maşina – Soluţie

Problema este de programare dinamică. Există mai multe feluri de rezolvare, iar acesta este unul dintre ele:

Asemănăm acel drum din mijloc cu o stiva, iar operaţiile sunt paranteze deschise (atunci când vine o maşina în stivă) si paranteze închise (când pleacă o maşină din stivă).

Construim:
`A[i][j]` = numărul de posibilităţi de a ajunge în poziţia cu i paranteze deschise si j închise
`B[i][j]` = numărul de posibilităţi de a încheia situaţia (N deshise, N închise) din poziţia (i, j), adică i deschise, j închise

Presupunem X < Y, altfel inversăm si scădem la sfârşit.
Variem momentul în care apare X în stivă (baza), şi anume poziţia pe care se găseşte în stivă când apare.

Construim:
`Q[i][j]` = numărul de posibilităţi de procesare a i maşini având j în stivă, ţinem i între x şi y şi j între baza şi baza + diferenţa între x şi y

Momentul în care se ia o decizie (dacă x e în faţa lui y sau invers), este atunci când stiva e la nivelul baza sau când urmează să procesăm pe y. Adunăm aceste momente, înmulţindu-le cu B-ul corespunzător (adică câte posibilităţi sunt până la sfârşit) şi cu A-ul corespunzător (câte posibilităţi să ajungem la baza).

Întrucât precizia este doar de două zecimale, putem folosi tipul double pentru a calcula toate posibilităţile.

matrice – Soluţie

Ideea este de a permuta rândurile matricii astfel încât să respecte proprietatea cerută şi anume aceea ca oricare coloană să nu conţină un subşir strict crescător de lungime mai mare decât $\lceil \sqrt{N} \rceil$ şi de a scrie permutarea respectivă folosind operaţiile permise.

Permutarea cerută se obţine astfel: mai întâi se sortează rândurile matricei descrescător după valorile primei coloane. Apoi pentru grupe de câte $\lceil \sqrt{N} \rceil$ rânduri, se sortează descrescător după valorile celei de-a doua coloane. Cel mai lung subşir crescător în prima coloană poate fi găsit numai în cadrul grupelor şi deci are lungimea maxim $\lceil \sqrt{N} \rceil$. În a doua coloană, cel mai lung subşir crescător poate fi obţinut alegând cât mai convenabil un element din fiecare grupă (cele din cadrul grupelor sunt sortate descrescător) şi deci are lungimea maximă posibilă egală cu numărul grupelor, adică $\lfloor \sqrt{N} \rfloor$.

Pentru a obţine o permutare cu ajutorul operaţiilor S_3, J_3, S_4, J_4, observăm că orice transpoziţie se poate scrie în funcţie de aceste operaţii. Orice permutare se poate descompune în cicluri (suma lungimilor lor fiind egală cu N). Cum orice ciclu de lungime L se poate descompune în L transpoziţii şi (vom demonstra că) orice transpoziţie se poate obţine prin efectuarea a 2 dintre operaţiile permise, rezultă că orice permutare se poate obţine prin efectuarea a cel mult $2*N$ operaţii permise.

Pentru a arăta că orice transpoziţie se poate obţine din operaţiile S_3, J_3, S_4, J_4, considerăm cazurile

$$\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \xrightarrow{S_4} \begin{bmatrix} b \\ c \\ d \\ a \end{bmatrix} \xrightarrow{J_3 \text{ (pentru ultimele 3)}} \begin{bmatrix} b \\ a \\ c \\ d \end{bmatrix} \Rightarrow$$ putem interschimba rândul a cu b dacă după ele mai sunt cel puţin 2 rânduri

$$\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \xrightarrow{J_4} \begin{bmatrix} d \\ a \\ b \\ c \end{bmatrix} \xrightarrow{S_3 \text{ (pentru primele 3)}} \begin{bmatrix} a \\ b \\ d \\ c \end{bmatrix} \Rightarrow$$ putem interschimba rândul c cu d dacă înaintea lor mai sunt cel puţin 2 rânduri

$$\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \xrightarrow{S_4} \begin{bmatrix} b \\ c \\ d \\ a \end{bmatrix} \xrightarrow{J_3 \text{ (pentru primele 3)}} \begin{bmatrix} d \\ b \\ c \\ a \end{bmatrix} \Rightarrow$$ putem interschimba rândul a cu d dacă între ele mai sunt cel puţin 2 rânduri

Pentru două rânduri a şi b, cel puţin unul dintre cazuri este adevărat.

ziduri – Soluţie

Soluţia 1

Se considera graful format astfel: pentru fiecare camera se considera un nod, iar intre doua camere vecine pe orizontala/verticala (deci doua noduri) o muchie de cost 1 daca nu exista zid intre cele doua camere, respectiv p daca exista zid. Pe acest graf se calculeaza cu algoritmul lui Dijkstra costurile minime de la camera stanga-sus la toate celelalte camere. Apoi se construieste graful orientat aciclic al tuturor drumurilor minime de la camera stanga sus la camera dreapta jos. Pe acest graf aciclic se calculeaza cu o parcurgere drumul dintre cele doua camere care foloseste numar minim de muchii formate din ziduri. Pentru o complexitate mica, algoritmul lui Dijkstra trebuie implementat cu heapuri. Astfel, complexitatea totala e de N^2logN.

Soluţia 2

Pe acelasi graf, putem afla aceasta distanta folosind algoritmul Bellman – Ford – Moore. Desi acesta poate conduce la o complexitate de O(N^4), ne putem baza pe faptul ca p este destul de mic (mai mic decat 100 (adica maximul lui N)) si acesta reduce la O(N^3) complexitatea, cu care putem obtine punctaj maxim.