Proiect individual

metoda reluării

backtracking

Țurcan Valeria

cl. 11 “C”

Informație generală

Backtracking este numele unui algoritm general de descoperire a tuturor soluțiilor unei probleme de calcul, algoritm ce se bazează pe construirea incrementală de soluții-candidat, abandonând fiecare candidat parțial imediat ce devine clar că acesta nu are șanse să devină o soluție validă.

În metoda reluării se presupune că soluția problemei pe care trebuie să o rezolvăm poate fi reprezentată printr-un vector: X=(x1, x2,.., xk,…, xn). Fiecare componentă xk a vectorului X poate lua valori dintr-o anumită mulțime Ak, k=1,2,..., n. Se consideră că cele mk elemente ale fiecărei mulțimi Ak sînt ordonate conform unui criteriu bine stabilit. Cînd voaloarea pentru xk este stabilită, se verifică anumite condiții de continuare referitoare la x1, x2,..., xk.

Dezavantaje

problemele rezolvate prin această metodă necesită un timp îndelungat

este o metodă destul de complicată

Avantaje

folosește puțină memorie

structura de memorare a datelor este STIVA

Exemple de probleme

```
Program P1;
var
  x: array[1..100] of byte;
  n: byte;
  nrsol: word;

procedure scriesol;
var
  i, j: byte;begin inc(nrsol);
  writeln('Solutia a', nrsol, 'este');
  for i := 1 to n do begin writeln;
  for j := 1 to n do if x[j] = i then write('X', ' ') else write('O', ' ');
  end; end;

function potcont(k: byte): boolean;
var
  i: byte;
  atac: boolean;
  begin
  atac := false;
  for i := 1 to k - 1 do if(x[i] = x[k]) or (k - i = abs(x[k] - x[i])) then
atac := true;
  potcont := not atac;
  end;

procedure back(k: byte);
var
  i: byte;
  begin
  for i := 1 to n do
  begin
  x[k] := i;
  if potcont(k) then if k = n then scriesol else back(k + 1);
  end; end;

begin
read(n);
 nrsol := 0;
 back(1);
 writeln(nrsol, 'solutii');
 end.
```

```
program Regine;

const
  NrMaxRegine = 30;

type
  Indice = 0..NrMaxRegine;
  Solutie = array[Indice] of 0..NrMaxRegine;

var
  C: Solutie;
  n: Indice;
  NrSol: word;

procedure Afisare;var
  i, j: Indice;
  begin
  inc(NrSol);
  writeln('Solutia nr. ', NrSol);
  for i := 1 to n do
  begin
  for j := 1 to n do
  if j = C[i] then write(' * ') else write(' o ');
  writeln
  end;
  writeln;
  readln;
  end;

procedure Plaseaza_Regina(k: Indice);{cand apelam procedura Plaseaza__Regina
cu parametrul k am plasat deja regine pe liniile 1, 2, ...,k-1}
var
  i, j: Indice;
  ok: boolean;
  begin
  if k - 1 = n then {am obtinut o solutie} Afisare{prelucrarea solutiei
consta in afisare} else {trebuie sa mai plasam regine pe liniile k,k+1,...,n}
for i := 1 to n do {determin multimea MC a candidatilor pentru pozitia k}
begin ok := true; {verific daca pot plasa regina de pe linia k in coloana
i}for j := 1 to k - 1 do if (C[j] = i) or (abs(C[j] - i) = (k - j)) then ok
:= false; {regina s-ar gasi pe aceeasi coloana sau aceeasi diagonala cu o
regina deja plasata}if ok then {valoarea i respecta conditiile interne} begin
C[k] := i;{i este un candidat, il extrag imediat}Plaseaza_Regina(k + 1); end;
end; end;

begin{program principal}write('n= ');
 readln(n);
 Plaseaza_Regina(1);
 end.
```

Concluzie

În concluzie, atunci când trebuie să rezolvăm o problemă încercăm în primul rând să elaborăm un algoritm care nu se bazează pe backtracking. Dacă nu reuşim să elaborăm un astfel de algoritm sau un astfel de algoritm nu există, analizăm datele de intrare. Dacă datele de intrare au dimensiuni rezonabil de mici, astfel încât un algoritm backtracking să furnizeze soluţii în timp util, abordăm problema în această manieră.