IPLT "Spiru Haret"

Tehnica Greedy

Referat elaborat de eleva clasei a 11 "D"
Guzun Laura

Mai,2020

Cuprins

1. Aspecte teoretice

Pe parcursul dezvoltării informaticii s-a stabilit că multe probleme de o reală importanță practică pot fi rezolvate cu ajutorul unor metode standard, denumite tehnici de programare, printre ele se numără şi *tehnica Greedy*[2].

Metoda de programare Greedy se aplică problemelor de optimizare. Această metodă constă în faptul că se construieşte soluția optimă pas cu pas, la fiecare pas fiind selectat în soluție elementul care pare „cel mai bun/cel mai optim" la momentul respectiv, în speranță că această alegere locală va conduce la optimul global[3].

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

1. se dă o mulțime A={a_1, a_2, ..., a_n} formată din n elemente;
2. se cere să determinăm o submulțime B, care îndeplinește anumite condiții pentru a fi acceptată ca soluție.

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
```
[1]

Soluția problemei se caută prin testarea consecutivă a elementelor din mulțimea A şi prin includerea unora din ele în submulțimea B[2].

Tehnica Greedy se aplica problemelor de optimizare, de minim, respectiv maxim[4].

1.1 Principiul de functionare a metodei Greedy[3]

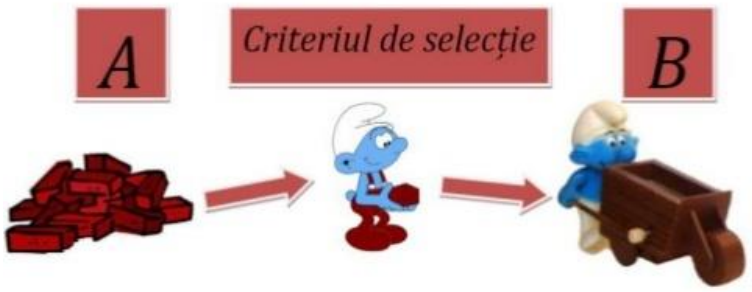
1. Se inițializează mulțimea soluțiilor S cu mulțimea vidă, S=Ø
2. La fiecare pas se alege un anumit element x∈A (cel mai promițător element la momentul respectiv) care poate conduce la o soluție optimă
3. Se verifică dacă elementul ales poate fi adăugat la mulțimea soluțiilor:
4. Dacă da, atunci :
   - va fi adăugat şi mulțimea soluțiilor devine S=SU{x} - un element introdus în mulțimea S nu va mai putea fi eliminat
5. Daca nu, atunci :
   - el nu se mai testează ulterior
6. Procedeul continuă, până când au fost determinate toate elementele din mulțimea soluțiilor.

1.2 Avantaje (+) / Dezavantaje (-)

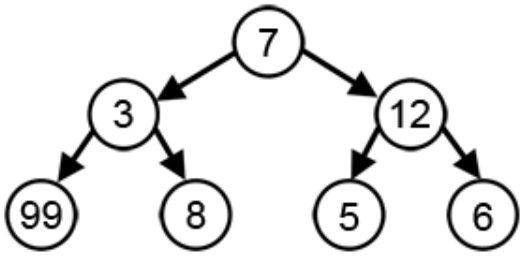
(-) Tehnica Greedy nu are o structura standard pentru toate tipurile de algoritm, de aceea nu se poate standardiza[4].

(+) 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[4].

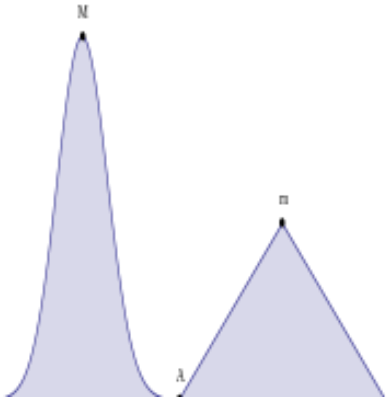
✓ *Pentru a evita trierea tuturor submulţimilor în metoda Greedy se utilizează un criteriu (o regulă) care asigură alegerea directă a elementelor necesare din mulţimea A. Evident, în absența unor astfel de criterii metoda Greedy nu poate fi aplicată[2].*



(-) Uneori, metoda Greedy nu duce la aflarea soluției 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 soluție, care conține numărul 99[6].



(-) 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*[6].



(-) Algoritmii *greedy* dau greș în găsirea soluției 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 soluții globale mai târziu. **(+)**Cu toate acestea, ei sunt utili, deoarece fac rapid alegeri și de multe ori dau aproximări bune ale soluției optime[6].

Deosebiri intre *Metoda Trierii* si *Tehinica Greedy*[2]

Tehnica Greedy	Metoda Trierii
◦ **(+)**Se aplică mai des, cu condiția că din enunțul problemei poate fi dedusă regula de selecție directă a elementelor necesare ◦ **(+)**Algoritmii sunt polinomiali (necesită mai puțin timp)	◦ Se aplică numai în scopuri didactice sau pentru elaborarea unor programe simple ◦ Algoritmii sunt exponențiali (necesită mai mult timp)

2. Probleme rezolvate

1. **Problema rucsacului [7](transportul unei greutati in functie de un anumit cost): O persoană are un rucsac cu ajutorul căruia poate transporta o greutate maximă G. Persoana are la dispoziție n obiecte şi cunoaşte pentru fiecare obiect greutatea şi câştigul care se obţtine în urma transportului său la destinație. Se cere să se precizeze ce obiecte trebuie să transporte persoana în aşa fel încat câştigul să fie maxim.**

Algoritmul:

Pas 1. Se calculează pentru fiecare obiect în parte eficienţa de transport rezultată prin împărţirea la câştigului la greutate.
Pas 2. Obiectele se sortează în ordine descrescătoare a eficienţei de transport şi se iau în calcul în această ordine
Pas 3. Câştigul iniţial va fi 0 iar greutatea rămasă de încărcat va fi G.
Pas 4. Atât timp cât nu a fost completată greutatea maxima a rucsacului şi nu au fost luate în considerare toate obiectele se procedează astfel:
• dintre obiectele neîncărcate se selectează acela cu cea mai ridicată eficienţă de trasnport şi avem două posibilităţi: a) obiectul încape in totalitate în rucsac deci se scade din greutatea rămasă de încărcat, greutatea obiectului, iar la câştig se cumulează câştigul datorat transportului acestui obiect, se tipăreşte 1 dacă întreg obiectul a fost încărcat; b) obiectul nu încape în totalitate în rucsac caz în care se calculează ce parte din el poate fi transportată, se cumulează câştigul obţinut cu transportul acestei părţi şi se tipăreşte procentul care s-a încărcat din obiect iar greutatea rămasă de încărcat devine 0.

```
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.
```

Descifrare:

- Obiect= obiectele ce se introdu în rucsac şi care sunt caracterizate prin cost(c), greutate(Go), eficienta(e).
- G= greutatea maximă ce se poate introduce în rucsac
- Ct= câştigul total ce se obţine în urma transportului obiectelor
- P= procentul care se calculează când obiectul nu se poate introduce în rucsac

2. Problema banilor[5]: Scrieți 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 conține 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.

```
Program Bani;
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; { Afisare }
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.
```

3. 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[1].

```
Program Greedy;
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]);
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.
```

4. **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[4].**

```
Program teatru ;
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.
```

5. Program Maxim[4].

```
Program P1;
Var n, a1, a2, c:Integer;
Begin
a1:=-MAXINT;  //initializam primele 2 numere si pe  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.
```

3. Concluzie

Algoritmii Greedy sunt foarte eficienti, dar nu conduc în mod necesar la o solutie optimă. Şi nici nu este posibilă formularea unui criteriu general conform căruia să putem stabili exact dacă metoda Greedy rezolvă sau nu o anumită problemă de optimizare. 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[3].

Metoda Greedy este foarte eficientă atunci cînd 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[1].

4. Bibliografie

1. http://caterinamacovenco.blogspot.com/p/metoda-greedy.html
2. https://www.slideshare.net/BalanVeronica/tehnica-greedy
3. https://sites.google.com/site/eildegez/home/clasa-xi/prezentarea-metodei-greedy
4. https://tpascalblog.wordpress.com/
5. http://dasinika.blogspot.com/2009/04/tehnica-greedy-pentru-problemele-pentru.html
6. https://ro.wikipedia.org/wiki/Algoritm_greedy
7. https://www.slideshare.net/yoanna_ioana/problema-rucsacului-presentation-948687