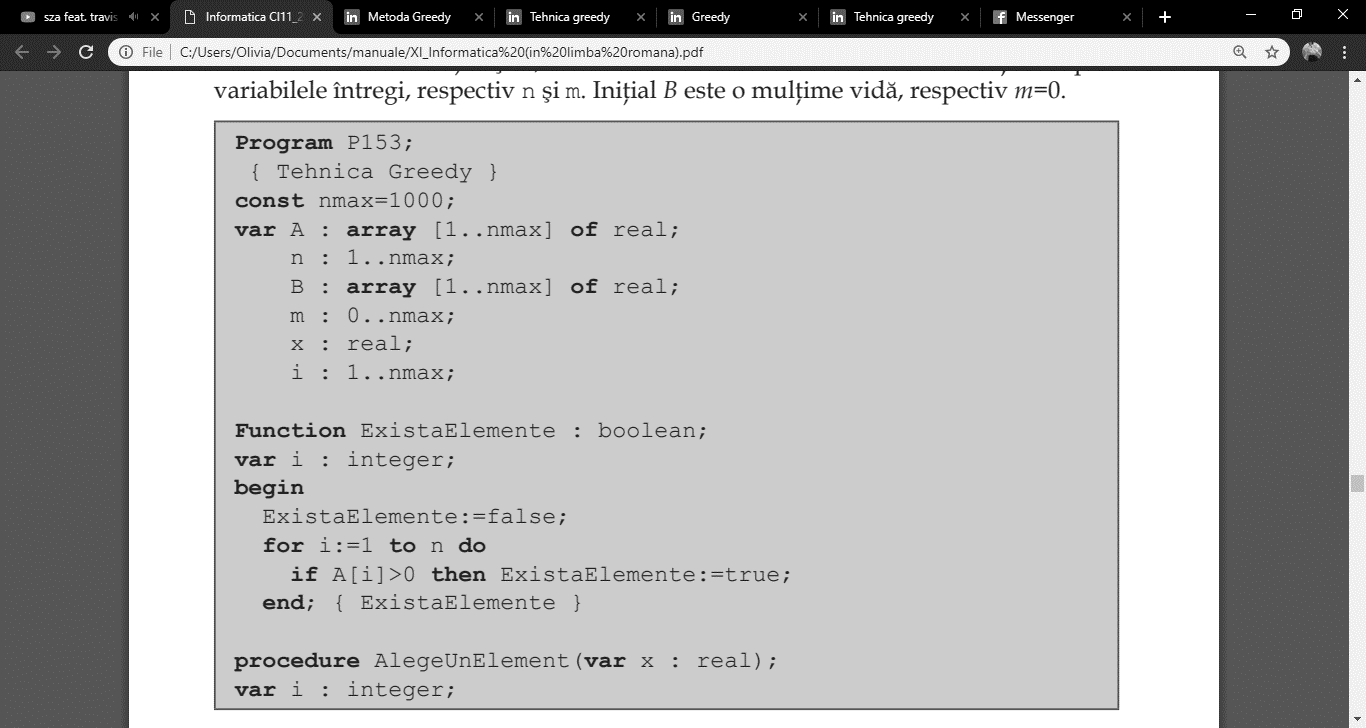
**for** i:=1 **to** k **do**

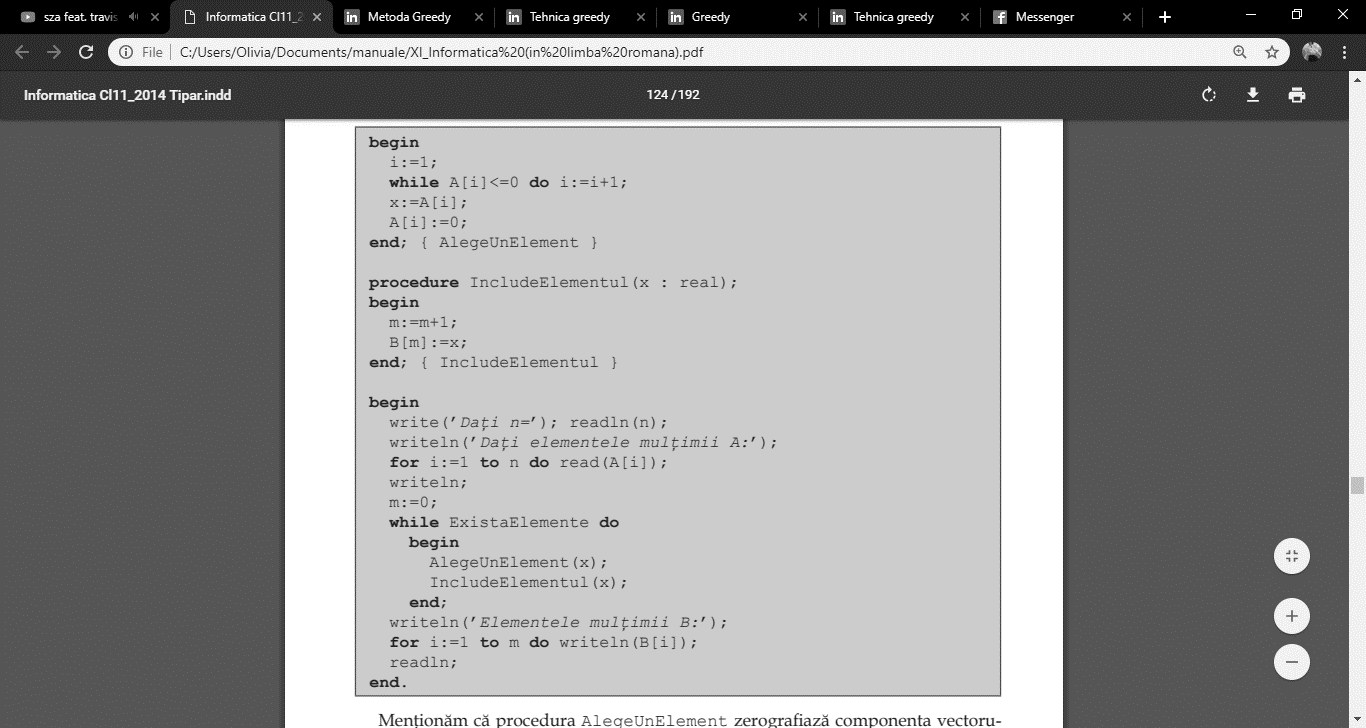
**write**(s[i]:5:2,' ');

**readln**;

**end**.

**2.*Numere reale.***

 Se consideră mulţimea A={a1, a2, ..., ai , ..., an} elementele căreia sunt numere reale, iar cel puţin unul din ele satisface condiţia ai >0. Elaboraţi un program care determină o submulţime B, B⊆A, astfel încît suma elementelor din B să fi e maximă.



**3.Problema continuă a rucsacului .**

Se dă un rucsac de o anumită capacitate, greutate și un numar de n obiecte specificandu-se masa obiectelor. Se cere un program care să determine variantă de introducere a obiectelor în rucsac astfel încât să încapă cât mai multe obiecte.

Algoritm de rezolvare:

* Citirea greutății fiecarui obiect;
* Citirea capacității rucsacului;
* Iniţializăm obiectele;
* Orsonarea obiectele crescător conform greutății ;
* Inițializarea volumului disponibil cu volumul obiectului.

*max\_ob* – numărul maxim de obiecte care pot fi puse în rucsac

*N* - numărul de obiecte disponibile.

*v\_dis* - volumul rămas disponibil din rucsac.

*O* - obiectele pe care vreau să le iau.

*Greutate* - greutatea fiecarui obiect.

**begin**

aux:=g[i];

g[i]:=g[i+1];

g[i+1]:=aux;

ok:=false;

**end;**

**end;**

*{ verifică dacă fiecare obiect încape în rucsac }*

**writeln;**  
 for i:=1 to n do write( g[i], '\*');

Gm:=0 ; i:=1;

**while** ( Gm +g[i]<=R ) **do**

**begin**

Gm:=Gm+g[i];

i:=i+1;

**end;**

**writeln**('sunt‘ ,i-1,‘ obiecte cu greutate‘ , Gm,‘) ;

**writeln** ( ‘ a ramas‘ , R-Gm,‘ loc liber‘ ) ;

**end.**

**Program** rucsac;

**Var** g:array [1..10] of **integer**;

i,n,Gm,R, aux : **integer**;

ok : **boolean**;

**begin**

**writeln**('nr obiecte'); readln(n);

**writeln**(‘capacitate rucsac'); readln(R);

**writeln**(' Obiectele de luat în rucsac:' );

**for** i:=1 to n do

**read** (g[i]);

*{ sortarea vectorului }*

ok:=false;

**while**(ok=false) **do**

**begin**

ok:=true;

**fo**r i:=1 to n-1 **do**

**if** g[i]>g[i+1] **then**

**4.Divizorii naturali.**

Se dă numărul natural k > 1. Se cere să se determine cel mai mic număr natural n având exact k divizori naturali proprii (diferiţi de 1 şi n).

**Program** k\_divizori\_naturali;

**var** v:boolean;

k,n,s,i:integer;

**procedure** VERIF(n,k:integer;var v:boolean);

**var** j,i:integer;

**begin**

i:=0;

**for** j:=2 **to** n-1 **do**

**if** n mod j = 0 **then**

i:=i+1;

**if** i = k **then**

v:=true

**else**

v:=false;

**end;**

**begin**

**write**('Numarul de divizori k > 1 '); readln(k);

**write**('Cel mai mic numar care are exact ',k,' divizori este ');

n:=k+2; s:=0;

**while** s = 0 **do**

**begin**

VERIF(n,k,v);

**if** v = true t**hen**

**begin**

**write**(n); s:=1;

**end;**

n:=n+1;

**end;**

**readln;**

**end.**

**5.Problema spectacolelor**

Într-un oraş de provincie se organizează un festival de teatru. Oraşul are o singură sală de spectacole, iar la festival şi-au anunţat participarea mai multe trupe. Aşadar, în sală, într-o zi, trebuie planificate N spectacole. Pentru fiecare spectacol se cunoaşte intervalul în care se desfăşoară: [ora\_inceput, ora\_sfarsit]. Se cere să se planifice un număr maxim de spectacole care, bineînţeles, nu se pot suprapune.

**Algoritm:**

1. Sortăm spectacolele după ora terminării lor;
2. Primul spectacol programat este cel care se termină cel mai devreme;
3. Alegem primul spectacol dintre cele care urmează în şir după ultimul spectacol programat care îndeplineşte condiţia că începe după ce s-a terminat ultimul spectacol programat;

**procedure** citire;

**var** hh, mm, i:integer;

**begin**

**write**(‘Numarul de spectacole:’);

**readln**(n);

for i:=1 to n do

**begin**

**write**(‘Spectacolul, i, incepe la:’);

**readln**(hh,mm);

v[i]. ora\_inc:=hh\*60+mm;

**write** (‘Spectacolul, i, se termina la:’);

**readln**(hh,mm);

v[i].ora\_sf:=hh\*60+mm;

v[i].ord:=i;

**end**;

**end;**

**procedure** greedy;

**var** ;integer;

**begin**

**writeln**(’Ordinea spectacolelor este:’);

ultim:=1;

nr:=1;

**write**(v[1].ord,’ ’);

for i:=2 to n do

if v[i].ora\_inc>v[ultim].ora\_sf then

**begin**

**write**(v[i].ord,’ ’);

ultim:=i;

Inc(nr);

end;

**writeln**(‘Se pot juca ’, nr, ‘ spectacole’); **end**;

**begin**

citire; sortare; greedy;

**end.**

**Program** spectacole;

**Type** spectacol=record

ora\_inc, ora\_sf:integer;

ord:integer;

**end;**

**Var** v:array[1..30] of spectacol;

n, ultim, nr:integer;

**procedure** sortare;

**var** i,j :integer; aux:spectacol;

**begin**

for i:=1 to n-1 do

for j:=i+1 to n do

if v[j].ora\_sf < v[i].ora\_sf then

begin

aux:=v[j];

v[j]:=v[i];

v[i]:=aux;

end;

**end;**

CONCLUZII:

Metoda Greedy constă în alegerea pe rând a câte un element și introducerea în mulțimea soluțiilor optime. Această metodă este mai favorabila numai în cazul rezolvării problemelor de optimizare.

Metoda Greedy este una dintre cele mai directe tehnici de proiectare a algoritmilor care poate fi aplicată la o gamă largă de probleme. Insa cu regret, metoda Greedy poate fi aplicată numai atunci cînd din enunţul problemei poate fi dedusă regula care asigură selecţia directă a elementelor necesare din mulţimea A.

Avantaje:

* Poate fi aplicata multor probleme;
* Planificarea activităților;
* Problema spectacolelor și problema fracționară a rucsacului;
* Algoritmii sunt polinomiali;
* Necesită mai puțin timp.

Dezavantaje:

* Algoritmii Greedy nu conduc în mod necesar la o solutie optimă;
* Nu este posibilă formularea unui criteriu general conform căruia să putem stabili exact dacă metoda Greedy rezolvă sau nu o anumită problemă de optimizare;
* Poate fi aplicată numai atunci cînd din enunțul problemei poate fi dedusă regula care asigură selecția directă a elementelor necesare.

DATE BIBLIOGRAFICE:

* <http://www.worldit.info/articole/algoritmica-articole/metoda-greedy/>
* <https://ru.scribd.com/doc/43454385/Metoda-Greedy#download>
* <https://sites.google.com/site/eildegez/home/clasa-xi/prezentarea-metodei-greedy>
* <https://www.infoarena.ro/metoda-greedy-si-problema-fractionara-a-rucsacului>
* <file:///C:/Users/Documents/manuale/XI_Informatica%20(in%20limba%20romana).pdf>

Mulțumesc pentru atenție!