Descriere solutie – problema rooks

autor: Mihail-Cosmin Pit-Rada

Solutia 1 – 100 puncte - ***Mihail-Cosmin Pit-Rada***

0) Se observa ca permutand 2 linii sau 2 coloane problema nu isi schimba generalitatea si admite aceleasi solutii, cu diferenta ca liniile/coloanele alese vor fi permutate. Astfel vom putea permuta liniile si coloanele convenabil incat, restrictiile vor ajunge in coltul stanga sus intr-o regiune rectangulara M1xN1 cu M1, N1 <= 17.
(1) Calculam numarul de configuratii posibile pentru zona M1xN1.
Sa presupunem ca am completat primele i linii cu r turnuri. Pentru a completa linia urmatoare este util sa cunoastem in ce coloane am amplasat cele r turnuri. Intrucat nu putem avea mai mult de 17 turnuri, putem codifica aceste coloane folosind intregi pe 32 de biti, pozitiile bitilor setati reprezentand coloanele interzise, in care au fost amplasate ce r turnuri. Astfel putem introduce urmatoarea definitie:

ways[i][rooks] = numarul de amplasari a r turnuri, in primele i linii, coloanele acestor r turnuri fiind reprezentate de bitii de 1 din rooks. De observat ca numarul r se poate deduce numarand bitii de 1 din rooks.

Detalii tehnice:
(*) pentru fiecare linie 0 <= i < M1 se pot codifica intr-un intreg pozitiile libere in care se pot amplasa turnuri
(*) folosind operatii de tipul & se pot obtine pozitiile candidat pentru linia ce urmeaza a fi analizata.
(*) in linia curenta se poate decide sa adaugam sau nu un turn.
(*) se pot calcula usor tranzitiile de la ways[i][rooks] la ways[i+1][newRooks] analizand pozitiile/bitii candidat
(*) se observa ca se poate minimiza consumul de memorie folosind doar 2 linii pe care le interschimbam
(*) pentru o implementare eficienta se observa ca in primele i linii nu putem avea mai mult de i turnuri, prin urmare se vor lua in calcul doar acele configuratii rooks cu cel mult i biti.
(*) complexitate:
(a) in primele i linii nu putem folosi mai mult de i turnuri
(b) pentru r turnuri avem cel mult Comb(N1, r) amplasamente posibile in rooks
(c) pentru r turnuri deja amplasate avem cel mult N1 - r candidati
(d) pentru linia i avem complexitatea: sum{ Comb(N1, r) * (N1 - r) | 0 <= r <= i }
(e) luand in considerare toate liniile pana la M1 avem:
sum{ Comb(N1, r) * (N1 - r)*(N1 - r) | 0 <= r < M1 } ~ O(M1 * N1 * 2^N1)
constanta din spatele notatiei O() fiind aproximativ 1/4

(2) Din ways[M1][*] putem deduce usor:
waysO[r] = numarul de amplasamente valide a r turnuri in zona M1xN1
waysO[r] = sum{ ways[M1][rooks] | rooks contine r biti setati }
Complexitate: O(2^N1)

(3) Calculam:

Comb[i][r]*Aranj[j][r]= numarul de configuratii cu r turnuri, intr-o regiune ixj fara restrictii de amplasare, adica fara #

Practic trebuie alese r linii si r coloane, ce genereaza rxr intersectii, in care se pot amplasa turnurile in r! moduri, de unde si formula de mai sus.
Complexitate: O(M * N)

(4) Putem extinde zona M1xN1 orizontal, spre dreapta, pentru a calcula pentru zona M1xN:
waysH[r] = numarul de amplasamante cu r turnuri in zona M1xN
waysH[r] = sum{ waysO[r1] * Comb[M1 - r1][r2] * Aranj[N - N1][r2] | r1 + r2 = r }
Recurenta se bazeaza pe observatia ca daca in zona originala M1xN1 avem r1 turnuri atunci r1 linii sunt ocupate de aceste turnuri, raman disponibile M1 - r1 linii, si mai trebuie amplasate r2 turnuri doar in partea dreapta, in cele N - N1 coloane. Aceste turnuri nu au restrictii de amplasare, deci se poate folosi (3).
Complexitate: O(M1 * N)

(5) In mod similar se extinde zona M1xN vertical, in jos, la zona MxN:
waysH[r] = numarul de amplasamante cu r turnuri in zona MxN
Recurenta este similara.
Complexitate: O(M * N)

(6) Rezultatul final va fi: waysH[0] + waysH[1] + ... + waysH[min(M, N)]

Soluția 2 – 100 puncte – ***prof. Ionel-Vasile Piț-Rada, Colegiul Național Traian, Drobeta Turnu Severin***

Se interschimbă liniile și coloanele astfel încât caracterele '#' să apară doar în zona 0<=i<L și 0<=j<K și aici să nu existe linie sau coloană formate doar cu caracterul '.' Determinăm, pentru fiecare 1<=r<=rmax , f(r) = numărul de așezări posibile pentru r turnuri. Asta se poate face cu operații pe biti. Apoi pentru fiecare valoare 0<=p<=min(M-L,N-K) se determină g1(p) numărul de așezări pentru p turnuri în zona L<=i<M și K<=j<N. Pentru fiecare r turnuri așezate în zona NV și p turnuri în zona SE sunt eliminate din "joc" în zonele SV și respectiv NE anumite linii și coloane , iar pentru liniile și coloanele rămase neatacate se poate calcula asemănător cu g1 (prin calcule cu formule combinatoriale) numărul de configurații posibile. Pentru cele trei zone care nu conțin '#' se poate precalcula numărul de configurații posibile pentru orice număr de turnuri așezate. Cu alte cuvinte vor trebui însumate toate produsele f(r1)*g(r2)*g(r3)*g(r4). Deoarece r1+r2+r3+r4<=N complexitatea calculării acestei sume este O((L^2)*K*N), iar cea de precalculare a tuturor valorilor f() este O(L*K*(2^K)) .
Complexitatea finală este O(L*K*(2^K)+ (L^2)*K*N)

Soluția 3 - 100 puncte – ***prof. Marius Nicoli , Colegiul Național „Frații Buzești", Craiova***

Se rearanjează matricea dată astfel încât caracterele # să fie grupate în primele M1 linii și N1 coloane. Se calculează apoi cu programare dinamică pentru fiecare număr r de turnuri posibil numărul de moduri de a fi așezate în această zonă. Se precalculează valorile pentru combinări și aranjamente și pentru numărul de AS[i][j] de aranjări ale turnurilor într-o zonă liberă de dimensiuni i x j.
AS[i][j]=AS[i-1][j-1]*j+AS[i-1][j], AS[0][j]=AS[i][0]=1
Complexitatea devine astfel O(p*p*2^p+M*N), unde p=min(M1,N1)