

**Laborator 9**  
**Metoda Backtracking**

**Probleme rezolvate**

1. Sa se genereze toti vectorii cu n elemente in care sa fie 0 de m ori, 1 de p ori si 2 de q ori.

**Rezolvare:**

Codul sursa este urmatorul:

```
#include<iostream.h>
int st[20],k,p,as,ev,n,m,q,i;
void init(int k,int st[ ])
{
    st[k]=-1;
}
int sucesor(int k,int st[ ])
{
    if ( st[k]<2 && k<=n )
        {
            st[k]=st[k]+1;
            as=1;
        }
    else as=0;
    return as;
}
int valid(int k,int st[ ])
{
    ev=1;
    return ev;
}
int solutie(int k)
{
    int a=0,b=0,c=0;
    if(k==n)
    {
        for (i=1;i<=n;i++)
            switch(st[i])
            {
                case 0:a++; break;
                case 1:b++; break;
                case 2:c++; break;
            }
        if ((a==m) && (b==p) && (c==q)) return 1;
        else return 0;
    }
    else return 0;
}
void tipar(void)
{
    for(int i=1;i<=k;i++)
```

```

        cout<<" "<<st[i];
    cout<<"\n";
}
int main(void)
{
    cout<<"Dati n = "; cin>>n;
    cout<<"Dati m = "; cin>>m;
    cout<<"Dati p = "; cin>>p;
    cout<<"Dati q = "; cin>>q;
    k=1; init(k,st);
    while (k>0)
    {
        do{
            as=sucesor(k,st);
            if (as) ev=valid(k,st);
        }while (!( !as || (as && ev)));
        if (as)
            if (solutie(k)) tipar();
            else
            {
                k++;
                init(k,st);
            }
        else
            k--;
    }
}

```

## 2. Variante de PRONOSPORT

Să se determine toate variantele de pronosport cu 13 rezultate din (1,x,2) care conțin exact n1 simboluri '1'; nx simboluri 'x' și n2 simboluri '2' ( cu condiția  $n1 + nx + n2 = 13$  ).

### **Rezolvare:**

Codul sursa este urmatorul:

```

#include<iostream.h>
int st[20],k,p,as,ev,n1,n2,nx;
/* in stiva vom pune:1,2,3(pentru x) */

void init(int k,int st[ ])
{
    st[k]=0;
}

int sucesor(int k,int st[ ])
{
    if ( st[k]<3 )
    {
        st[k]=st[k]+1;
        as=1;
    }
    else as=0;
    return as;
}

```

```
int valid(int k,int st[ ])
{
    ev=1;
    int y=0,z=0,t=0,i;
    for (i=1;i<=k;i++)
        if (st[i]==1) y++;
        else if (st[i]==2) z++;
        else t++;
    if (y<=n1 && z<=n2 && t<=nx) ev=1;
    else ev=0;
    return ev;
}

int solutie(int k)
{
    int y=0,z=0,t=0,i;
    for (i=1;i<=k;i++)
        if (st[i]==1) y++;
        else if (st[i]==2) z++;
        else t++;
    if (y==n1 && k==13 && z==n2 && t==nx) return 1;
    else return 0;
}

void tipar(void)
{
    int i;
    for (i=1;i<=13;i++)
        switch (st[i])
            {
                case 1:cout<<"1 ";break;
                case 2:cout<<"2 ";break;
                case 3:cout<<"x ";break;
            }
    cout<<"\n";
}

int main(void)
{
    cout<<"Dati n1 = "; cin>>n1;
    cout<<"Dati nx = "; cin>>nx;
    n2 = 13 - n1 - nx;
    k=1; init(k,st);
    while (k>0)
    {
        do{
            as=sucesor(k,st);
            if (as) ev=valid(k,st);
        }while (!( !as || (as && ev)));
        if (as)
            if (solutie(k)) tipar();
            else
            {
                k++;
                init(k,st);
            }
        else
    }
}
```

```
        k--;  
    }  
}
```

3. Generarea numerelor binare.

Sa se genereze toate numere binare de 6 cifre care sa aiba 2 cifre de "0" si 4 cifre de "1".

**Rezolvare:**

Codul sursa este urmatorul:

```
#include<iostream.h>  
int st[20],k,p,as,ev;  
  
void init(int k,int st[ ])  
{  
    st[k]=-1;  
}  
  
int sucesor(int k,int st[ ])  
{  
    if ( st[k]<1 && k<=6 )  
        {  
            st[k]=st[k]+1;  
            as=1;  
        }  
    else as=0;  
    return as;  
}  
  
int valid(int k,int st[ ])  
{  
    int i,j=0;  
    ev=1;  
    if(k>2)  
        {  
            for(i=1;i<=k-1;i++)  
                if(st[i]==0) j++;  
            if((st[k]==0) && (j==2)) ev=0;  
        }  
    return ev;  
}  
  
int solutie(int k)  
{  
    int j=0,i;  
    for(i=1;i<=k;i++)  
        if(st[i]==0) j++;  
    if((k==6) && (j==2)) return 1;  
    else return 0;  
}
```

```
void tipar(void)
{
    int i;
    for (i=1;i<=k;i++)
        cout<<st[i]<<" ";
    cout<<"\n";
}

int main(void)
{
    int a=0;        // numar cate solutii sunt
    k=1; init(k,st);
    while (k>0)
    {
        do{
            as=succesor(k,st);
            if (as) ev=valid(k,st);
        }while (!( !as || (as && ev)));
        if (as)
            if (solutie(k)){ a++; tipar();    }
            else
            {
                k++;    init(k,st);
            }
        else
            k--;
    }
}
```

**Probleme propuse spre rezolvare**

1. Să se ruleze programele prezentate mai sus, urmărind apelurile și valorile parametrilor de apel.
2. Să se afișeze toate soluțiile ecuației în mulțimea numerelor naturale:  $3x+y+4xz=100$ .
3. Folosind metoda backtracking sa se genereze in ordine lexicografica cuvintele de cate pentru litere din multimea  $A=\{a,b,c,d,e\}$ , cuvinte care nu contin doua vocale alaturate. Primele 8 cuvinte generate sunt, in ordine: abab, abac, abad, abba, abbb, abbc, abbd, abbe.
4. Să se determine toate cuvintele ce conțin numai literele a,b,c de lungime 10 care conțin exact 3 simboluri 'a'; 4 simboluri 'b' și 3 simboluri 'c'.
5. Să se determine toate numerele în baza 8 de lungime 10 care conțin cel mult 5 cifre de 7 și exact 3 cifre 0.
6. Să se afișeze toate posibilitățile de colorare a unui graf neorientat cu n vârfuri folosind m culori astfel încât două noduri vecine să aibă culori diferite.
7. Avem la dispoziție 6 culori: alb, galben, roșu, verde, albastru și negru. Să se precizeze toate drapelele tricolore care se pot proiecta, știind că trebuie respectate următoarele reguli:
  - orice drapel are culoarea din mijloc galben sau verde;
  - cele trei culori de pe drapel sunt distincte.Exemple: "alb galben roșu", "roșu verde galben"  
Indicatii: Folosim o stivă cu 3 nivele, reprezentând cele trei culori de pe drapel și codificăm culorile prin numere:  
1 – alb  
2 – galben  
3 – roșu  
4 – verde  
5 – albastru  
6 - negru
8. Să se descompună un număr natural N, în toate modurile posibile, ca sumă de P numere naturale ( $P \leq N$ ).  
Indicatii: Folosim o stivă cu P nivele, unde fiecare nivel ia valoarea unui termen din sumă. Variabila S reține suma primelor k nivele pentru a putea testa validitatea elementului așezat pe poziția curentă fără a mai face o parcurgere suplimentară a stivei. Condiția de validitate devine:  $S+ST[k] < N$ .  
Pentru a evita afișarea descompunerilor identice, cu ordinea termenilor schimbată, forțăm ordinea crescătoare (nu strict crescătoare!) a termenilor din sumă, prin inițializarea unui nivel din stivă cu valoarea nivelului anterior.
9. Se dă un număr natural par N. Să se afișeze toate șirurile de N paranteze care se închid corect.  
Indicatii: Un șir de N paranteze care se închid corect, este un șir în care fiecărei paranteze deschise îi corespunde o paranteză închisă la o poziție ulterioară în șir.  
Exemple de șiruri corecte:  $(())$ ;  $((()))$ ;  $()()()$   
Exemple de șiruri incorecte:  $(())()$ ;  $((())$ ;  $)()()$

## PROIECTAREA ALGORITMILOR

### Laborator 9

Deducem din propoziția de mai sus o primă condiție necesară pentru ca un șir de paranteze să se închidă corect și anume că nu trebuie să deschidem mai mult de  $N/2$  paranteze. După cum se vede însă din aceste exemple (incorecte), această condiție nu este suficientă:  $(( ))( ; ))((( ; ))(( ( .$  A doua condiție este să nu închidem mai multe paranteze decât am deschis.

Pentru formalizarea condițiilor notăm cu  $N_d(k)$  și  $N_i(k)$  numărul de paranteze deschise, respectiv închise, până la poziția  $k$  a șirului, inclusiv. Cele două condiții sunt:

1.  $N_d(k) \leq N/2, 1 \leq k \leq n$
2.  $N_d(k) \geq N_i(k), 1 \leq k \leq n$

Pentru implementare folosim o stivă cu  $N$  nivele:

st[i] = 1, dacă paranteza de pe poziția  $i$  este deschisă

2, dacă paranteza de pe poziția  $i$  este închisă.

În variabilele  $N_d$  și  $N_i$  avem numărul de paranteze de fiecare fel așezate în șir până la poziția curentă,  $k$ .

Pentru a așeza o paranteză deschisă pe poziția curentă trebuie să fi deschis mai puțin de  $N/2$  paranteze, adică  $N_d < N/2$ .

Pentru a așeza o paranteză închisă pe poziția curentă trebuie să mai avem o paranteză pe care să o închidem, adică  $N_d > N_i$ .

La urcarea și coborârea în stivă se modifică  $N_d$  sau  $N_i$  în funcție de tipul parantezei de pe nivelul respectiv.

10.  $N$  copii se aseaza in sir indian. Se cunosc numele celor  $n$  copii. Sa se gaseasca toate posibilitatile de aranjare in sir.

11. Se cer toate solutiile de asezare in linie a  $m$  caini si  $n$  pisici astfel incat sa nu existe o pisica intre doi caini

12. Sa se genereze toate numerele palindrome de lungime  $n$

16. Sa se decompuna un numar in suma de numere prime. Generati toate solutiile.

17.  $N$  copii se aseaza in cerc. Se cunosc numele celor  $n$  copii. Sa se gaseasca toate posibilitatile de rearanjare in cerc.

18. Se citește un număr. Sa se genereze toate numerele avand aceleasi cifre ca el. Care este cel mai mare?

19. Sa se genereze anagramele unui cuvânt.

20. Sa se genereze toate triunghiurile de perimetru  $n$ .

21. Sa se genereze toate numerele de lungime  $n$  formate doar cu cifre pare / impare

22. Scrieti un program care sa afiseze toate numerele de  $n$  ( $n \leq 10$ ) cifre, formate numai din cifre distincte si care sunt divizibile cu 4.

23. Fiind data o multime de  $n$  cuburi, fiecare cub fiind caracterizat de lungimea laturii si culoarea sa, sa se scrie un program care sa genereze toate turnurile care se pot forma cu  $p$  cuburi astfel incat doua cuburi vecine sa nu aiba aceleasi culoare iar deasupra unui cub sa nu se poata aseza un cub cu latura mai mare.

24. Un grup de copii are la dispozitie  $n$  cartonase cu  $n$  cuvinte distincte pentru jocul "cerc de cuvinte". In acest joc un copil trebuie sa spuna un cuvânt care sa aiba primele doua litere identice cu ultimele doua ale cuvântului spus de predecesorul lui. fiind dat un cuvânt de inceput pentru joc, afisati varianta cu cele mai multe cuvinte care se pot obtine cu ajutorul cartonaselor date. Observatie: un sir de cuvinte nu va contine un cuvânt de mai multe ori.

25. O persoana a uitat numarul de telefon al unui prieten. Stie doar ca numarul are 6 cifre, incepe cu 4 si contine 3 zerouri dintre care doua sunt alaturate. fisati toate variantele pe care trebuie sa le incerce pentru a vorbi cu prietenul sau.

26. La o masa rotunda sunt  $n$  persoane de diverse nationalitati, pentru fiecare persoana precizandu-se doua limbi straine cunoscute de ea. Se cere sa ajutati organizatorii mesei rotunde

## PROIECTAREA ALGORITMILOR

### **Laborator 9**

sa aranjeze persoanele astfel incat fiecare sa poata conversa atat cu cea din stanga cat si cu cea din dreapta.

27. Sa se genereze numerele mai mici decat  $n$  citit care trecute in baza 2 au in componenta lor exact  $p$  cifre de 1.

28. Sa se genereze toate secventele in cod binar de lungime  $n$ . Pentru fiecare secventa se va genera numarul asociat in baza 10. Sa se genereze toate codurile posibile.

29. Sa se genereze toate numerele naturale ale caror cifre se gasesc printre cifrele lui  $x$  citit si au lungimea cel mult egala cu lungimea lui  $x$ . Cifrele se pot repeta.

30. In cate moduri poate ajunge un pion de pe prima linie a unei table bidimensionale cu  $n$  linii si  $n$  coloane pe ultima linie a tablei. Se cunoaste coloana de plecare  $p$ . Pionul se poate deplasa numai pe o casuta alaturata si numai pe o linie mai mare.

31. Sa se determine partiile unui numar pt care suma inverselor obtinute este subunitara. Ex.  $n=5$   $3+2=5$  si  $1/3+1/2 < 1$ .

32. Se citeste un numar natural. Sa se determine toate numerele avand aceleasi cifre sau o parte din cifre si care sunt divizibile cu  $p$  citit

33. Sa se determine toate numerele cu cifre distincte. Cate astfel de numere sunt?

34. Sa se genereze toate numerele avand cifre distincte de la 0 la  $p$ . Numarul de cifre poate fi  $\geq 1$  si  $\leq p+1$ . Cate astfel de numere sunt?

## **Backtracking generalizat in plan**



1. Sa se completeze o tabla de 5\*5 cu saritura calului.  
Iata doua solutii:

Aflati numarul de solutii si afisati tablele.

2. Se da un tablou bidimensional (m linii, n coloane si m\*n componente) cu elemente distincte. Se dau si coordonatele primului punct. Ss se genereze un traseu in matrice pornind de la punctul dat astfel incat suma  $\sum k*a[i][j]$  (unde  $k=1 \rightarrow n*m$ ) sa fie maxima. In matrice se poate avansa numai la elemente vecine (pe cele 8 directii).

Exemplu: n=m=4;pozitia initiala : linia 3,coloana 2;

13	14	3	4
12	2	15	5
11	1	16	6
10	9	8	7

Suma maxima va fi:  $1*1+2*2+3*3+\dots+16*16$ .

3. Un labirint dreptunghiular cu m linii si n coloane contine culoare (reprezentate prin 0) si pereti (reprezentati prin 1). Se dau coordonatele initiale ale unui soricel si coordonatele finale , unde trebuie sa ajunga. Soricelul poate avansa sus, jos , stanga sau dreapta.(Incercati si cu cele 8 directii)

- a. Sa se genereze toate solutiile.
- b. Sa se afiseze souldtia cea mai lunga (scurta)

### **Backtracking in plan**

Fie urmatoare problema: un soricel se gaseste intr-un labirint de forma dreptunghiulara cu m linii si n coloane. Peretii sunt marcati cu 1 si culoarele cu 0. Se cunosc coordonatele initiale ale soricelului: Li, Ci. Sa se determine toate posibilitatile pe care le are soricelul pentru a iesi din labirint. Soricelul poate avansa pe 4 directii cate o celula (sus, dreapta , jos, stanga).

O astfel de problema presupune o abordare Backtracking in plan. Traseul soricelului va fi retinut de un vector cu doua campuri: coordonatele x si y. Vom defini un tip struct:

```
struct pozitie  
{int x,y;};
```

si vom declara un vector care retine drumul: pozitie d[50];

## PROIECTAREA ALGORITMILOR

### Laborator 9

Pentru generarea unui drum vom defini un subprogram recursiv `oid ies(int x,int y)` care primește ca parametri coordonatele unei componente din matrice. Inițial se apelează pentru coordonatele inițiale ale soricelului. O componentă din matrice va putea aparține drumului dacă evident este culoar ( $a[x][y]=0$ ). O celulă a matricii determinată ca aparținând drumului se marchează cu 2 (pentru a preveni ciclările):

```
a[x][y]=2;
```

Se încarcă valoarea corespunzătoare în vectorul `d` pe nivelul curent:

```
d[k].x=x;
```

```
d[k].y=y;
```

De fiecare dată când s-a determinat o nouă celulă care aparține drumului se determină dacă s-a ajuns la soluție (condiție care diferă de la o problemă la alta).

În cazul problemei noastre se iese din labirint când se ajunge la linia 0 sau coloana 0 sau linia  $m+1$  sau coloana  $n+1$ . Testul este:

```
if((x<1)||(x>m)||(y<1)||(y>n))
    tipar(k);
```

În caz afirmativ se tipărește (se afișează vectorul `d` și/sau matricea `a`) altfel (dacă soluția este incompletă) se încearcă parcurgerea, pe rând, a celor 4 celule alăturate. Acest lucru se realizează prin autoapelul funcției `ies` pe cele patru direcții:

```
ies(x-1,y);
    ies(x,y+1);
    ies(x+1,y);
    ies(x,y-1);
```

Observație: vectorul `d` funcționează ca o stivă cu două câmpuri.

La revenire din apel se eliberează celulă pentru a o putea accesa și în cadrul altor prelucrări:  $a[x][y]=0$  și se eliberează componenta drumului  $k=k-1$  (practic se coboară în stivă).

```
void ies(int x,int y)
{if(a[x][y]==0)
  {k++;
  a[x][y]=2;
  d[k].x=x;
  d[k].y=y;
  if((x<1)||(x>m)||(y<1)||(y>n))
    tipar(k);
  else
    {ies(x-1,y);
    ies(x,y+1);
    ies(x+1,y);
    ies(x,y-1);
    }
  a[x][y]=0; //la revenire din apel demarchez celula pentru a o putea
             //accesa si in cadrul altei prelucrari
  k--; //eliberez componenta din vectorul drumului
}
```

}

Fie urmatorul labirint: m=6 n=10 Li=4, Ci=3

```
1 1 1 1 0 1 1 1 1 1
1 1 1 1 0 1 1 1 1 1
1 1 1 1 0 1 1 1 1 1
1 1 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 0 1 1
1 1 1 1 1 1 1 0 1 1
```

Solutiile vor fi:

solutia 1

(4,3) (4,4) (4,5) (3,5) (2,5) (1,5) (0,5)

```
1 1 1 1 2 1 1 1 1 1
1 1 1 1 2 1 1 1 1 1
1 1 1 1 2 1 1 1 1 1
1 1 2 2 2 0 0 0 0 0
1 1 1 1 1 1 1 0 1 1
1 1 1 1 1 1 1 0 1 1
```

solutia 2

(4,3) (4,4) (4,5) (4,6) (4,7) (4,8) (4,9) (4,10) (4,11)

```
1 1 1 1 0 1 1 1 1 1
1 1 1 1 0 1 1 1 1 1
1 1 1 1 0 1 1 1 1 1
1 1 2 2 2 2 2 2 2 2
1 1 1 1 1 1 1 0 1 1
1 1 1 1 1 1 1 0 1 1
```

solutia 3

(4,3) (4,4) (4,5) (4,6) (4,7) (4,8) (5,8) (6,8) (7,8)

```
1 1 1 1 0 1 1 1 1 1
1 1 1 1 0 1 1 1 1 1
1 1 1 1 0 1 1 1 1 1
1 1 2 2 2 2 2 2 0 0
1 1 1 1 1 1 1 2 1 1
1 1 1 1 1 1 1 2 1 1
```

Programul complet este:

```
#include<fstream.h>

struct pozitie
    {int x,y;};

int a[20][20];//labirintul

int k,n,m,Li,Ci,nr_sol;

pozitie d[50];

void afis_mat() //afiseaza matricea
{cout<<endl;
for(int i=1;i<=m;i++)
    {for(int j=1;j<=n;j++)
        cout<<a[i][j]<<" ";
    cout<<endl;}
}

void tipar(int k) //tipareste vectorul drum
{nr_sol++;
cout<<"solutia "<<nr_sol<<endl;
for(int i=1;i<=k;i++)
    cout<<"("<<d[i].x<<','<<d[i].y<<") ";
afis_mat();
getch();
cout<<endl;}

void ies(int x,int y) //genereaza drumul
{if(a[x][y]==0)
    {k++;
    a[x][y]=2;
    d[k].x=x;
    d[k].y=y;
    if((x<1)||((x>m))||((y<1))||((y>n)))
        tipar(k);
    else
        {ies(x-1,y);
        ies(x,y+1);
        ies(x+1,y);
        ies(x,y-1);
        }
    a[x][y]=0; //la revenire din apel demarchez celula pentru a o putea
    //accesa si in cadrul altei prelucrari
    k--;//eliberez componenta din vectorul drumului
    }
}

void citire()
{ fstream f;
f.open("labir.txt",ios::in);
if(f)
    cout<<"ok";
else
    cout<<"eroare la deschidere";
```

```
//Citesc matricea ce reprezinta labirintul
f>>m>>n;

for(int i=1;i<=m;i++)
  for(int j=1;j<=n;j++)
    f>>a[i][j];

f>>Li>>Ci; //coordonatele punctului in care se afla soricelul
}

int main()
{
    k=0;
    citire();
    ies(Li,Ci);
    afis_mat();
}
```

**Probleme propuse spre rezolvare**

1. Aceeasi problema ca in exemplu cu diferenta ca soricelul poate avansa in celule alaturate pe cele 8 directii posibile
2. Un soricel se gaseste intr-un labirint de forma dreptunghiulara cu m linii si n coloane. Peretii sunt marcati cu 1 si culoarele cu 0. Se cunosc coordonatele initiale ale soricelului: Li, Ci. Sa se determine toate posibilitatile pe care le are soricelul pentru a iesi ajunge la cascaval. Se cunosc coordonatele cascavalului: Lf,Cf.
3. Aceeasi problema ca la 2 cu diferenta ca se afiseaza doar cea mai scurta solutie.
4. Un labirint dreptunghiular cu m linii si n coloane contine culoare (reprezentate prin 0) si pereti (reprezentati prin -1). Se dau coordonatele initiale ale unui soricel si coordonatele finale , unde trebuie sa ajunga. Pe culoare se gasesc bucati de branza pt care se cunoaste greutatea in grame.
  - a) Sa se genereze toate solutiile., pt fiecare se afiseaza cantitatea consumata
  - b) Sa se afiseze solutia cea mai „consistenta” Indicatie: se vor marca cu -2 celulele parcurse si se vor retine in vectorul drum inclusiv cantitatea consumata. Matricea se borda cu pereti (-1).
5. Pe o tabla de forma dreptunghiulara se afla un cal. Se cunosc coordonatele initiale ale calului. Acesta trebuie sa ajunga intr-o pozitie finala sarind numai sub forma sariturii calului. Stiind ca numai anumite celule sunt permise sariturii (acestea sunt marcate) sa se determine ce posibilitati sunt ca sa se ajunga in pozitia finala.
6. Romeo si Julieta se gasesc intr-un labirint (se cunosc culoarele si peretii si coordonatele celor doi indragostiti).
  - a) Exista posibilitatea ca Romeo sa ajunga la Julieta?
  - b) in cazul in care cei doi se indreapta simultan unul catre celalalt pentru fiecare solutie se va afisa locul intalnirii (coordonatele celulelor alaturate sau celulei comune de intalnire)