

Soluție borg

Numărul total de moduri în care se pot plasa K motoare în paralelipiped, fără restricția ca fiecare plan să conțină cel puțin un motor, este C_{N*M*H}^K . Din acestea, trebuie să le scădem pe cele care nu sunt bune.

Dacă notăm cu x_i ($1 \leq i \leq N$) numărul de moduri în care se pot plasa cele K motoare astfel încât planul i (de tip 1) să fie gol, cu y_i ($1 \leq i \leq M$) numărul de moduri astfel încât planul i (de tip 2) să fie gol și cu z_i ($1 \leq i \leq H$) numărul de moduri astfel încât planul i (de tip 3) să fie gol, atunci numărul căutat va fi $C_{N*M*H}^K - \left| \bigcup_{1 \leq i \leq N} x_i \cup \bigcup_{1 \leq i \leq M} y_i \cup \bigcup_{1 \leq i \leq H} z_i \right|$. Cardinalul reuniunii se poate dezvolta cu principiul includerii și excluderii, fiecare mulțime rezultată conținând o intersecție de a mulțimi X , b mulțimi Y și c mulțimi Z . Cardinalul unei astfel de mulțimi ar reprezenta numărul de moduri în care se pot pune K motoare, astfel încât a dintre planele de tip 1 să fie goale, b dintre planele de tip 2 să fie goale și c dintre planele de tip 3 să fie goale, adică ar fi $C_{(N-a)*(M-b)*(H-c)}^K$. În dezvoltarea produsă de principiul includerii și excluderii vor fi $C_N^a * C_M^b * C_H^c$ astfel de mulțimi.

Așadar numărul total va fi
$$\sum_{a=1}^N \sum_{b=1}^M \sum_{c=1}^H (-1)^{a+b+c} * C_N^a * C_M^b * C_H^c * C_{(N-a)*(M-b)*(H-c)}^K .$$

Se observă că este nevoie de toate combinările până la maximum dintre N , M și H , dar numai de cele care au K sus pentru restul. Acestea se pot calcula în complexitatea $O(N*M*H*K)$ folosind regula generală

$C_n^k = C_{n-1}^{k-1} + C_{n-1}^k$ ($k \leq K$ și $n \leq N*M*H$). Pentru a reduce complexitatea la $O(N*M*H)$, putem calcula C_n^K direct din C_{n-1}^K , folosindu-ne de faptul că numărul 30103 este prim și deci orice număr are un invers modular (care înmulțit cu el dă restul 1 la împărțirea la 30103).



Soluție diamant

Soluția se bazează pe faptul ca valoarea maximă în modul a lui **X** este $1*1+1*2+....+1*19+1*20+2*1+2*2+....2*20+3*1+.....+19*20+20*20$ în cazul în care valorile lui **N** și **M** sunt maxime adică 20 și avem +1 (sau -1) în toată matricea. Suma are valoarea 44100. Odată observat acest fapt se poate deduce că o soluție asemănătoare cu cea a problemei rucsacului se încadrează în limita de timp. Capacitatea rucsacului ar fi **X** iar obiectele ar fi pătrățelele, greutatea fiind $i*j$ pentru pătrățelul de pe linia i coloana j .

Soluție vitale

Există mai multe soluții de rezolvare în complexitate $O(m \log m)$. Dam una dintre ele. Ideea pornește de la algoritmul lui Kruskal pentru arbore de cost minim care adaugă muchii în graf în ordinea crescătoare a costurilor. Pentru toate muchiile de cost fixat x ne uităm la componentele conexe ale grafului ce conține muchiile de cost mai mic decăt x , acest graf va conține niște componente conexe. Putem să le contractăm acele componente conexe în noduri pentru și să ignorăm muchiile de cost mai mare sau egal cu x care evident nu sunt muchii vitale. Acum observăm că muchiile vitale de cost x sunt muchiile critice în graful care conține nodurile menționate anterior și muchiile de cost x . Contractarea nodurilor în componente conexe se realizează folosind structuri de mulțimi disjuncte, iar determinarea muchiilor critice se face cu binecunoscutul algoritm pentru aflarea componentelor biconexe de complexitate $O(m)$. Acest algoritm are complexitate $O(m \log m)$ în faza de sortare, fazele de contracție ale grafului vor avea complexitate totală maxim $O(m \log^* n)$ folosind structuri de mulțimi disjuncte, iar fazele de determinare ale muchiilor critice durează $O(m)$ în total, pentru că la pasul de cost x determinarea componentelor biconexe are complexitate $O(mx)$ unde mx este numărul de muchii de cost x , cum în total numărul de muchii este m , obținem complexitatea $O(m)$. Astfel complexitatea finală este $O(m \log m)$.

Algoritmi neoptimi la care ne-am gândit au fost generarea unui arbore parțial de cost minim, iar apoi eliminarea câte unei muchii din graf după care determinăm arborele de cost minim pe graful obținut. Dacă acest arbore are cost diferit de cel original, evident muchia eliminată a fost una critică. Acest algoritm are complexitatea $O(n * m)$ pentru că avem $n-1$ muchii într-un arbore parțial, iar după ce muchiile grafului sunt sortate algoritmul Kruskal are complexitate $O(m)$. Acesta ar fi luat în jur de 30-40 de puncte.

Alt algoritm ar fi determinarea randomizată a mai multor arbori parțiali și returnarea muchiilor comune tuturor arborilor. Acesta ar fi luat în jur de 40 - 50 de puncte.