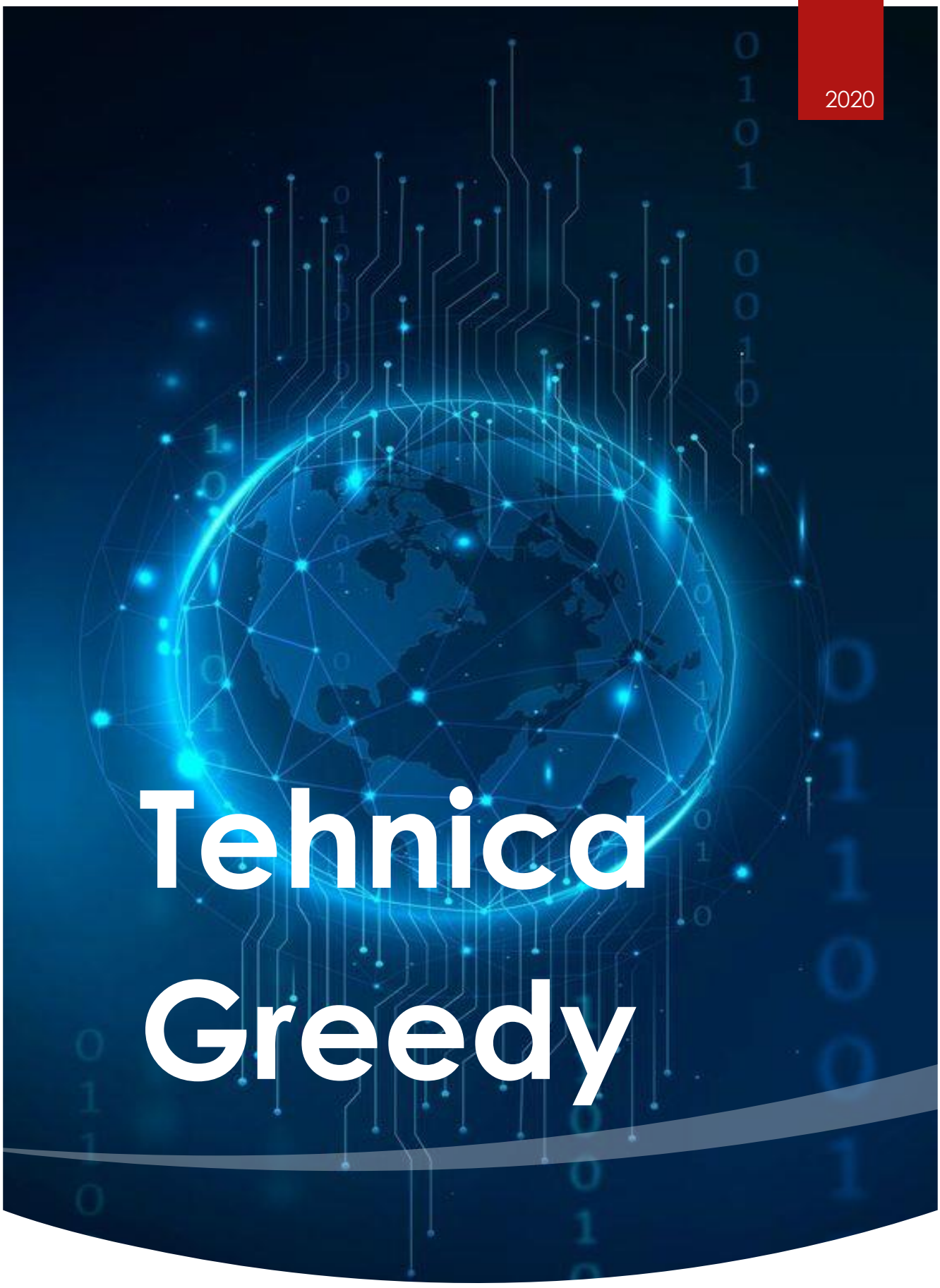
2020



Tehnica Greedy

GORGAN BOGDAN

IPLT ,,SPIRU HARET''

Cuprins

Introducere

Pe parcursul dezvoltării informaticii s-a stabilit că multe probleme de o reală importanţă practică pot fi rezolvate cu ajutorul unor metode standard, denumite **tehnici de programare**: recursia, trierea, metoda reluării, metodele euristice, **tehnica Greedy** ş.a. Tehnica de studiu ce va fi studiata in acest referat este **Tehnica Greedy**.

1. Aspecte teoretice

Metoda de programare Greedy se aplică problemelor de optimizare. Această metodă constă în faptul că se construieşte soluţia optimă pas cu pas, la fiecare pas fiind selectat în soluţie elementul care pare „cel mai bun/cel mai optim" la momentul respectiv, în speranţă că această alegere locală va conduce la optimul global .

Pentru Exemplificare:

Metoda respectivă presupune că problemele pe care trebuie să le rezolvăm au următoarea structură:

1. se dă o mulţime A={a1, a2, ..., an} formată din n elemente;
2. se cere să determinăm o submulţime B, care îndeplinește anumite condiţii pentru a fi acceptată ca soluţie.

SCHEMA GENERALĂ

Schema generală a unui algoritm bazat pe metoda Greedy poate fi redată cu ajutorul unui ciclu:

```
While ExistăElemente do
  begin
 AlegeUnElement(x);
 IncludeElementul(x);
  End.
```

Soluţia problemei se caută prin testarea consecutivă a elementelor din mulţimea A şi prin includerea unora din ele în submulţimea B. **Tehnica Greedy** se aplica problemelor de optimizare, de minim, maxim.

2. Exemple de programe

a. Problema rucsacului:

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

```
   Program rucsac;
 Type obiect=record
   C,Go,E:real;
    end;
 var ok:boolean;
   v:array[1..100] of obiect;
   aux:obiect;
   i,n,t:integer;
   G, ct:real;
Begin
  write('n='); readln(n);
   write('G='); readln(G);
  For i:=1 to n do Begin
 write('castigul pentru obiectul', i);
 readln(v[i].c);
 write('greutatea obiectului ',i);
   readln(v[i].Go);
    v[i].e:=v[i].c/v[i].Go;
End;
   t:=1;
 Repeat
  ok:=true;
   for i:=1 to n-t do
    if v[i].e<v[i+1].e then Begin
  ok:=false;
 aux:=v[i];
  v[i]:=v[i+1];
  v[i+1]:=aux;
end;
  t:=t+1;
 until ok;
  i:=1; ct:=0;
 while (G<>0) and (i<=n) do Begin
 if v[i].Go<G then Begin
  ct:=ct+v[i].c;
  G:=G-v[i].Go;
   writeln('obiectul',I,'1');
 end
 else begin
  P:=G*100/v[i].c*p)/100;
   writeln('obiectul,I,se introduce in procent de ',P);
```

```
 end;
  i:=i+1;
  end;
   writeln('castigul total este ',ct);
    end;
 readln;
End.
```

b.Cat mai multe obiecte:

Se dă un rucsac de o anumită capacitate, greutate și un numar de n obiecte specificîndu-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.

```
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 in rucsac:' );
   for i:=1 to n do
    read (g[i]);
     ok:=false;
 while(ok=false) do
  begin
   ok:=true;
 for i:=1 to n-1 do
  if g[i]>g[i+1] then
 begin
  aux:=g[i];
   g[i]:=g[i+1];
    g[i+1]:=aux;
  ok:=false;
 end;
   end;
 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' ) ;
   readln;
end.
```

c.Problema bancnotelor:

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. Intrare: Fişierul text BANI.IN şi de la tastatură se citeşte suma S. Ieşire: Dacă e posibil de plătit această sumă S, atunci la ecran se va afişa valoarea bancnotei şi numărul de bancnote respective utilizate la plată. Dacă bancnote de careva valoare nu se folosesc, atunci nu se afişează această valoare.

```
  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;
    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.
```

d. *Program Teatru :*

Sa se scrie un program care determina numarul maxim de spectacole pe care o persoana le poate viziona într-o anumită perioadă de timp.

```
   type teatru=record
    ins, sfs:integer;
 {ora de inceput si de sfarsit a unui spectacol calculata in
minute}
    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.
```

e. Problema 7,(manual pag. 125):

```
const MM = 15;
  NN = 15;
   type
 TMaze = array [1..MM, 1..NN] of byte;
  var
   abc: boolean;
 m, n, i, j : Integer;
A: TMaze =
((1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
(0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
(1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
(1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
(1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0));

 procedure L(i, j: integer);
begin
 if not abc then
```

```
  if A[i, j] = 0 then
   begin
  if (i = 1) or (i = m) or (j = 1) or (j = n) then
  abc := True;
   A[i, j] := 1;
    L(i, j - 1);
     L(i, j + 1);
    L(i - 1, j);
   L(i + 1, j);
  if abc then
   WriteLn(i, ' ', j);
   end;
    end;
  begin
   WriteLn('M, N:=');
    ReadLn(m, n);
   for i:=1 to m do
    for j:=1 to n do begin
   Write('A[', i, ',', j, ']:=');
    ReadLn(A[i, j]);
   end;
  WriteLn; WriteLn('I, J:='); ReadLn(i, j); WriteLn;
   abc := False;
  L(i, j);
   if not abc then
  WriteLn('Nu exista iesire');
 end.
```

3. Avantaje si dezavantaje

(+)Algoritmii greedy sunt performanti chiar daca problemele au dimensiuni mari. Tot timpul trebuie gasita multimea elementelor initiale ce se va optimiza in functie de criterii .

(+)Cu toate acestea, ei sunt utili, deoarece fac rapid alegeri și de multe ori dau aproximări bune ale soluţiei optime.

(-) Tehnica Greedy nu are o structura standard pentru toate tipurile de algoritm, de aceea nu se poate standardiza .

(-) Uneori, metoda Greedy nu duce la aflarea soluţiei optime. Să luăm următorul exemplu (imaginea de mai jos). Cu un obiectiv de a ajunge la cea mai mare sumă, la fiecare pas, algoritmul greedy va alege ceea ce pare a fi alegerea optimă imediat, asa ca va alege 12 în loc de 3 la al doilea pas, și nu va ajunge cea mai bună soluţie, care conţine numărul 99.

(-) Un alt exemplu ar fi urmatoarea situatie: Începând din A, un algoritm greedy va găsi maximul local din m, fără a fi conștient de maximul global din M .

(-)Algoritmii greedy dau greș în găsirea soluţiei optime globale mai ales pentru că nu operează exhaustiv pe toate datele. Ei își pot lua angajamente pentru anumite alegeri prea devreme, ceea ce îi împiedică să găsească cele mai bune soluţii globale mai târziu.

4. Concluzii

În concluzie putem spune că fiecare program necesita o abordare diferita în corespondență cu cerințele și soluțiile lui, iar Tehnica Greedyeste foarte eficienta, dar nu conduce în mod necesar la o solutie optimă. Din acest motiv, orice algoritm Greedy trebuie însotit de o demonstratie a corectitudinii sale . Demonstratia faptului că o anumită problemă are proprietatea alegerii Greedy se face de obicei prin inductie matematică . Metoda Greedy este foarte eficientă atunci cînd dorim să aflăm rezultatul optim în cît mai scurt timp posibil, deoarece algoritmii sunt polinomiali.

5. Bibliografie

- https://www.cyberforum.ru/pascalabc/
- https://www.slideshare.net/BalanVeronica/metoda-greedy1
- http://timofti7.simplesite.com/435052889
- https://forum.lazarus.freepascal.org/index.php?topic=24264.15
- https://ru.scribd.com/doc/214802757/Problema-Rucsacului
- http://dasinika.blogspot.com/2009/04/tehnica-greedy-pentru-problemele-pentru.html