Laborator 7:
tehnica backtracking.căutarea în şiruri

Întocmit de: Claudia Pârloagă

Îndrumător: Asist. Drd. Gabriel Danciu

I. NOŢIUNI TEORETICE

A. Prezentarea tehnicii BACKTRACKING

Tehnica se foloseşte în rezolvarea problemelor care îndeplinesc simultan următoarele condiţii:

- soluţia problemei poate fi pusă sub forma unui vector $S = x_1, x_2, ..., x_n cu x_1 \epsilon A_1, x_2 \epsilon A_2, ..., x_n \epsilon A_n$;
- multimile $A_1, A_2, ..., A_n$ sunt mulţimi finite, iar elementele lor se consideră a se afla într-o relaţie de ordine bine stabilită;
- multimile $A_1, A_2, ..., A_n$ pot coincide;
- $x_1, x_2, ..., x_n$ pot fi la rândul lor vectori.

Produsul cartezian $A_1 x A_2 x, ..., x A_n$ se numeşte **spaţiul soluţiilor** iar un element $x_1, x_2, ..., x_n$ din spaţiul soluţiilor este soluţie a problemei daca îndeplineşte anumite condiţii cerute de problemă.
Tehnica **Backtracking** are la bază următorul principiu:

- se construieşte soluţia pas cu pas: $x_1, x_2, ..., x_n$
- dacă se constată că, pentru o valoare aleasă, nu avem cum să ajungem la soluţie, se renunţă la acea valoare(se face un pas înapoi) şi se reia căutarea din punctul în care am rămas.

CONCRET:

- se alege primul element $x_1 \epsilon A_1$;
- aleg apoi, pe rând, din fiecare mulţime A_i un element x_i(după o anumită ordine bine stabilită) şi construiesc soluţii parţiale; la fiecare alegere, testez dacă soluţia parţială la care s-a ajuns poate conduce la o soluţie finală.
- fie generată secvenţa $x_1, x_2, ..., x_k$ la un moment dat $k(k \leq n)$. Dacă se constată că această secvenţă nu conduce la o soluţie(adică alegând următorul x_{k+1} nu ajung la soluţie) atunci renunţ la valoarea $x_k \epsilon A_k$(fac un pas înapoi până la x_{k-1}) şi caut următorul $x_k \epsilon A_k$(care nu a fost încă testat) cu care să construiesc soluţia parţială $x_1, x_2, ..., x_k$

De fapt, vectorul soluţie funcţionează ca o stivă: la fiecare moment se procesează elementul din vârf, adică adaug un nou element în vârf sau extrag un element din vârf dacă acesta nu îndeplineşte condiţiile cerute.

PAŞI:

- se alege primul element $x_1 \epsilon A_1$
- la pasul k, presupunem că au fost generate elementele $x_1, x_2, ..., x_k$, $x_i \epsilon A_i$, ce reprezintă o soluţie parţială care poate conduce la o soluţie a problemei.
- la pasul k+1: verific dacă am ajuns la soluţie
  - dacă am soluţie, atunci aceasta se tipăreşte, apoi se reia algoritmul de la pasul k cu soluţia parţială $x_1, x_2, ..., x_{k-1}$ generată şi aleg rumătorul element $x_k \epsilon A_k$
  - altfel:
    - se alege primul element disponibil $x_{k+1} \epsilon A_{k+1}$ (deci care nu a fost încă testat pentru construcţia unei soluţii pornind de la soluţia parţială $x_1, x_2, ..., x_k$):

- dacă nu s-a găsit un astfel de element(am epuizat toate elementele mulţimii A_{k+1} sau orice element aleg nu obţin o soluţie parţială validă), atunci renunţ la x_k şi reiau căutarea de la pasul $k-1$, deci consider generată soluţia parţială $x_1, x_2, ..., x_{k-1}$ şi vreau să generez soluţia parţială $x_1, x_2, ..., x_k$
- dacă am găsit primul element disponibil $x_{k+1} \epsilon A_{k+1}$ atunci testez dacă se verifică condiţiile de continuare(verific dacă $x_1, x_2, ..., x_{k+1}$ poate conduce la o soluţie a problemei):
  1. dacă sunt îndeplinite condiţiile de continuare, trec la pasul $k+2$, similar cu pasul $k+1$
  2. dacă nu sunt îndeplinite condiţiile de continuare, renunţ la $x_{k+1} \epsilon A_{k+1}$ şi reiau pasul $k+1$, căutând următorul $x_{k+1} \epsilon A_{k+1}$ disponibil.

Algoritmul se termină atunci când, am testat toate elementele şi nu mai am nici un element netestat.

B. Generarea permutărilor

Fie o mulţime cu n elemente $A = x_1, x_2, \ldots, x_n$. Să se genereze toate permutările mulţimii A.
Folosim, pentru generare, mulţimea $1, 2, ..., n$. Condiţiile care se impun sunt următoarele:

- fiecare element din vectorul soluţie se alege din mulţimea $1, 2, ..., n$;
- un element x_k este valid dacă el nu a mai fost adăugat anterior în vectorul soluţie.

Modalităţi prin care se poate verifica dacă elementul x_k a fost plasat deja in vectorul soluţie:

- parcurgerea elementelor deja generate pentru a verifica dacă x_k apare sau nu printre ele;
- folosirea unui vector cu n elemente în care vom avea valori 0 sau 1 corespunzătoare elementelor mulţimii iniţiale. Valoarea 1 va preciza faptul că elementul de pe poziţia corespunzătoare a fost plasat anterior în vectorul soluţie, iar valoarea 0 că nu.

Exemplu: pentru n=3 se obţin 6 soluţii:
1 2 3; 1 3 2; 2 1 3; 2 3 1; 3 1 2; 3 2 1.

C. Problema damelor pe tabla de şah

Fiind dată o tablă se şah de dimensiunea $n \times n, n \geq 3$, se cer toate soluţiile de aranjare a n dame(regine) astfel încât să nu se afle două dame pe aceeşi linie, coloană sau diagonală(damele să nu se atace reciproc).

Exemplu: presupunem ca avem o tablă de şah 4×4. O soluţie ar fi următoarea:

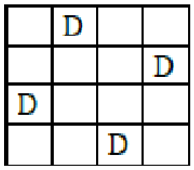

Figura 1:

Algoritm:

```
Dame(x, n)
pentru i=1,n    //intai initializam vectorul solutiilor cu 0, pentru ca initial nu avem nici
     x[i]=0     //o dama plasata
 sfarsit pentru
 k=1
 cât timp k>0
   daca k= n+1 atunci
       scrie(x[i], i=1,n)
       k = k-1
   altfel
       daca x[k]<n atunci
          x[k] = x[k]+1
          daca valid(x,k)=1 atunci
             k=k+1
          sfarsit daca
       altfel
          x[k]=0
          k=k-1
       sfarsit daca
   sfarsit daca
sfarsit cat timp
Return
```

Funcţia care testează îndeplinirea condiţiilor de continuare:

```
valid(x,k)
  valid=1
  pentru i=1,k-1
   daca x[i] = x[k] atunci //doua dame sunt pe aceeasi coloana
       valid = 0
   sfarsit daca

daca |x[i] - x[k]| = k - i atunci //doua dame sunt pe aceeasi
 //diagonala
       valid=0
   sfarsit daca
  sfarsit pentru
  Return
```

II. TEMĂ

Pentru fiecare din următoarele probleme se va scrie pseudocodul şi apoi codul în Java corespunzător.

Problema 1:

Cum aţi extinde metoda Rabin-Karp pentru problema căutării într-un text a unui set de k şabloane? Exemplu:
textul: abaacadaeedfadeadbcababacadafbcafceab
set-ul de şabloane: {ade, cea, cada, adbc}

Problema 2:

Fie un labirint reprezentat prin matricea următoare: $\begin{pmatrix} \mathbf{0} & 0 & 0 & 0 & 1 & 0 \\ \mathbf{0} & 1 & 1 & 0 & 0 & 1 \\ 1 & \mathbf{0} & 1 & 1 & 1 & 0 \\ 0 & \mathbf{0} & \mathbf{0} & 1 & 1 & 1 \\ 1 & 1 & 1 & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix}$

Intrarea în labirint este marcată cu albastru(stânga sus) iar ieşirea din labirint este marcată cu roşu(dreapta jos). Drumul unic de la intrare la ieşire este marcat cu negru.
Pentru a ajunge de la intrare la ieşire condiţia este: pe 0 se poate trece pe 1 nu.
Să se scrie un algoritm ce respectă tehnica backtracking şi determină acest drum unic de ieşire din labirint.