IOI 2003

În perioada 16-23 august 2003 a avut loc cea de-a XV-a ediție a concursului internațional de programare IOI. În continuare vă vom prezenta soluțiile problemelor propuse spre rezolvare la acest concurs.

P060301: Comparări de cod

La început, vom înlocui, în ambele programe, numele variabilelor prin indici. În acest mod, detaliile legate de o variabilă vor fi accesate în $O(1)$.

Presupunem alese două secvențe de L linii, câte una din fiecare program. Să vedem cum putem verifica dacă aceste secvențe se potrivesc.

Fie linia i din fiecare set. Este clar că variabilele din stânga semnului "=" trebuie să se potrivească. Marcăm această potrivire. Numărul de operații este constant pentru fiecare linie.

Dacă în partea dreaptă se află aceeași variabilă de două ori, este clar că ea trebuie să se potrivească. Avem, din nou, număr constant de operații pentru fiecare linie.

Dacă variabila X nu apare decât în partea dreaptă a ecuațiilor din unul din programe, atunci toate ecuațiile corespunzătoare din programul celălalt trebuie să aibă cel puțin o variabilă în comun. Acest pas se poate rezolva în $O(L)$.

Dacă am stabilit că o variabilă X se potrivește cu o variabilă Y (din celălalt program) și ulterior obținem că se potrivește cu o variabilă Z, atunci secvențele de linii nu se potrivesc.

Condițiile se verifică pentru ambele programe. Nerespectarea unei condiții înseamnă că secvențele de linii nu se potrivesc. Este evident că aceste condiții sunt necesare pentru ca secvențele să se potrivească. Se poate demonstra că ele sunt și suficiente.

Partea mai puțin intuitivă este cea în care nu știm dacă o variabilă X din partea dreaptă a unei linii se potrivește cu Y sau cu Z, și verificăm dacă toate liniile corespondente liniilor în care apare X au cel puțin o variabilă în comun.

Să analizăm cazurile care apar:

- liniile nu au nici o variabilă în comun; atunci, secvențele de L linii nu se potrivesc;
- liniile au două variabile în comun; atunci, în toate liniile în care apare X apare o altă variabilă T, și putem potrivi fie X cu Y și T cu Z, fie invers. În caz contrar, secvențele nu se potrivesc.
- liniile au o singură variabilă în comun; atunci, ea corespunde lui X.

Folosind această metodă de verificare putem construi o primă variantă de rezolvare: alegem, pe rând, câte o linie din fiecare program și determinăm lungimea celei mai lungi secvențe care se potrivește, începând cu liniile respective. Determinarea se realizează cu ajutorul metodei de mai sus, folosind căutarea binară pentru alegerea lungimii.

Acest algoritm ar fi obținut aproximativ 55% din punctajul total.

Pentru a obține punctajul maxim se procedează în felul următor:

- se alege, pe rând, o diferență între indicii liniilor la care se caută cele două secvențe identice. Diferența variază de între $-R$ la R. Astfel, dacă o secvență începe la linia a treia a programului *RBN*, iar cealaltă la linia 7 din programul *HAL*, diferența este 4.
- pentru fiecare diferență aleasă se pornește de la primele două linii cu diferența respectivă între numerele lor de ordine (de exemplu, de la liniile 1 și 5 pentru diferența 4) și se construiește o secvență cât mai lungă care să se potrivească, începând cu aceste două linii. Aceasta se face prin adăugarea de linii la sfârșitul secvențelor, cât timp condițiile sunt respectate. În momentul în care condițiile sunt încălcate, se elimină linii de la începutul secvențelor, până când condițiile sunt din nou respectate.

Acest algoritm necesită actualizări incrementale legate de respectarea condițiilor, la adăugarea și eliminarea unei linii. Pentru ca algoritmul să fie eficient, actualizările trebuie să se realizeze în timp constant.

Analiza complexității

Ordinul de complexitate al operației de citire a datelor de intrare este $O(R + H)$, iar cel al operației de scriere a rezultatelor este $O(1)$.

Pentru fiecare diferență cuprinsă între $-R$ și R se efectuează $O(H)$ operații, presupunând că actualizarea se face în

timp constant, deci ordinul de complexitate al acestei operații este $O(R \cdot H)$.

În final, ordinul de complexitate al algoritmului de rezolvare pentru această problemă este $O(R \cdot H)$.

P060302: Întreținerea potecilor

Această problemă admite două variante eficiente de rezolvare. Ambele rezolvări ar fi obținut punctajul maxim în cadrul acestui concurs.

Prima variantă constă în a reconstrui, după fiecare săptămână, arborele parțial de cost minim, folosind algoritmul lui *Kruskal*.

Arborele parțial determinat după o anumită săptămână poate conține numai muchii din arborele parțial anterior, plus muchia introdusă în săptămâna respectivă.

Acest fapt este evident dacă ne gândim la modul de funcționare a algoritmului lui *Kruskal*.

Muchiile sunt sortate după cost; muchia nouă ar intra pe o anumită poziție.

Presupunem că reconstruim arborele. Este clar că, parcurgând lista de muchii până înainte de poziția muchiei nou introduse, se obține aceeași configurație ca la construirea precedentă a arborelui.

Dacă muchia nouă ar fi respinsă, deoarece ar introduce un ciclu, s-ar obține același arbore. Dacă muchia nouă unește două componente, atunci una dintre muchiile care au fost în arborele precedent va fi respinsă la un moment dat ca introducând un ciclu și, ceea ce este mai important, nici o muchie care a fost respinsă la construirea arborelui de la pasul precedent nu ar fi acceptată, deoarece capetele ei ar fi în aceeași componentă.

Se poate face o demonstrație formală prin inducție.

A doua variantă constă în construirea arborelui de la un anumit pas pe baza arborelui de la pasul precedent.

La introducerea unei noi muchii, între capetele ei există un singur drum în arbore. Dacă muchia de lungime maximă de pe acest drum este mai lungă decât muchia nouă, atunci este de preferat să o eliminăm și să o introducem pe cea nouă. Astfel se obține un arbore cu un cost mai bun.

Extinderea acestor doi algoritmi pentru a funcționa și la început, când încă nu s-a format un arbore, este foarte simplă. Nici una din variante nu ridică dificultăți deosebite la implementare.

Analiza complexității

Operațiile de citire a datelor de intrare și scriere a rezultatelor au ordinul de complexitate $O(W)$.

Primul algoritm constă în determinarea a W arbori parțiali, pentru N grafuri având cel mult N muchii fiecare. Deoarece algoritmul lui *Kruskal* pentru determinarea arborelui parțial de cost minim al unui graf are ordinul de complexitate $O(M \cdot \log N)$, unde M este numărul total de muchii al grafului, ordinul de complexitate al acestui algoritm este $O(W \cdot N \cdot \log N)$.

În cazul celui de-al doilea algoritm, adăugarea unei noi muchii se face în $O(N)$, deci acesta are ordinul de complexitate $O(W \cdot N)$.

Se poate observa că ordinul de complexitate al rezolvării acestei probleme nu depinde de ordinul de complexitate al operațiilor de citire a datelor de intrare și scriere a rezultatelor, ci depinde numai de ordinul de complexitate al algoritmului utilizat.

P060303: Inversare

Pentru această problemă, concurenții cunoșteau datele de intrare; sarcina lor era să furnizeze fișierele de ieșire corespunzătoare.

Pentru $N < 9$, se inițializau regiștrii cu valorile N, N-1, ..., 0 și se afișau aceste valori. Se efectuau 0 operații S.

Pentru valori mai mari ale lui N se folosesc metode euristice. Să examinăm una dintre acestea.

Evident, unul din regiștri va fi blocat pe valoarea 0; în caz contrar, 0 nu poate fi afișat. Astfel, mai rămân 9 regiștri de lucru.

Încercăm să rezolvăm problema folosind o singură operatie S. Inițializăm primul registru cu N, următorul cu N - 2. Afișăm N, apoi primul registru poate fi modificat, deci îl facem egal cu N - 1 printr-o operație S. Afișăm N - 1, apoi putem face un nou S, afișăm N - 2, facem un nou S, și trebuie să afișăm N - 3.

Putem pune N - 5 în al treilea registru. În acest caz, primul S nespecificat ar fi pus primul registru pe N - 4, iar al doilea S ar fi pus al doilea registru pe N - 3, după care s-ar fi afișat N - 3. Al treilea registru rămâne nemodificat până la afișarea lui N - 5, după care poate fi folosit ca registru de lucru, la fel cum au fost folosiți primii doi.

Continuând acest procedeu, se pot rezolva testele pentru care valoarea lui N este mai mică sau egală cu 44.

Generalizând această strategie pentru mai multe operații S consecutive, s-ar fi obținut aproximativ 90 de puncte.

P060304: Frontiera

Considerăm că NR este numărul maxim de vârfuri ale unui poligon.

Considerăm că observatorul (*Fermierul Don*) are un câmp vizual de 360 grade. În acest câmp vizual, la anumite unghiuri, care se vor calcula, se află stâlpii gardului.

Fiecare poligon blochează o porțiune din câmpul vizual al observatorului. Se poate determina această porțiune într-un timp de $O(NR)$.

Porțiunile pot fi puse în corespondență cu intervale închise cu capetele între 0 și 360 de grade, prin "deschiderea" câmpului vizual al observatorului; se observă că poligoanele care intersectează axa de "deschidere" vor conduce la formarea a două intervale.

Se sortează aceste intervale în funcție de punctul de început, apoi, folosind un algoritm similar sortării prin interclasare, se determină stâlpii gardului care nu aparțin nici unui interval.

Analiza complexității

Ordinul de complexitate al operației de citirea a datelor de intrarea este $O(R \cdot NR)$.

Algoritmul de determinare a porțiunilor blocate din câmpul vizual și transformare a lor în intervale are ordinul de complexitate $O(R \cdot NR)$.

Algoritmul de sortare a porțiunilor blocate are ordinul de complexitate $O(R \cdot \log R)$, iar algoritmul de determinare a stâlpilor vizibili are ordinul de complexitate $O(N + R)$.

În concluzie, ordinul de complexiatea al algoritmului de rezolvare pentru această problemă este $O(N + R \cdot (NR + \log R))$.

P060305: Ghicește vaca

În continuare, prin a rezolva o submulțime înțelegem stabilirea unei strategii de a pune întrebări pentru a determina vaca, optimă din punct de vedere al numărului maxim de întrebări, și de calculare a acestui număr.

Abordarea intuitivă constă în a alege la fiecare pas întrebarea care împarte vacile cât mai echitabil, din punct de vedere al numărului de vaci din submulțimile în care se face împărțirea. Această soluție nu este optimă; o împărțire echitabilă nu asigură faptul că submulțimile în care s-a făcut împărțirea se vor rezolva ușor. Astfel, probabil că este preferabil să împărțim vacile în submulțimi, nu neapărat apropiate ca număr de elemente, dar care se pot rezolva mai ușor. În plus, dacă există mai multe întrebări care ar conduce la o împărțire la fel de bună din punct de vedere al numărului de vaci din submulțimi, nu avem nici o modalitate de a alege una dintre ele. În orice caz, o astfel de rezolvare ar fi obținut aproximativ 85 puncte, deci un punctaj mulțumitor.

Rezolvarea optimă a problemei constă în a alege, la fiecare pas, întrebarea care împarte vacile, astfel încât numărul de întrebări pentru rezolvarea celei mai mari submulțimi rezultate din împărțire să fie minim. Această informație nu este cunoscută inițial, deci va trebui să fie calculată.

Rezultă o funcție recursivă care, primind o submulțime a mulțimii vacilor, încearcă, pe rând, fiecare întrebare, se reapelează pentru submulțimile rezultate din împărțire și întoarce numărul minim de întrebări.

Pentru a evita recalcularea unor valori, funcția va testa, la început, dacă valoarea pe care trebuie să o returneze a fost calculată deja, la un apel anterior; această operație se realizează prin memorarea valorilor calculate. Acesta este algoritmul intuitiv; la implementare se vor aduce optimizări de cod.

Aparent există cel mult 2^{50} submulțimi care vor fi testate, deci funcția nu se va încadra în timp. În practică, nu toate cele cel mult 2^{50} submulțimi sunt accesibile folosind întrebările pentru departajare.

Fiecare submulțime care va fi testată poate fi identificată prin mulțimile de valori ale celor cel mult 8 atribute. Pentru un atribut fixat există cel mult șapte mulțimi posibile de valori, excluzând mulțimea vidă ($\{x\}$, $\{y\}$, $\{z\}$, $\{x, y\}$, $\{x, z\}$, $\{y, z\}$). În total există cel mult 7^8 submulțimi, adică mai puțin de 6 milioane de submulțimi.

Implementarea este destul de dificilă, deoarece codul trebuie optimizat.

Este mai eficientă utilizarea unei metode nerecursive similară. Astfel, metoda recursivă realizează o parcurgere în adâncime într-un graf care are drept noduri submulțimile mulțimii vacilor, având câte un arc de la fiecare mulțime la mulțimile în care ea se împarte punând întrebari. Putem înlocui această parcurgere cu o parcurgere în lățime.

Deoarece această metodă se poate obține din cea recursivă cu puține modificări, se pot testa ambele variante și se reține cea mai rapidă.

P060306: Roboții uimitori

Deoarece lungimile traseelor paznicilor sunt mici, se observă că, după 12 minute, toți paznicii se vor întoarce în pozițiile inițiale. Astfel, singura modificare în spațiul problemei, de la minutul i la minutul $i + 12$, constă în pozițiile roboților.

Construim o matrice tridimensională A, în care $A[P_1, P_2, t]$ reprezintă numărul de mutări pentru a aduce primul robot în poziția P_1 și al doilea robot în poziția P_2, la un moment de timp de forma $12 \cdot k + t$. P_1 identifică o poziție a primului robot în primul labirint, sau faptul că acesta a ieșit din labirint. P_2 identifică, în mod similar, starea celui de-al doilea robot.

Elementele matricei A se completează folosind algoritmul lui *Lee*. Practic, o stare este dată de P_1, P_2 și t; din această stare se poate ajunge în cel mult 4 stări, folosind comenzile disponibile. Unele din aceste stări sunt invalide, deoarece duc la capturarea unuia dintre roboți.

Starea inițială este introdusă într-o coadă și la fiecare pas, în coadă sunt introduse apoi stările valide în care se poate ajunge din ea.

Această soluție, dacă ar fi fost implementată corect, ar fi obținut punctajul maxim.

Analiza complexității

Ordinul de complexitate al operației de citirea a datelor de intrare este $O(R_1 \cdot C_1 + G_1 + R_2 \cdot C_2 + G_2)$.

Algoritmul de determinare a comenzilor care trebuie transmise roboților are ordinul de complexitate $O(12 \cdot P_1 \cdot P_2)$, unde P_1 și P_2 reprezintă, în acest caz, numărul de poziții prin care poate trece primul robot, respectiv al doilea robot și aceste numere sunt mărginite superior de $R_1 \cdot C_1$, respectiv $R_2 \cdot C_2$.

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

Nota redacției

Toate variabilele, care au fost folosite în rezolvarea acestor probleme și nu au fost explicate, au semnificația dată în enunțurile problemelor, care au fost publicate în *GInfo* 13/6 (octombrie 2003).