Metoda Backtracking

Se aplică pentru problemele în care soluţia poate fi reprezentată sub forma unui vector **X=(x[1],x[2],...,x[k])**.

X∈S[1]xS[2]x...xS[n]

S[1]xS[2]x...xS[n] - se numeşte spaţiul soluţiilor posibile.

|S[i]|=s[i], S[i]={a[i,1],a[i,2],...,a[i,s[i]]}

Între componentele vectorului X sunt date mai multe **condiţii interne**, care depind de natura problemei date. **Soluţiile posibile** pentru care sunt verificate condiţiile interne le numim **soluţii rezultat**. În unele probleme se cere alegerea dintre soluţiile rezultat a **soluţiei optime** după un anumit criteriu.

Observaţii:

Astfel de probleme pot fi rezolvate prin generarea tuturor elementelor produsului cartezian S[1]xS[2]x...xS[n] (întreg spaţiul soluţiilor), dar această metodă nu este eficientă. Această metodă se numeşte **metoda forţei brute – brute-force.**

Soluţie parţială = o soluţie pentru care sunt indeplinite condiţiile de continuare. Condiţiile de continuare sau de validare se obţin din condiţiile interne aplicate pentru o soluţie parţială.

Construirea soluţiei:

* Soluţia se construieşte sub forma unei stive, astfel încât la fiecare pas se alege o valoare pentru elementul curent x[k] (soluţia se construieşte progresiv). La început k=1 şi x[1] se iniţializează ***cu o valoare anterioară primei valori*** din S[1].

* Cât timp stiva nu este vidă, completăm soluţia parţială cu noi componente. Elementului x[k] din vârful stivei i se atribuie următoarea valoare din mulţimea S[k] (dacă există).
  Pentru soluţia parţială obţinută **(x[1],x[2],...,x[k])** sunt verificate ***condiţiile de validare***. Apar astfel două posibilităţi:

1) Condiţia de continuare este **adevărată**:
   - în acest caz testăm egalitatea **k==n**, adică verificăm dacă avem valori pentru toate componentele vectorului soluţie. În caz afirmativ, am obţinut o soluţie a problemei, pe care o reţinem.
   - În cazul **k<n** incrementăm k cu 1 şi iniţializăm x[k] cu o valoare anterioară primei valori din mulţimea S[k].

2) În situaţia în care condiţia de continuare este **falsă**, alegem pentru x[k] următoarea valoare posibilă din S[k] şi verificăm din nou condiţiile de validare. Dacă nu se poate face altă alegere, revenim pe stivă la nivelul anterior şi verificăm următoarea valoare pentru x[k-1].

Pentru orice problemă particulară care este rezolvată prin metoda backtracking trebuie precizate:

(1) **spaţiul soluţiilor**, adică mulţimile `S[1],S[2],...,S[n]`.
Nu este necesar ca aceste mulţimi să aibă acelaşi număr de elemente.

(2) **condiţiile de validare** a unei soluţii parţiale **`(x[1],x[2],...,x[k])`**.
Alegerea acestor condiţii influenţează **decisiv eficienţa metodei**.

Procedura generală backtracking iterativă

```
1.   int x[31],n;
2.
3.  void back()
4.  {
5.   int k=1;
6.   x[k]=init(k);
7.   while (k>0)
8.   {
9.    while (există o valoare netestată în Sk)
10.   {
11.    x[k]=urmator(k);
12.    if valid(x[1],x[2],...,x[k])
13.    if (k==n) solutie(x[1],x[2],...,x[n]);
14.    else {
15.          k=k+1;
16.          x[k]=init(k);
17.         }
18.   }
19.   k=k-1;
20.  }
21. }
```

Procedura generală backtracking recursivă

```
1.   int x[31],n;
2.
3.  void back(int k)
4.  {
5.   x[k]:=init(k);
6.   while (are_succesor(x[k])
7.     if valid(k)
8.       if (este_solutie(k)) solutie();
9.       else back(k+1);
10. }
```

* O variantă recursivă în cazul în care toate mulţimile `S[k]` sunt egale, `S[k]={1,2,…,n} k=`$\overline{1,n}$ este următoarea:

```
1.   void back(int k)
2.  {
3.   int i;
4.   for(i=1;i<=n;i++)
5.   {
6.    x[k]=i;
7.    if (valid(k))
8.      if (este_solutie(k)) solutie();
9.      else back(k+1);
10.  }
11. }
```

Probleme propuse:

1. Generarea produsului cartezian a n mulţimi.

2. Generarea permutărilor de n elemente.

3. Problema damelor:
Să se afişeze toate posibilităţile de aşezare a 8 (n) regine pe o tablă de şah 8x8 (nxn) în aşa fel încât să nu se atace reciproc.

4. Generarea combinărilor de n elemente luate câte k.

5. Generarea funcţiilor injective (aranjamente)
Să se afişeze toate funcţiile injective f:A $\rightarrow$ B, unde A şi B sunt două mulţimi cu m respectiv n elemente.

6. Să se afişeze toate partiţiile unui număr natural nenul n.
Prin partiţie a lui n se înţelege o descompunere a lui n ca sumă de numere naturale nenule.

7. Plata unei sume de bani cu bancnote de valori date
Să se afişeze toate modalităţile de a plăti o sumă de bani S cu bancnote de valori b1>b2>...>bn. Să se determine modalităţile de plată cu număr minim de bancnote.

8. Colorarea hărţilor
S-a demonstrat că orice hartă plană poate fi colorată cu maximum 4 culori. Pentru o hartă plană dată să se afişeze toate modalităţile de colorare a hărţii cu maxim 4 culori.

Indicaţie: Relaţia de vecinătate a două ţări de pe hartă se păstreaza sub forma unei matrici pătratice T, ale cărei elemente au valorile 0 sau 1:

$$T_{i,j} = \begin{cases} 0, \textit{ dacă ţara } t_i \textit{ nu este vecină cu ţara } t_j \\ 1, \textit{ţara } t_i \textit{ nu este vecină cu ţara } t_j \end{cases}$$

Observaţii:
i) Matricea T este o matrice pătratică.
ii) Pe diagonala principală elementele au valoarea 0 (ţara t_i nu este vecină cu ea insăşi).
iii) Matricea T este simetrică faţă de diagonala principală.

9. Să se descompună numărul natural n în toate modurile posibile ca sumă de exact p numere naturale nenule (p<=n).

10. Să se descompună numărul natural n în toate modurile posibile ca sumă de numere prime.

11. Problema determinării unui subşir crescător maximal.
Se dă un şir de numere întregi. Să se determine un subşir crescător de lungime maximă.

12. Problema comis-voiajorului

Un comis-voiajor trebuie să viziteze n oraşe. Iniţial, porneşte din oraşul 1, parcurge toate oraşele fără să treacă de două ori prin acelaşi oraş astfel încât la întoarcere să ajungă înapoi în orasul 1.
Cunoscând drumurile de lagătură existente între oraşe să se tiparească toate rutele posibile pe care le poate urma comis-voiajorul.

$$A_{i,j} = \begin{cases} 0, \textit{dacă i nu este vecin cu } j \\ 1, \textit{dacă i este vecin cu } j \end{cases}$$

Backtracking în plan

13. Problema labirintului

Se dă un labirint sub forma unei matrici de tip mxn cu elemente 0 (liber) şi 1 (zid). Să se afle toate drumurile de la un punct de intrare in labirint (is,js) la un punct de iesire din labirint (ie,je), aflat la marginea labirintului - pe prima sau ultima linie/coloană.

14. Problema bilei

Un teren este reprezentat sub forma unei matrici T de tip mxn.
Elementele T[i,j] ale matricii reprezintă cotele unor porţiuni din teren.
O bilă se găseşte pe o anumită porţiune (ib,jb) cu o anumită cotă.
Să se determine toate traseele posibile ale bilei pentru a ieşi de pe teren, ştiind că bila se poate deplasa pe orice porţiune de teren învecinată cu o cotă strict mai mică decât cota la care se află bila.

15. Problema calului

Pe o tablă de şah se aşează un cal într-o căsuţă a tablei. Să se precizeze toate drumurile pe care le poate urma calul prin săritură astfel încât să fie parcurse toate căsuţele tablei.