## Problema 3 – towerx

*prof. Adrian Panaete*
*C.N. "A. T. Laurian" Botoșani*

Este aproape evident ca la final șirul care se obține va fi o permutare a numerelor 1, 2, ..., N. În plus dacă avem pozitia în care apare N singurul lucru pe care îl vom avea de facut este sa copiem în spatele acelei poziții soluția pentru N-1 ( evident ca la terminarea celor N poziții copierea va continua pornind de la poziția 1) . Acest lucru se explica astfel : cand punem valoarea N se șterg toate valorile 1, 2, ..., N-1 deci vectorul va avea doar pozitia ocupata de N nenula. Continuăm parcurgerea din poziția imediat urmatoare acelei pozitii din care practic nu avem altceva de facut decât să repetăm pașii care rezolvă problema pentru N-1 cu singura diferența că acum pozitiile sunt rotite ca și cum pozitia respectiva ar fi poziția 1.

Va trebui deci sa avem solutia și pentru k=N-1 si inductiv pentru k=N-2,...,k=1.

Notam P[k][j]=poziția în care apare pentru prima data valoarea j atunci cand rezolvăm problema pentru k. Putem determina această poziție folosind metoda programării dinamice:

Evident P[k][1]=1 pentru orice k.

Fie P[k][j-1] poziția in care apare j-1 pentru prima data. E evident ca pana la aparitia acelui j-1 nu avea cum să apară j.

La momentul primei aparitii a lui j-1, vectorul de lungime k va ramâne nul exceptând pozitia P[k][j-1]. E ca și cum am reîncepe rezolvarea pentru k-1 asteptând acum prima apariție a lui j-1 și plecând cu poziția ce urmeaza lui P[k][j-1] în loc de pozitia 1. Doar ca în pozitia în care ar trebui sa apara j-1 va apare j pentru ca deja îl avem pe j-1.

Tragem concluzia ca P[k][j]=P[k][j-1]+P[k-1][j-1] din care se scade k dacă suma este strict mai mare decât k (optimizare 1– se evită operația modulo).

Deoarece k apare doar o data în vectorul de lungime k deducem că P[k][k] va fi exact ce cautăm adică prima ( și ultima ) apariție a lui k în permutarea de k.

Dinamica aplicată are complexitatea O($N^2$).

Având acum la dispoziție P[k][k] = poziția lui k în permutarea de lungime k (pentru orice k de la 1 la N) se poate genera soluția tot în complexitate O($N^2$) pentru N simulând așezarea valorilor k= N, N-1,N-2, ....1 pe pozițiile rămase libere . Deși complexitatea teoretică rămâne O($N^2$) la acest pas, pentru a accelera procesul de poziționare a valorilor în permutarea soluție se poate simula o coada circulara din care să eliminăm pozițiile imediat ce acestea au fost deja ocupate de o valoare(optimizare 2).

Complexitatea finală a soluției este O($N^2$) si se încadrează în timpul de execuție făra a utiliza cele două optimizări sugerate pe parcursul descrierii.

În ceea ce privește complexitatea de memorie se observă la dinamică faptul că sunt necesare doar linia anterioară unei linii din matricea P[][] pentru a genera linia curentă. Pentru a ne încadra în restricțiile de memorie aplicăm un truc relativ clasic în problemele de programare dinamică -simulăm doar două linii din matrice (cu doi vectori de lungime N). Deci pentru a obține punctaj maxim trebuie să folosim complexitate de memorie O(N).