**Proiect la informatică**

**Tema: Metoda Reluării**

**Realizat de: Nicolaescu Cătălina, 11C**

**Metoda backtracking se aplica problemelor in care solutia se poate prezenta sub forma unui vector x={x1,x2,…,xn} unde x1 apartine unei multimi S1, x2 apartine multimii S2 s.a.m.d.Si i=1…n sunt multimi finite. Cerinta problemei este, de obicei, gasirea tuturor solutiilor posibile sau gasirea numarului de solutii care satisfac anumite conditii specifice problemei. De multe ori metoda se foloseste si pentru gasirea unei singure solutii (dupa gasirea acesteia se intrerupe executia programului), a unei solutii maxime/minime insa, pentru astfel de cazuri recomandam gasirea unei alte solutii de rezolvare datorita faptului ca metoda Backtracking consuma resurse mari de memorie si timp.**

**Conditiile interne, numite si conditii de continuitate, sunt acele conditii care trebuie indeplinite de un element pentru a fi adaugat la solutie. Validitatea elementului se verifica in functie de elementele aflate in fata lui in vectorul solutie (vectorul x).**

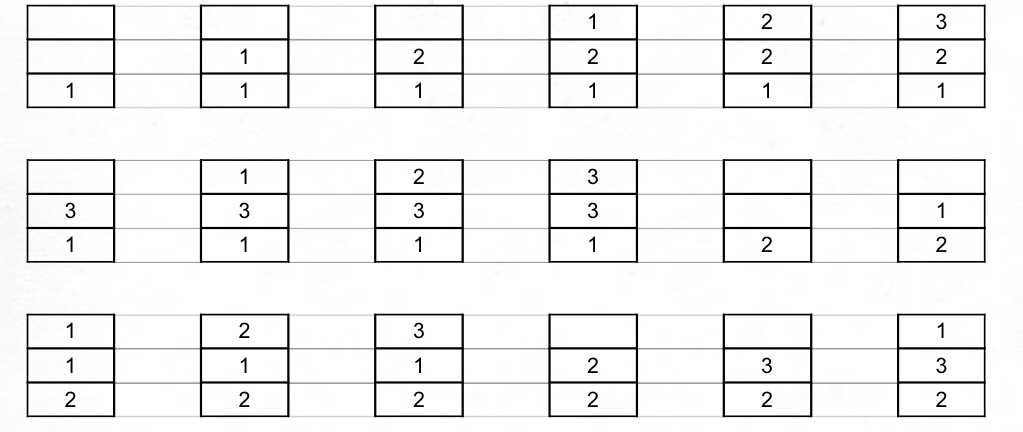
**Metoda evita generarea tuturor solutiilor posibile. Se va cauta obtinerea unei solutii prin alegerea succesiva de elemente din multimile S1, S2, S3,…,Sn. Astfel, la pasul k, se va alege un element xk din multimea Sk. Inainte de a trece la pasul k+1, se verifica daca sunt satisfacute conditiile de continuitate. In cazul in care pentru o valoare xk aleasa, conditiile de continuare nu sunt satisfacute, se va alege o alta valoare din multimea Sk, pana cand fie se va gasi o valoare care sa satisfaca conditiile de continuitate, fie se epuizeaza toate elementele multimii Sk. In cazul in care se gaseşte un element xk convenabil avem doua cazuri: fie s-a ajuns la obtinerea unei solutii caz in care se afiseaza solutia, fie nu s-a ajuns la o solutie caz in care se trece la gasirea urmatorului element din vectorul solutie (cel de pe pozitia k+1). Exista şi cazul in care pozitia k din vectorul solutie nu s-a putut completa cu un element xk din Sk, adica nu s-a gasit un element care sa indeplineasca conditiile de continuitate. In acest caz se va reveni la pasul anterior (k-1). Se va renunta la valoarea x(k-1) aleasa şi se va cauta alegerea unei alte valori (urmatoarea valoare din multimea k-1).**

**Ca orice algoritm in care sunt prezente instructiuni repetitive, algoritmul backtracking poate fi reprezentat intr-o maniera recursiva sau nerecursiva. Vom opta pentru varinata nerecursiva.**

**Generarea permutărilor. Se citeşte un număr natural n. Să se genereze toate permutările mulţimii {1, 2, 3, …,n}.**

**Generarea permutărilor se va face ţinând cont că orice permutare va fi alcătuită din elemente distincte ale mulţimii A. Din acest motiv, la generarea unei permutări, vom urmări ca numerele să fie distincte.**

**Prezentăm algoritmul corespunzător cazului n=3:**

****

**· se încarcă în stivă pe nivelul 1 valoarea 1;**

**· încărcarea valorii 1 pe nivelul al 2-lea nu este posibilă, întrucât această valoare se găseşte şi pe nivelul 1 al stivei;**

**· încărcarea valorii 2 pe nivelul al 2-lea este posibilă, deoarece această valoare nu mai este întâlnită;**

**· valoarea 1 din nivelul al 3-lea se regăseşte pe nivelul 1;**

**· valoarea 2 din nivelul al 3-lea se regăseşte pe nivelul al 2-lea;**

**· valoarea 3 pe nivelul al 3-lea nu e întâlnită pe nivelurile anterioare; întrucât nivelul 3 este completat corect. Tipărim: 1 2 3**

**……**

**Algoritmul continuă până când stiva devine vidă.**

**Probleme**

**1 Generarea permutărilor**

**Pentru această problemă trebuie specificate următoarele lucruri:**

**Numele fişierului de intrare este perm.in care conţine o singură valoare ce va fi citită în variabila de tip byte n.**

**17**

**Tipul de bază al vectorului soluţie este byte.**

**Domeniul unui element este 1..n**

**Se doresc toate soluţiile**

**Condiţiile de continuare sunt de tip condiţionare de toate cele de dinainte, şi anume diferit**

**de toate cele de dinainte.**

**La final vectorul are lungimea n**

**Construirea vectorului se face pe baza domeniului**

**Codul sursă generat este următorul: {Autor: }**

**Program Backtracking; uses crt;**

**type TSolutie=array[0..20] of Byte;**

**var v:TSolutie;**

**solutii:array[1..100] of TSolutie; nrsol:integer;**

**n:Byte;**

**procedure Citire; var f:text;**

**begin**

**assign(f,'perm.in');**

**reset(f);**

**readln(f,n);**

**Close(f);**

**end;**

**procedure Afisare; var i,k:byte;**

**begin**

**for k:=1 to nrsol do begin**

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

**write(solutii[k,i],','); end;**

**writeln;**

**end end;**

**function LungimeData(k:byte):boolean; begin**

**LungimeData:=k=n; end;**

**function final(k:byte):boolean; begin**

**final:=LungimeData(k); end;**

**18**

**function Continuare(pozitie:integer; element:Byte):boolean; var c:byte;**

**begin**

**Continuare:=true;**

**for c:=1 to pozitie-1 do**

**if v[c]=element then begin**

**end;**

**Continuare:=false;**

**exit; end;**

**procedure back(k:integer); var element:Byte;**

**y0:byte;**

**begin**

**if final(k-1) then**

**begin**

**inc(nrsol);**

**solutii[nrsol]:=v;**

**end**

**else begin**

**for y0:=1 to n do begin**

**element:=y0;**

**if Continuare(k,y0) then begin**

**v[k]:=element;**

**back(k+1); end;**

**end; end**

**end;**

**procedure Initializeaza; begin**

**nrsol:=0; end;**

**begin**

**Citire;**

**Initializeaza;**

**back(1);**

**Afisare;**

**end.**

**2 Generarea combinărilor**

**Pentru această problemă trebuie specificate următoarele lucruri:**

**Numele fişierului de intrare este comb.in care conţine două valoari ce vor fi citite în variabilele de tip byte n şi m.**

**Tipul de bază al vectorului soluţie este byte.**

**Domeniul unui element este v[k]+1..n**

**Se doresc toate soluţiile**

**19**

**La final vectorul are lungimea m**

**Construirea vectorului se face pe baza domeniului Se iniţializează poziţia 0 cu 0.**

**{Autor: }**

**Program Backtracking;**

**uses crt;**

**type TSolutie=array[0..20] of Byte;**

**var v:TSolutie;**

**solutii:array[1..100] of TSolutie; n:Byte;**

**m:Byte;**

**nrsol:word;**

**procedure Citire; var f:text;**

**begin**

**assign(f,'comb.in');**

**reset(f);**

**readln(f,n);**

**readln(f,m);**

**Close(f);**

**end;**

**procedure Afisare; var i,k:byte;**

**begin**

**for k:=1 to nrsol do begin**

**for i:=1 to m do begin**

**write(solutii[k,i],','); end;**

**writeln;**

**end end;**

**function LungimeData(k:byte):boolean; begin**

**LungimeData:=k=m; end;**

**function final(k:byte):boolean; begin**

**final:=LungimeData(k); end;**

**function Continuare(pozitie:integer; element:Byte):boolean; begin**

**Continuare:=true; end;**

**procedure back(k:integer); var element:Byte;**

**y0:byte;**

**20**

**begin**

**if final(k-1) then**

**begin**

**inc(nrsol);**

**solutii[nrsol]:=v;**

**end**

**else begin**

**for y0:=v[k]+1 to n do begin**

**element:=y0;**

**if Continuare(k,y0) then begin**

**v[k]:=element;**

**back(k+1); end;**

**end; end**

**end;**

**procedure Initializeaza; begin**

**nrsol:=0;**

**v[0]:=0; end;**

**begin**

**Citire;**

**Initializeaza;**

**back(1);**

**Afisare;**

**end.**

**3 Problema labirintului: se cere să se ajungă din camera A în camera B printr-un drum de lungime minimă.**

**Pentru această problemă trebuie specificate:**

**Ca date de intrare ce se citesc din fişierul labirint.in avem dimensiunile labirintului**

**(n,m), structura labirintului sub forma unei matrici cu 0 şi 1 (0 pentru cameră liberă şi 1**

**pentru perete), poziţia A (de plecare), poziţia B (de final).**

**Domeniul de bază al elementelor vectorului soluţie este de tip înregistrare de doi întregi**

**reprezentând coordonatele pe x, respectiv y a unei poziţii din labirint. Tipul acesta l-am**

**denumit poz.**

**Se iniţializează prima poziţie a vectorului soluţie cu poziţia A.**

**Condiţia de final este ca să ajungem în poziţia B.**

**Lungimea totală a drumului să fie minimă.**

**Drumul se construieşte plecând de la ultima poziţie parcursă (x,y) şi mutând în una din**

**poziţiile (x+1,y-1), (x+1,y+1), (x-1,y+1), (x-1,y-1).**

**Domeniul de definiţie al unei poziţii este 1..n x 1..m.**

**21**

**Se face o testare de apartenenţă a unei poziţii la domeniul de definiţie al unui element. {Autor: }**

**Program Backtracking;**

**uses crt;**

**type poz=record x:byte; y:byte; end; TSolutie=array[0..20] of pozitie;**

**var v:TSolutie; solutie:TSolutie; lungime:byte; n:Byte;**

**m:Byte;**

**a:array[1..n,1..m] of byte; ix:Byte;**

**iy:Byte;**

**fx:Byte;**

**fy:Byte;**

**min:integer;**

**procedure Citire; var f:text;**

**i1, i2:byte;**

**begin**

**assign(f,'labirint.in');**

**reset(f);**

**readln(f,n);**

**readln(f,m);**

**for i1:=1 to n do for i2:=1 to m do**

**readln(f,a[i1,i2]);**

**readln(f,ix);**

**readln(f,iy);**

**readln(f,fx);**

**readln(f,fy);**

**Close(f);**

**end;**

**procedure Afisare; var i,k:byte;**

**begin**

**for i:=1 to min do begin**

**write('(');**

**write(solutie[i].x,',');**

**write(solutie[i].y,',');**

**writeln(')'); end;**

**end;**

**function UltimaPozitie(k:byte):boolean; begin**

**UltimaPozitie:=(v[k].x=fx) and(v[k].y=fy); end;**

**function final(k:byte):boolean; begin**

**final:= (a[v[k].x,v[k].y]=0) and UltimaPozitie(k); end;**

**22**

**function Continuare(pozitie:integer; element:pozitie):boolean; var c:byte;**

**begin**

**Continuare:=true;**

**for c:=1 to pozitie-1 do**

**if (v[c].x=element.x) and(v[c].y=element.y) then begin**

**end;**

**end end;**

**Continuare:=false;**

**exit; end;**

**procedure back(k:integer); var element:pozitie;**

**begin**

**if final(k-1) then**

**if k<min**

**then begin**

**min:=k-1;**

**solutie:=v;**

**end else**

**else begin**

**element.x:=v[k-1].x+1; element.y:=v[k-1].y+1; if Continuare(k,element) then begin**

**v[k]:=element;**

**back(k+1); end;**

**element.x:=v[k-1].x+1; element.y:=v[k-1].y-1; if Continuare(k,element) then begin**

**v[k]:=element;**

**back(k+1); end;**

**element.x:=v[k-1].x-1; element.y:=v[k-1].y+1; if Continuare(k,element) then begin**

**v[k]:=element;**

**back(k+1); end;**

**element.x:=v[k-1].x-1; element.y:=v[k-1].y-1; if Continuare(k,element) then begin**

**v[k]:=element;**

**back(k+1); end;**

**procedure Initializeaza; begin**

**v[1].x:=ix;**

**v[1].y:=iy;**

**min:=10000;**

**23**

**end;**

**begin**

**Citire;**

**Initializeaza;**

**back(2);**

**Afisare;**

**end.**

**4 Sa se afiseze solutiile reale ale ecuatiei de gradul al doilea.**

**Analiza problemei - elaborarea algoritmului:**

**Fie ecuatia de gradul II ax2+bx+c=0**

**-daca toti coeficientii ecuatiei sunt egali cu 0 atunci avem o ecuatie**

**nedeterminata care are o infinitate de solutii (S=R).**

**-daca a,b=0 ,iar c<>0 atunci avem o ecuatie care nu are solutii.**

**-daca a=0 ,b,c <>0 atunci ecuatia se reduce la o ecuatie de gradul I care**

**are o singura solutie x=-c/b.**

**-daca a,b,c <>0 atunci trebuie calculat discriminantul (delta) ecuatiei d=b\*b-4\*a\*c**

**-daca d>=0 atunci ecuatia are solutii reale x1,2=(-b+-sqrt(d))/(2\*a)**

**-daca d<0 atunci ecuatia nu are solutii reale.**

**program ecuatie;**

**var a,b,c,d:real;**

**BEGIN**

**write('a=');readln(a);**

**write('b=');readln(b);**

**write('c=');readln(c);**

**if a=0 then**

**if b=0 then**

**if c=0 then**

**writeln('Ecuatie nedeterminata, S=R')**

**else writeln('Ecuatia nu are solutii.')**

**else writeln('Ecuatie de gradul I cu solutia x=',-c/b:6:2)**

**else**

**begin**

**d:=b\*b-4\*a\*c;**

**if d>=0 then**

**begin**

**writeln('x1=',(-b-sqrt(d))/(2\*a):6:2);**

**writeln('x2=',(-b+sqrt(d))/(2\*a):6:2);**

**end**

**else writeln('Ecuatia nu are solutii reale.');**

**end;**

**readln;**

**END.**

**5 Turnuri de cuburi**

**Se dau n cuburi numerotate 1,2,...,n, de laturi Li si culori Ci, i=1,2,...,n (fiecare culoare este codificata printr-un caracter). Sa se afiseze toate turnurile care se pot forma luând k cuburi din cele n disponibile, astfel încât:**

**-laturile cuburilor din turn sa fie in ordine crescatoare;**

**-culorile a oricare doua cuburi alaturate din turn sa fie diferite.**

**program cuburi;**

**type stiva=array [1..100] of integer;**

**var st:stiva;**

**i,n,p,k:integer;**

**as,ev:boolean;**

**L:array [1..10] of integer;**

**C:array [1..10] of char;**

**procedure init(k:integer;var st:stiva);**

**begin**

**st[k]:=0;**

**end;**

**procedure succesor(var as:boolean;var st:stiva;k:integer);**

**begin**

**if st[k]<n then**

**begin**

**st[k]:=st[k]+1;**

**as:=true;**

**end**

**else as:=false;**

**end;**

**procedure valid(var ev:boolean;st:stiva;k:integer);**

**var i:integer;**

**begin**

**ev:=true;**

**for i:=1 to k-1 do if L[st[k]]<=L[st[i]] then ev:=false;**

**if C[st[k]]=C[st[k-1]] then ev:=false;**

**end;**

**function solutie(k:integer):boolean;**

**begin**

**solutie:=(k=p);**

**end;**

**procedure tipar;**

**var i:integer;**

**begin**

**for i:=1 to p do write(st[i],' ');**

**writeln;**

**end;**

**begin**

**write('n= ');read(n);**

**write('p= ');read(p);**

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

**begin**

**write('L[',i,']=');readln(L[i]);**

**write('C[',i,']=');readln(C[i]);**

**end;**

**k:=1;init(k,st);**

**while k>0 do**

**begin**

**repeat**

**succesor(as,st,k);**

**if as then valid(ev,st,k);**

**until (not as) or (as and ev);**

**if as then if solutie(k) then tipar**

**else begin**

**k:=k+1;**

**init(k,st);**

**end**

**else k:=k-1;**

**end;**

**end.**

**Dintr-un nr. de 6 cursuri optionale un elev trebuie sa aleaga 3. Sa se afiseze toate posibilitatile de alegere precum si nr. lor.**

**program cursuri;**

**const n=6;**

**p=3;**

**type stiva=array [1..10] of integer;**

**var st:stiva;**

**ev,as:boolean;**

**k:integer;**

**procedure init(k:integer;var st:stiva);**

**begin**

**if k>1 then st[k]:=st[k-1]**

**else if k=1 then st[k]:=0;**

**end;**

**procedure succesor(var as:boolean;var st:stiva;k:integer);**

**begin**

**if st[k]<n-p+k then begin st[k]:=st[k]+1;**

**as:=true;**

**end**

**else as:=false;**

**end;**

**procedure valid(var ev:boolean;var st:stiva;k:integer);**

**var i:integer;**

**begin**

**ev:=true;**

**for i:=1 to k-1 do if st[i]=st[k] then ev:=false;**

**end;**

**function solutie(k:integer):boolean;**

**begin**

**solutie:=(k=p);**

**end;**

**procedure tipar;**

**var i:integer;**

**begin**

**for i:=1 to p do write (st[i]);**

**writeln;**

**end;**

**begin;**

**k:=1;init(k,st);**

**while k>0 do**

**begin**

**repeat**

**succesor (as,st,k);**

**if as then valid(ev,st,k);**

**until (not as) or (as and ev);**

**if as then**

**if solutie(k) then tipar**

**else begin**

**k:=k+1;**

**end**

**else k:=k-1;**

**end;**

**readln;**

**end.**

**6 Lui IRINEL îi plac nr. formate numai din cifre pare cifre aflate în ordin\_\_\_\_\_\_\_escatoare. Sa se determine si sa se afiseze pe ecran toate nr. de n cifre (0<n<10) care îi plac lui Gigel. Valoarea lui n este un nr. natural care se citeste de la tastatura.**

**program nr\_lui\_IRINEL;**

**type stiva=array[1..100] of integer;**

**var st:stiva;**

**i,n,k:integer;**

**as,ev:boolean;**

**procedure init(k:integer;var st:stiva);**

**begin**

**st[k]:=-1;**

**end;**

**procedure succesor(var as:boolean;var st:stiva;k:integer);**

**begin**

**if st[k]<9 then begin st[k]:=st[k]+1;**

**as:=true;**

**end**

**else as:=false;**

**end;**

**procedure valid(var ev:boolean;st:stiva;k:integer);**

**var i:integer;**

**begin**

**ev:=true;**

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

**if st[i] mod 2 <> 0 then ev:=false;**

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

**if st[i]<st[i+1] then ev:=false;**

**end;**

**function solutie(k:integer):boolean;**

**begin**

**solutie:=(k=n);**

**end;**

**procedure tipar;**

**var i:integer;**

**begin**

**for i:=1 to n do write(st[i]);**

**writeln;**

**end;**

**begin**

**write('n= ');readln(n);**

**k:=1 ;init(k,st);**

**while k>0 do**

**begin**

**repeat**

**succesor(as,st,k);**

**if as then valid(ev,st,k);**

**until (not as) or (as and ev);**

**if as then if solutie(k) then tipar**

**else begin**

**k:=k+1;**

**init(k,st);**

**end**

**else k:=k-1;**

**end;**

**readln;**

**end.**

**Bibliografie**

* [**https://sites.google.com/site/matriciinformaticapascal/home/metoda-backtracking**](https://sites.google.com/site/matriciinformaticapascal/home/metoda-backtracking)
* [**http://www.scritub.com/stiinta/informatica/c/Probleme-rezolvate-si-exerciti63855.php**](http://www.scritub.com/stiinta/informatica/c/Probleme-rezolvate-si-exerciti63855.php)
* [**http://www.scritub.com/stiinta/informatica/METODA-BACKTRACKING1055131414.php**](http://www.scritub.com/stiinta/informatica/METODA-BACKTRACKING1055131414.php)