I.P.L.T. ,,Spiru Haret,,

Referat

Tema: Tehnica GREEDY

Elev: Bondari Sofia
Clasa: 11 D
Profesor: Guțu Maria

Chișinău 2020

Cuprins:

1.Aspecte teoretice

1.2 Ce este tehnica greedy

Algoritmii *greedy* (greedy = lacom) sunt in general simpli si sunt folositi la probleme de optimizare, cum ar fi: sa se gaseasca cea mai buna ordine de executare a unor lucrari pe calculator, sa se gaseasca cel mai scurt drum intr-un graf etc. In cele mai multe situatii de acest fel avem:

- o multime de *candidati* (lucrari de executat, varfuri ale grafului etc)
- o functie care verifica daca o anumita multime de candidati constituie o *solutie posibila*, nu neaparat optima, a problemei
- o functie care verifica daca o multime de candidati este *fezabila*, adica daca este posibil sa completam aceasta multime astfel incat sa obtinem o solutie posibila, nu neaparat optima, a problemei
- o *functie de selectie* care indica la orice moment care este cel mai promitator dintre candidatii inca nefolositi
- o *functie obiectiv* care da valoarea unei solutii (timpul necesar executarii tuturor lucrarilor intr-o anumita ordine, lungimea drumului pe care l-am gasit etc); aceasta este functia pe care urmarim sa o optimizam (minimizam/maximizam)

Pentru a rezolva problema noastra de optimizare, cautam o solutie posibila care sa optimizeze valoarea functiei obiectiv. Un algoritm greedy construieste solutia pas cu pas. Initial, multimea candidatilor selectati este vida. La fiecare pas, incercam sa adaugam acestei multimi cel mai promitator candidat, conform functiei de selectie. Daca, dupa o astfel de adaugare, multimea de candidati selectati nu mai este fezabila, eliminam ultimul candidat adaugat; acesta nu va mai fi niciodata considerat. Daca, dupa adaugare,

multimea de candidati selectati este fezabila, ultimul candidat adaugat va ramane de acum incolo in ea. De fiecare data cand largim multimea candidatilor selectati, verificam daca aceasta multime nu constituie o solutie posibila a problemei noastre. Daca algoritmul greedy functioneaza corect, prima solutie gasita va fi totodata o solutie optima a problemei. Solutia optima nu este in mod necesar unica: se poate ca functia obiectiv sa aiba aceeasi valoare optima pentru mai multe solutii posibile general.

1.2 Descrierea formala a unui algoritm greedy

function *greedy*(*C*)
  {*C* este multimea candidatilor}
  $S \leftarrow \varnothing$ {*S* este multimea in care construim solutia}
  while not *solutie*(*S*) **and** $C \neq \varnothing$ **do**
    $x \leftarrow$ un element din *C* care maximizeaza/minimizeaza *select*(*x*)
    $C \leftarrow C \setminus \{x\}$
    if *fezabil*($S \cup \{x\}$) **then** $S \leftarrow S \cup \{x\}$
  if *solutie*(*S*) **then** **return** *S*
    else **return** "nu exista solutie"

Este de inteles acum de ce un astfel de algoritm se numeste "lacom" (am putea sa-i spunem si "nechibzuit"). La fiecare pas, procedura alege cel mai bun candidat la momentul respectiv, fara sa-i pese de viitor si fara sa se razgandeasca. Daca un candidat este inclus in solutie, el ramane acolo; daca un candidat este exclus din solutie, el nu va mai fi niciodata reconsiderat. Asemenea unui intreprinzator rudimentar care urmareste castigul imediat in dauna celui de perspectiva, un algoritm greedy actioneaza simplist. Totusi, ca si in afaceri, o astfel de metoda poate da rezultate foarte bune tocmai datorita simplitatii ei.

Functia *select* este de obicei derivata din functia obiectiv; uneori aceste doua functii sunt chiar identice.

Un exemplu simplu de algoritm greedy este algoritmul folosit pentru rezolvarea urmatoarei probleme. Sa presupunem ca dorim sa dam restul unui client, folosind un numar cat mai mic de monezi. In acest caz, elementele problemei sunt:

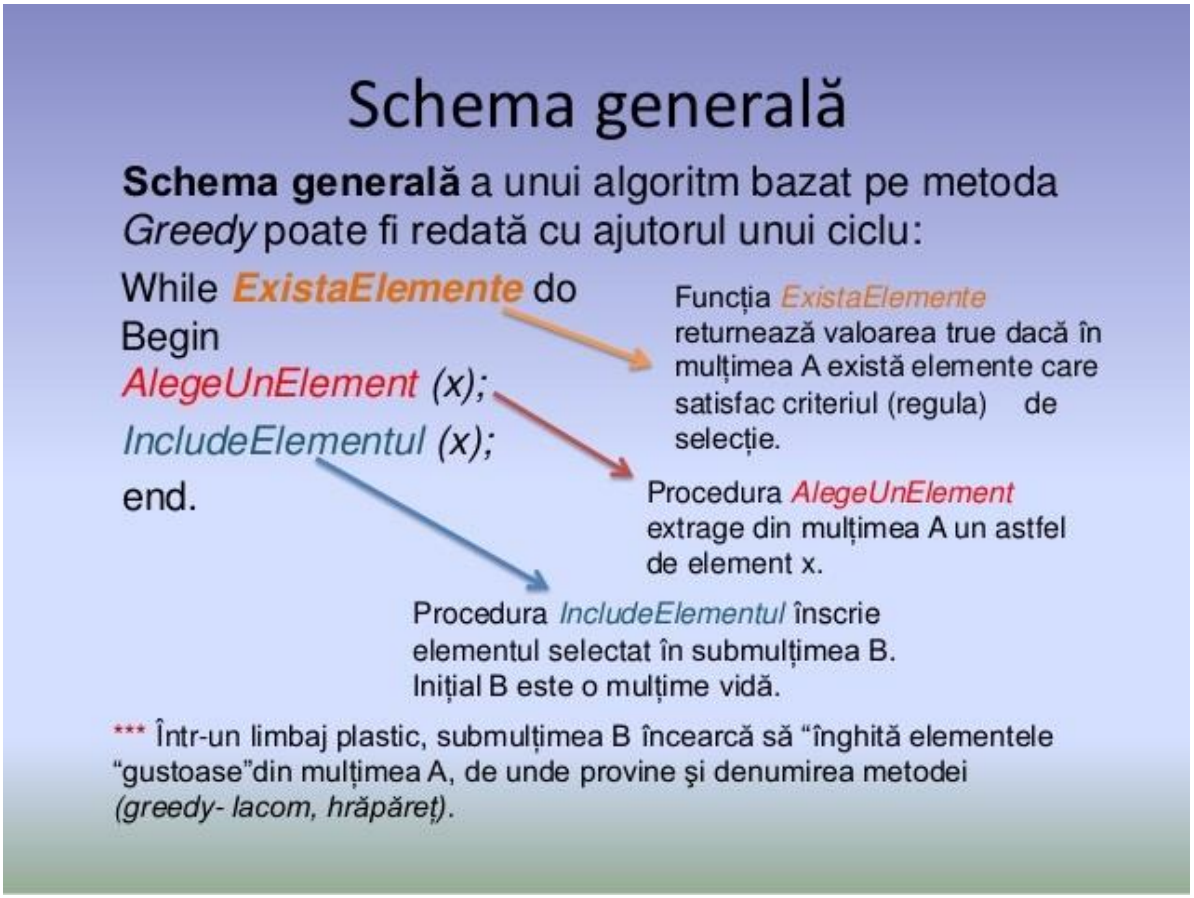
- candidatii: multimea initiala de monezi de 1, 5, si 25 unitati, in care presupunem ca din fiecare tip de moneda avem o cantitate nelimitata
- o solutie posibila: valoarea totala a unei astfel de multimi de monezi selectate trebuie sa fie exact valoarea pe care trebuie sa o dam ca rest
- o multime fezabila: valoarea totala a unei astfel de multimi de monezi selectate nu este mai mare decat valoarea pe care trebuie sa o dam ca rest
- functia de selectie: se alege cea mai mare moneda din multimea de candidati ramasa

•functia obiectiv: numarul de monezi folosite in solutie; se doreste minimizarea acestui numar

Se poate demonstra ca algoritmul greedy va gasi in acest caz mereu solutia optima (restul cu un numar minim de monezi). Pe de alta parte, presupunand ca exista si monezi de 12 unitati sau ca unele din tipurile de monezi lipsesc din multimea initiala de candidati, se pot gasi contraexemple pentru care algoritmul nu gaseste solutia optima, sau nu gaseste nici o solutie cu toate ca exista solutie.

Evident, solutia optima se poate gasi incercand toate combinarile posibile de monezi. Acest mod de lucru necesita insa foarte mult timp.

Un algoritm greedy nu duce deci intotdeauna la solutia optima, sau la o solutie. Este doar un principiu general, urmand ca pentru fiecare caz in parte sa determinam daca obtinem sau nu solutia optima. [1],[2]



1.2 Avantaje și dezavantaje

În tabelul de mai jos sunt prezentate avantajele și dezavantajele ethnic Greedy:

Avantaje	Dezavantaje
✓ Minimizarea timpului mediu de asteptare ✓ Interclasarea optima a sirurilor ordonate ✓ Ajută la determinarea celor mai scurte drumuri în graturi.	✓ Este folosită doar în două cazuri, și anume atunci când știm sigur că se ajunge la soluția dorită; Atunci când nu dispunem de o soluție în timp polinominal. ✓ Exista problem în care algoritmul clachează.

2.Rezolvare de probleme

1. O persoană are un rucsac cu care poate transporta o greutate maximă G. Persoana are la dispoziţie n obiecte si cunoaşte pentru fiecare obiect greutatea si câştigul care se obţine în urma transportului său la destinaţie. Se cere să se precizeze ce obiecte trebuie să transporte persoana în aşa fel încât câştigul sa fie maxim.

```
var a:matrice; c:tablou; f:text;
loc,n,g,i,j:integer; max,castig,dg:real;
begin
assign (f,'rucsac.txt'); reset (f);
readln(f,n,g);
for i:=1 to n do
begin readln(f,a[i,1],a[i,2]);
a[i,3]:=a[i,1]/a[i,2]; a[i,4]:=0;
end;
{sortam tabloul dupa eficienta}
for i:=1 to n-1 do
begin max:=a[i,3];loc:=i;
for j:=i+1 to n do
if a[j,3]>max then begin max:=a[j,3]; loc:=j; end;
c:=a[i]; a[i]:=a[loc]; a[loc]:=c;
end;
{Aflam cat din fiecare obiect se pune in rucsac si calculam castigul}
castig:=0;
i:=1; dg:=g;
writeln ('greutatea ','costul ','eficienta ','rucsac');
```

```
while (i<=n) and (dg>0) do
begin;
if dg>=a[i,2]
then begin castig:=castig+a[i,1];
dg:=dg-a[i,2]; a[i,4]:=1;
end
else begin castig:=castig+dg*a[i,3];
a[i,4]:=dg/a[i,2];dg:=0;
end;
writeln (a[i,1]:6:2,a[i,2]:8:2,a[i,3]:12:2,a[i,4]:10:2);
i:=i+1;
end;
writeln ('greutatea rucsacului este ',g-dg:0:2);
writeln ('costul este ',castig:0:2);
end.
```

2. Program Teatru

Program P3;

type teatru=record

ins, sfs:integer; (ora de inceput si de sfarsit a unui spectacol calculata in minute scurse fata de miezul noptii

ord:integer; (numarul de ordin al spectacolului)

end;

Var v:array [1..30] of teatru;

n, ultim, nr:integer; (n=numarul de spectacole, in variabila ultim avem in permanenta ultimul spectacol selectat, nr=numarul maxim de spectacole)

Procedure sortare_piese;

Var i,j:integer;

temp:teatru;

Begin

```
For i:=1 to n-1 do
for j:=i+1 to n do
if v[j].sfs<v[i].sfs then
begin
temp:=v[i];
v[i]:=v[j];
v[j]:=temp;
end;
Procedure citire_piese;
Var hh,mm,i:integer;
begin
Write (‘Numarul de piese de teatru n= ‘); Readln (n);
for i:=1 to n do begin
Write (‘Piesa cu nr ‘,i, cand incepe? (ora si minutul)’);
Readln (hh,mm);
v[i].ins:=hh*60+mm;
Write (‘Piesa cu nr ‘,i, cand se termina? (ora si minutul)’);
Readln (hh,mm);
v[i].ins:=hh*60+mm;
v[i].ord:=i;
end; end;
Procedure afis_piese;
Var i:integer;
```

```
Begin
Write (‘Inceputurile si sfarsiturile pieselor in minute scurse de la miezul noptii: ‘);
for i:=1 to n do
write (‘(‘,v[i].ins,’,’,v[i].sfs,’,’,v[i].ord,’)’);
writeln;
end;
Procedure algo_greedy;
Var i:integer;
Begin
Write (‘Piesele posibile, in ordine: ‘);
ultim:=1; nr:=1;
write (v[i], ‘ ‘);
for i:=2 to n do
If (v[i].ins>v[ultim].sfs) then
Begin
Write (v[i].ord, ‘ ‘);
ultim:=i;
nr:=nr+1; end;
Writeln (‘In total se pot alege maxim’,nr,’ piese’);
end;
Begin
citire_piese;
afis_piese;
```

sortare_piese;

afis_piese;

algo_greedy;

end.

3. Program Maxim

Program P1;

Var n, a1, a2, c:Integer;

Begin

a1:=-MAXINT; (initializam primele 2 numere si n cu o constanta predefinita)

a2:=-MAXINT;

n:=-MAXINT;

While n<>0 Do Begin

If (n>a1) Then a1:=n; (daca numarul n este mai mare decat primul cel mai mare numar atunci maximul este n)

If (a2<a1) Then Begin

c:=a1;

a1:=a2;

a2:=c; end; (interschimbare)

Readln (n); end;

Writeln ('a1, ' ',a2');

end.

4. Problema banilor. Scrieţi un program, care afişează modalitatea de plată, folosind un număr minim de bancnote, a unei sume întregi S de lei (S<20000). Plata se efectuează folosind bancnote cu valoarea 1, 5, 10, 50, 100, 200 şi 500 de lei. Numărul de bancnote de fiecare valoare se citeşte din fişierul text BANI.IN, care

conţine 7 rânduri, în fiecare din care sunt indicate numărul de bancnote respectiv de 1, 5, 10, 50, 100, 200 şi 500 de lei.

```
Program V3P7_02;
type tablou=array[1..3,1..7] of integer;
var s,ss,i : integer; a:tablou; f:text;
{In primul rind al tabelului vom pastra nominalul bancnotelor}
{In al doilea rind - numarul bancnotelor citite din fisier}
{In al treilea rind - numarul bancnotelor obtinute la schimb}
Procedure Afisare(sa:integer);
begin writeln('suma ',s);
if sa<>0
then writeln('nu poate fi transformata cu bancnotele date ')
else
begin writeln('se plateste cu urmatoarele bancnote');
for i:=1 to 7 do
if a[3,i]<>0
then writeln('bancnote de ',a[1,i]:6,' sau folosit ',a[3,i]);
end;
end;
Procedure calcul(var sa:integer);
var nb:integer;
begin
i:=7;
while (i>=1) and (sa>0) do
begin nb:=sa div a[1,i];
if nb<>0 then if nb>= a[2,i]
then a[3,i]:=a[2,i]
else a[3,i]:=nb;
sa:=sa-a[3,i]*a[1,i];
i:=i-1;
end;
end; { calcul }
begin
a[1,1]:=1; a[1,2]:=5; a[1,3]:=10; a[1,4]:=50;
a[1,5]:=100; a[1,6]:=200; a[1,7]:=500;
assign (f,'bani.in'); reset(f);
for i:=1 to 7 do readln(f,a[2,i]);
write ('introduceti suma de lei S ');readln(s);
ss:=s; calcul(ss); Afisare(ss);
end.
```

5. Problema magazinului

```
Program p2;
Type benz=record
ins, sfs:integer;
ord:integer; end;
Var v:array [1..100] of benz;
n, ultim, nr:integer;
Procedure citire_clienti;
Var hh, mm, i:integer;
begin
Write (‘n= ‘); Readln (n);
for i:=1 to n do begin
Write (‘Clientul cu nr. ‘,i,’cand este servit? (ora si minutul)’);
Readln (hh, mm);
v[i].ins:=hh*60+mm;
Write (‘clientul cu nr ‘, i, ‘ cand a terminat alimentarea ? ‘);
Readln (hh, mm);
v[i].sfs:=hh*60+mm;
v[i].ord:=i; end; end;
Procedure afisare_clienti;
Var i:integer;
Begin
Write (‘ cand incepe sa fie servit si cand a terminat alimentarea: ‘);
```

```
for i:=1 to n Do
Write (‘(‘v[i].ins,’,’,v[i].sfs, ‘,’,v[i].ord’)’);
Writeln;  end;
Procedure sortare_clienti;
Var i,j:integer;
t:benz;
Begin
for i:=1 to n-1 Do
for j:=i+1 to n Do
if (v[j].sfs<v[i].sfs) then
Begin
t:=v[i]; v[i]:=v[j];
v[j]:=t; end; end;
Procedure alg_greedy;
var i:integer;
Begin
Write (‘posibilii clienti, in ordine: ‘);
ultim:=1;
nr:=1;
Write (v[i].ord, ‘ ‘);
for i:=2 to n do
if (v[i].ins>v[ultim].sfs) then
begin
```

```
Write (v[i].ord, ‘ ‘);
ultim:=i;
nr:=nr+1;
end;
Writeln (‘in total se pot alege maxim’,nr, ‘clienti’);
begin
citire_clienti;
afisare_clienti;
sortare_clienti;
afisare_clienti;
alg_greedy;
END.
```

[2],[3],[4]

3.Concluzie:

Metoda Greedy este foarte eficientă atunci când dorim să aflăm rezultatul optim în cît mai scurt timp posibil, deoarece algoritmii sunt polinomiali. Cu regret, aceasta poate fi aplicată numai atunci cînd din enunțul problemei poate fi dedusă regula care asigură selecția directă a elementelor necesare din mulțimea dată. Specificul acestei metode constă în faptul că se construiește soluția optimă pas cu pas, la fiecare pas fiind selectat (sau "Înghițit") în soluție elemental care pare "cel mai bun" la momentul respectiv, în speranța că va duce la soluția optimă globală.

4.Bibliografie:

1.http://timofti7.simplesite.com/435052889

2.https://tpascalblog.wordpress.com/category/tehnica-greedy-prezentare-pascal/

3. https://www.slideshare.net/BalanVeronica/metoda-greedy1

4. http://dasinika.blogspot.com/2009/04/tehnica-greedy-pentru-problemele-pentru.html