Etapa FINALĂ

În continuare vom prezenta soluțiile problemelor propuse spre rezolvare la etapa finală a ediției 2002/2003 a concursului de programare Bursele Agora.

P060314: Dragoni

Această problemă a fost propusă spre rezolvare la runda 17 a concursului având alt enunț și titlul său era *Heroes of Might & Magic*.

Această problemă se reduce la determinarea succesivă a unor înfășurători convexe. Vom determina o înfășurătoare pentru toate cele *N* puncte, apoi o alta pentru punctele rămase după eliminarea celor care se află pe prima înfășurătoare și vom continua până în momentul în care vor fi eliminate toate punctele.

Datorită faptului că toate cele *N* puncte se vor afla pe exact o înfășurătoare, este recomandabil să utilizăm *potrivirea Jarvis*. Determina punctelor de pe a *i*-a înfășurătoare ordinul se realizează în $O(N \cdot h_i)$, unde h_i este numărul de puncte de pe a *i*-a înfășurătoare.

Pentru toate cele *K* înfășurători vom avea ordinul de complexitate:

$$\sum_{i=1}^{K} O(N \cdot h_i) = O(N \cdot N) = O(N^2),$$

deoarece avem $h_1 + h_2 + ... + h_K = N$.

Dacă utilizăm *scanarea Graham*, există posibilitatea să obținem ordinul de complexitate $O(N^2 \cdot \log N)$, deoarece este posibil ca fiecare înfășurătoare să conțină doar trei puncte, deci putem avea până la $[N / 3] + 1$ înfășurători și pentru fiecare dintre acestea ordinul de complexitate este $O(N \cdot \log N)$.

Pe măsura determinării înfășurătorilor vom păstra pentru fiecare punct în parte numărul de ordine al înfășurătorii din care face parte începând cu 1. După determinarea tuturor înfășurătorilor aceste numere de ordine vor fi scrise în fișierul de ieșire.

Analiza complexității

Citirea datelor de intrare implică citirea celor *N* perechi de coordonate ale punctelor de observație, așadar ordinul de complexitate al operației de citire a datelor de intrare este $O(N)$.

Pentru cele *N* puncte vom determina o serie de înfășurători convexe, operație al cărei ordin de complexitate este $O(N^2)$, dacă se utilizează *potrivirea Jarvis*.

Metoda prin care a fost estimat acest ordin (în cadrul prezentării soluției) nu este corectă în întregime din punct de vedere logic. Practic, pentru a *i*-a înfășurătoare numărul de puncte pentru care aceasta este determinată nu este *N*, ci $N - h_1 - h_2 - ... - h_{i-1}$.

Obținem astfel, pentru a *i*-a înfășurătoare următoarea formulă pentru calculul ordinului de complexitate:

$$O\left(\left(N - \sum_{j=1}^{i-1} h_j\right) \cdot h_i\right).$$

Ordinul de complexitate al întregii operații ar deveni:

$$\sum_{i=1}^{K} O\left(\left(N - \sum_{j=1}^{i-1} h_j\right) \cdot h_i\right).$$

Cu toate acestea, dacă efectuăm calculele, rezultatul final va fi, în cazul cel mai defavorabil, tot $O(N^2)$.

Operația de scriere a rezultatelor în fișierul de ieșire are ordinul de complexitate $O(N)$.

În concluzie, ordinul de complexitate a algoritmului de rezolvare a acestei probleme este $O(N) + O(N^2) + O(N) = \boldsymbol{O(N^2)}$.

P060315: Joc

Aceasta este o problemă *NP* iar 15 dintre testele cu care s-au verificat soluțiile au fost generate aleator.

Punctajele obținute de concurenți la această problemă nu au variat foarte mult, dar, după cum era de așteptat, au fost distincte. Problema a avut ca scop departajarea concurenților și a fost folosit un astfel de tip de problemă pentru a fi în concordanță cu rundele de tipul $4 \cdot k + 2$.

Această problemă poate fi rezolvată folosind metoda *backtracking*, dar nici pentru date de test foarte mici, o astfel de rezolvare nu s-ar fi încadrat în timp. O rezolvare *backtracking* pentru această problemă este foarte greu de implementat.

Problema a fost propusă spre rezolvare la un concurs de programare în *PHP*. Aceasta avea numele *Pegs*, enunțul original poate fi găsit pe *site*-ul www.codewalkers.com și diferă de cea propusă la etapa finală prin dimensiunea tablei

n care este tot timpul 7 și timpul de execuție este de 30 de secunde.

Problema se poate rezolva folosind un algoritm euristic care să încerce la fiecare pas stabilirea mutării optime.

Un astfel de algoritm are ordinul de complexitate $O(n)$, dă rezultate suficient de bune comparativ cu cele furnizate de concurenți, poate fi îmbunătățit folosind diverse strategii și pot fi căutate soluții până în momentul anterior expirării timpului de execuție.

Analiza complexității

Ordinul de complexitate al operației de citire a datelor de intrare este $O(m)$, unde m este numărul de scobituri care conțin bile.

Datorită faptului că nu cunoaștem nici o rezolvare care să se încadreze în timpul de execuție, putem spune că ordinul de complexitate pentru abordarea *backtracking* este $6 \cdot O(k!)$, unde constanta 6 reprezintă numărul maxim de mutări care se pot executa pentru a ocupa o scobitură liberă, iar k reprezintă numărul mediu de mutări care se pot executa pentru o anumită configurație. Determinarea valorii k este destul de dificilă.

Ordinul de complexitate al operației de scriere a rezultatelor este $O(m)$.

În concluzie, ordinul de complexitate a algoritmului de rezolvare a acestei probleme este $O(m) + O(m!) + O(m) = \mathbf{O(m!)}$.

P060316: Produse

Pentru a rezolva această problemă considerăm p_k o permutare a mulțimii $\{1, 2, ..., N\}$.

Produsul asociat acestei permutări, conform enunțului problemei, este egal cu $a_{1,p_{k1}} \cdot a_{2,p_{k2}} \cdot ... \cdot a_{N,p_{kN}}$, unde p_{kl} reprezintă valoarea celui de-al l-lea element al permutării p_k. Acest produs este în continuare egal cu $1^2 \cdot p_{k1} \cdot 2^2 \cdot p_{k2} \cdot ... \cdot N^2 \cdot p_{kN}$, care este egal cu $N!^2 \cdot N! = N!^3$.

Datorită faptului că avem $N!$ permutări distincte, vom avea $N!$ produse distincte, deci produsul al cărui număr de cifre trebuie calculat este egal cu $N!^3 \cdot N! = N!^4$.

În acest moment am ajuns la problema determinării numărului de cifre al lui $N!^4$.

Pentru a determina numărul de cifre al lui $N!^4$ pornim de la observația că orice număr natural K poate fi scris sub forma $K = 10^{\lg K}$, de unde rezultă că numărul de cifre al lui K este egal cu $[\lg K] + 1$. Folosind această observație știm că numărul de cifre al lui $N!^4$ este egal cu $[\lg N!^4] + 1 = 4 \cdot [\lg N!] + 1$.

Conform formulei lui *Stirling* de determinare a valorii $\ln N!$ obținem următoarea aproximare:

$$\ln N! \approx \left(N + \frac{1}{2}\right) \cdot \ln N - N + \frac{1}{2} \cdot \ln(2 \cdot \pi \cdot N) + \frac{1}{12 \cdot N} - \frac{1}{360 \cdot N^3}$$

În final, rezultatul problemei este dat de expresia:

$$[4 \cdot \ln N!/\ln 10] + 1.$$

Problema determinării numărului de cifre al lui $n!$ a devenit deja o problemă clasică, ea apărând pentru prima dată la *Olimpiada Balcanică de Informatică* din 1997, unde a fost propusă de dl. prof. dr. *Adrian Atanasiu*. Această problemă a fost propusă spre rezolvare și la runda a 2-a a concursului de programare *Bursele Agora* ediția 1999/2000.

Analiza complexității

Ordinul de complexitate a operației de citire a datelor de intrare este $O(1)$.

Ordinul de complexitate a operației de determinare a numărului de cifre al lui $N!^4$ este $O(1)$.

Ordinul de complexitate a operației de scriere a rezultatelor este $O(1)$.

În concluzie, ordinul de complexitate a algoritmului de rezolvare pentru această problemă este $O(1) + O(1) + O(1) = \mathbf{O(1)}$.

P060317: Mijloc

Pentru a rezolva această problemă vom împărți, cel puțin teoretic, șirul inițial care are $2 \cdot N$ elemente în două șiruri a și b care au N elemente și sunt sortate crescător.

Datorită faptului că șirul inițial are lungimea $2 \cdot N$ se observă că există două elemente care, dacă sortăm șirul, se află la mijlocului lui.

Pentru a determina poziția pe care se află unul dintre elementele mediane ale șirului inițial (elementul pentru care există, în șirul inițial, N elemente mai mari sau egale cu el și N elemente mai mici sau egale cu el) vom folosi algoritmul descris în continuare.

La fiecare pas considerăm elementele i și j care se află la mijlocul celor două șiruri a și b. Dacă $i > j$ atunci vom elimina din șirul a toate elementele care se află înaintea elementului i și din șirul b toate elementele care preced elementul j, altfel vom elimina din șirul a toate elementele care preced elementul i și din șirul b toate elementele care se află înaintea elementului j.

Acest algoritm se încheie în momentul în care șirurile a și b au un singur element. Unul dintre aceste două elemente este soluția problemei. Cele două elemente care rămân nu reprezintă cele două mediane ale șirului inițial decât dacă nu se află în aceeași jumătate a acestuia.

Algoritmul prezentat efectuează $\log N$ apeluri ale funcției `IsGreater` furnizată de bibliotecă și cele două apeluri permise ale procedurii `Result`.

Analiza complexității

Pentru această problemă datele sunt citite de către biblioteca folosită.

Ordinul de complexitate a operației de determinare a unui element median este $O(\log N)$.

Ordinul de complexitate a operației de furnizare a rezultatului este $O(1)$.

În concluzie, ordinul de complexitate a algoritmului de rezolvare pentru această problemă este $O(1) + O(\log N) = \mathbf{O(\log N)}$.