ONI 2002, Brăila

Olimpiada NAȚIONALĂ de Informatică

Vă prezentăm în continuare soluțiile problemelor propuse la ONI 2002 la clasele a IX-a și a X-a. Aceste soluții au fost realizate de redacția GInfo pe baza soluțiilor oficiale prezentate de autorii problemelor și au fost verificate folosind seturile de date cu care au fost testate soluțiile concurenților.

P050201: Pentagon

Rezolvarea acestei probleme este foarte simplă, ea neimplicând decât parcurgerea pe coloane a unei matrice, contorizarea unor secvențe de zerouri și păstrarea unei statistici referitoare la aceste secvențe.

Pentru a determina blocurile necesare, vom determina blocurile pe fiecare coloană (datorită faptului că lățimea unui bloc este întotdeauna 1 și toate blocurile sunt dispuse pe verticală, un bloc poate ocupa o singură coloană).

Pentru a determina blocurile de pe o coloană va trebui să determinăm secvențele de zerouri de pe coloana respectivă. Vom lua în considerare doar secvențele de lungime maximă pentru a minimiza numărul total de blocuri. De exemplu, dacă avem șase zerouri consecutive, am putea folosi un bloc de lungime 6, dar și două blocuri de lungime 5 și 1, 4 și 2 etc. Evident, este obligatoriu să folosim un singur bloc pentru ca numărul total al blocurilor utilizate să fie minim.

Așadar, pentru fiecare coloană vom determina lungimile secvențelor de zerouri. O secvență de zerouri poate începe fie pe prima linie a coloanei, fie în momentul în care întâlnim o linie pe care se află un element cu valoarea 0 în timp ce pe linia anterioară se află un element cu valoarea 1. Secvența se va termina fie la terminarea parcurgerii coloanei (se ajunge pe ultima linie a acesteia), fie în momentul în care întâlnim o linie pe care se află un element cu valoarea 1 în timp ce pe linia anterioară se află un element cu valoarea 0.

În momentul detectării terminării unei secvențe (presupunem că lungimea acesteia este x), numărul blocurilor de lungime x este incrementat. Pentru păstrarea numărului de blocuri se utilizează un șir a, unde a_x indică numărul blocurilor de lungime x.

La sfârșit, vom afișa statistica cerută pe baza datelor păstrate în șirul a. Vor fi afișate toate perechile de forma i a_i care respectă condiția $a_i \neq 0$.

Analiza complexității

Datorită faptului că numărul de elemente care compun o secvență de zerouri poate fi determinat pe măsură ce este parcursă secvența, întregul algoritm constă într-o singură traversare a matricei. Ordinul de complexitate al unei astfel de traversări este $O(m \cdot n)$, unde m reprezintă numărul de linii ale matricei, iar n reprezintă numărul de coloane.

Pentru citirea datelor, matricea este parcursă o singură dată, deci ordinul de complexitate al acestei operații este tot $O(m \cdot n)$.

Pentru afișarea datelor se parcurge o singură dată șirul a; acesta nu poate conține mai mult de m elemente deoarece nu pot fi folosite blocuri mai înalte decât înălțimea zidului. Așadar, scrierea soluției are ordinul de complexitate $O(m)$.

În concluzie, ordinul de complexitate al algoritmului de rezolvare a acestei probleme este $O(m \cdot n) + O(m \cdot n) + O(m) = O(m \cdot n)$.

Dificultate	12
Algoritmi	★☆☆☆☆
Structuri de date	★☆☆☆☆
Originalitate	★☆☆☆☆
Cunoștințe	★☆☆☆☆
Implementare	★☆☆☆☆

www.ginfo.ro/revista/12_6/surse/pentagon.cpp

P050202: Pod

Rezolvarea problemei se bazează pe o variantă simplă a metodei programării dinamice. Se observă foarte ușor că numărul de posibilități de a ajunge pe cea de-a i-a scândură depinde doar de numărul de posibilități de a ajunge pe cele trei scânduri aflate în fața ei. Vom nota cu t_i numărul de posibilități de a ajunge pe cea de-a i-a scândură. Vom con-

sidera malul opus ca fiind cea de-a (N + 1)-a scândură, unde N este numărul scândurilor care formează podul. Soluția problemei va fi dată de valoarea t_{N+1}.

În cazul în care cea de-a i-a scândură lipsește, pe ea nu se poate ajunge, deci vom avea $t_i = 0$.

În cazul în care această scândură există, dar este deteriorată, pe ea se poate ajunge doar de pe scândura precedentă, deci vom avea $t_i = t_{i-1}$.

În cazul în care scândura există și nu este deteriorată, pe ea se poate ajunge de pe scândura anterioară (chiar dacă este deteriorată) sau de pe oricare dintre precedentele două (dacă nu sunt deteriorate). Pentru a exprima relația matematică pentru t_i, vom folosi funcția s_i definită prin $s_i = t_i$ dacă a i-a scândură există și nu este deteriorată și $s_i = 0$ în caz contrar. Folosind această funcție, relația este $t_i = s_{i-3} + s_{i-2} + t_{i-1}$.

Inițial, vom avea $t_0 = 1$, deoarece se poate spune că există o singură posibilitate de a ajunge pe malul pe care ne aflăm.

Datorită faptului că valorile t_i pot avea până la 80 de cifre, este necesară simularea operației de adunare pentru numere mari.

Determinarea unei soluții corecte se poate realiza destul de ușor dacă, la fiecare pas, păstrăm indicele unei scânduri anterioare de pe care se poate trece pe scândura curentă. Acest indice va fi fie cel al scândurii anterioare (dacă aceasta există și numărul posibilităților de a ajunge la ea este nenul), fie al uneia dintre precedentele două (dacă există, nu este deteriorată și numărul posibilităților de a ajunge la ea este nenul). Dacă valoarea t_{N+1} este nenulă, atunci există cu siguranță cel puțin o soluție. În final, modalitatea de traversare va fi generată cu ajutorul unei tehnici recursive foarte simple.

Analiza complexității

Operația de adunare a numerelor mari are ordinul de complexitate $O(NCif)$, unde $NCif$ este numărul de cifre al celui mai mare dintre numerele care se adună. Deoarece $NCif$ este cel mult 80, ordinul de complexitate al operației de adunare poate fi considerat a fi $O(80) = 80 \cdot O(1) = \boldsymbol{O(1)}$.

Trebuie efectuate cel mult $2 \cdot N$ astfel de adunări, deci operația de determinare a valorilor t_i are ordinul de complexitate $O(N) \cdot O(1) = \boldsymbol{O(N)}$.

Pentru reconstituirea drumului, la fiecare pas trebuie păstrat indicele unei scânduri precedente de pe care se poate ajunge pe scândura curentă. Există doar trei posibilități de a ajunge pe scândura curentă, deci ordinul de complexitate al acestei operații este $O(1)$. Pentru determinarea scândurii anterioare corespunzătoare fiecărei scânduri din componența podului sunt efectuate $O(N)$ astfel de operații, deci ordinul de complexitate al operației de determinare a unei modalități de traversare este $\boldsymbol{O(N)}$.

Citirea datelor de intrare se realizează într-un timp cu ordinul de compelexitate $\boldsymbol{O(N)}$ deoarece pot exista cel mult N - 1 scânduri care lipsesc și cel mult N care sunt deteriorate.

Scrierea datelor de ieșire constă în generarea unei modalități de traversare care are lungimea cel mult N și a numărului modalităților de traversare. Deoarece determinarea scândurii precedente se realizează pe baza unor indici păstrați pentru fiecare scândură în parte, ordinul de complexitate al acestei operații este $O(1)$. Datorită faptului că vor fi cel mult N astfel de determinări, ordinul de complexitate al operației de determinare a modalității de traversare este $O(N)$. Scrierea numărului posibilităților de traversare poate fi considerat a fi o operație elementară, deci are ordinul de complexitate $O(1)$. Așadar, ordinul de complexitate al operației de scriere a datelor de ieșire este $O(N) + O(1) = \boldsymbol{O(N)}$.

În concluzie, algoritmul de rezolvare al acestei probleme are ordinul de complexitate $O(N) + O(N) + O(N) + O(N) = \boldsymbol{O(N)}$.

Dificultate 25	
Algoritmi	★★☆☆☆
Structuri de date	★☆☆☆☆
Originalitate	★☆☆☆☆
Cunoștințe	★★☆☆☆
Implementare	★★☆☆☆

www.ginfo.ro/revista/12_6/surse/pod.cpp

P050203: Suma

Pentru ca numărul termenilor să fie minim, trebuie ca suma termenilor care apar cu semn negativ să fie cât mai mică posibil.

Inițial vom presupune că nu va trebui să scădem nici o valoare și ne propunem să găsim cel mai mic număr N pentru care $S_N = \sum_{i=1}^{N} i \geq S$. Valoarea N se obține rezolvând ecuația de gradul II $S_N = S$ (vom avea $S_N \leq S + N$) și rotunjind prin adaos rădăcina pozitivă obținută. Avem:

$$\frac{N \cdot (N+1)}{2} = S$$

$$N^2 + N - 2 \cdot S = 0$$

$$N = \frac{\sqrt{1+8 \cdot S} - 1}{2}$$

Așadar, valoarea N este $\left\lceil \frac{\sqrt{1+8 \cdot S} - 1}{2} \right\rceil$. Datorită faptului că N este cea mai mică valoare pentru care suma atinge sau depășește valoarea S, soluția problemei nu poate fi dată de nici un număr $N' < N$. În cele ce urmează vom demonstra că soluția problemei este dată de N, $N + 1$ sau $N + 2$. Se observă că numerele S_N, S_{N+1} și S_{N+2} nu pot avea toate aceeași paritate, deoarece $N + 1$ și $N + 2$ au parități diferite. Vom nota prin D_N diferența dintre valoarea S_N și valoarea care trebuie obținută (S). Deoarece se scade aceeași valoare din S_N, S_{N+1} și S_{N+2} este evident că D_N, D_{N+1} și D_{N+2} nu pot avea toate aceeași paritate.

Soluția problemei este dată de indicele celui mai mic număr par dintre aceste trei numere; fie acest indice N'.

Suma totală a elementelor cărora trebuie să le fie modificat semnul este $D_{N'}/2$. Prin modificarea semnului unui element, valoarea expresiei scade cu dublul acestui element, deci avem $S_{N'} - D_{N'} / 2 \cdot 2 = S$.

Fie $x = D_{N'} / 2$ (suma totală a elementelor care trebuie scăzute). Deoarece $N' \le N + 2$, obținem $S_{N'} \le S_N + N + 1 + N + 2$. Folosind relația $S_N \le S + N$, se obține relația $S_{N'} \le S + 3 \cdot N + 3$. Cu alte cuvinte, avem $x \le \frac{3 \cdot N + 3}{2} \le \frac{3 \cdot N' + 3}{2}$. Așadar, valoarea x poate fi obținută alegând doar două numere cuprinse între 1 și N'. În cazul în care $x > N'$ vom scădea N' și $N' - x$, iar în caz contrar vom scădea doar x.

Analiza complexității

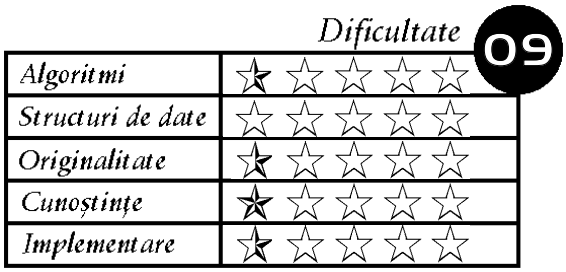
Valoarea N poate fi obținută folosind o simplă formulă matematică, deci ordinul de complexitate al acestei operații este $O(1)$.

Valorile S_N, S_{N+1}, S_{N+2}, D_N, D_{N+1}, D_{N+2} sunt calculate tot cu ajutorul unor formule matematice simple, așadar ordinul de complexitate al operației de determinare a acestora este tot $O(1)$.

Identificarea valorii N' se face pe baza verificării parității a cel mult trei numere, deci și această operație are ordinul de complexitate $O(1)$.

Determinarea valorii x și a celor cel mult două numere cărora li se va modifica semnul este realizată tot pe baza unor calcule simple al căror ordin de complexitate este $O(1)$.

În concluzie, ordinul de complexitate al unui algoritm eficient de rezolvare a acestei probleme este $O(1) + O(1) + O(1) + O(1) = O(1)$.



www.ginfo.ro/revista/12_6/surse/suma.cpp

P050204: Becuri

Este evident că pentru a obține prima linie a matricei nu putem decât fie să comutăm toate coloanele pe care trebuie să se afle becuri aprinse și să nu comutăm prima linie, fie să comutăm prima linie și să comutăm toate coloanele pe care trebuie să se afle becuri stinse. Analog, pentru prima coloană putem fie să comutăm toate liniile pe care trebuie să se afle becuri aprinse și să nu comutăm prima coloană, fie să comutăm prima coloană și să comutăm toate liniile pe care trebuie să se afle becuri stinse.

Astfel avem patru posibilități de a obține configurația cerută:

- comutăm liniile cărora le corespund becuri aprinse pe prima coloană și coloanele cărora le corespund becuri aprinse pe prima linie;
- comutăm liniile cărora le corespund becuri aprinse pe prima coloană și coloanele cărora le corespund becuri stinse pe prima linie;
- comutăm liniile cărora le corespund becuri stinse pe prima coloană și coloanele cărora le corespund becuri aprinse pe prima linie;
- comutăm liniile cărora le corespund becuri stinse pe prima coloană și coloanele cărora le corespund becuri stinse pe prima linie.

Vom verifica fiecare dintre aceste patru variante și apoi o vom alege pe cea care implică cele mai puține comutări de linii și coloane.

Dacă nici una dintre cele patru variante nu duce la obținerea configurației cerute, putem trage concluzia că aceasta nu poate fi obținută prin comutarea unor linii și a unor coloane.

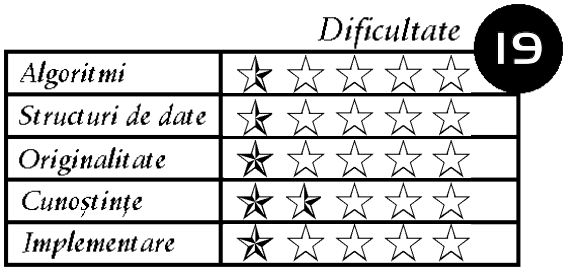
Analiza complexității

Citirea datelor de intrare implică parcurgerea unei matrice pătratice, deci ordinul de complexitate al acestei operații este $O(N^2)$.

Verificarea fiecăreia dintre cele patru posibilități de a obține soluția implică parcurgerea primei linii și a primei coloane; ordinul de complexitate al unei astfel de parcurgere este $O(N) + O(N) = O(N)$. Pentru fiecare element este posibil ca linia/coloana corespunzătoare să fie comutată. Operația de comutare implică traversarea liniei sau coloanei, așadar are ordinul de complexitate $O(N)$. Ca urmare, ordinul de complexitate al fiecăreia dintre cele patru posibilități este $O(N) \cdot O(N) = O(N^2)$. Întreaga operație de determinare a soluției are ordinul de complexitate $4 \cdot O(N^2) = O(N^2)$.

Scrierea datelor de ieșire implică traversarea șirurilor care indică dacă o linie sau coloană a fost comutată; ordinul de complexitate al operației este $O(N)$.

În concluzie, algoritmul de rezolvare al acestei probleme are ordinul de complexitate $O(N^2) + O(N^2) + O(N) = O(N^2)$.



www.ginfo.ro/revista/12_6/surse/becuri.cpp

P050205: Discuri

Pentru identificarea discurilor dispensabile va trebui să determinăm atingerile dintre discurile care influențează lățimea figurii. De exemplu, dacă avem șase discuri, cu raze de `1000`, `1`, `2`, `3`, `1000`, respectiv `500`, atunci atingerile care influențează lățimea figurii sunt între primul și al cincilea disc, respectiv între al cincilea și al șaselea.

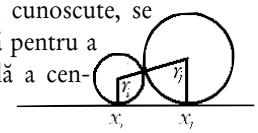
Pentru fiecare disc *i* vom determina, pe rând, coordonatele orizonatale pe care le-ar avea discul dacă ar atinge unul dintre discurile anterioare. În final, vom alege discul (sau axa *Oy*) pentru care coordonata orizontală a centrului noului disc este maximă și putem afirma că discul *i* ajuge în această poziție. Dacă discul *i* va atinge un disc anterior *j*, atunci discurile cu numerele de ordine cuprinse între *j* + 1 și *i* - 1 sunt dispensabile.

După ce vom lua în considerare toate cele *N* discuri, vom putea determina numerele de ordine ale tuturor discurilor dispensabile.

În final vom verifica dacă există discuri introduse la sfârșit care sunt dispensabile. Petntru aceasta vom determina lățimea figurii și ultimul disc care o influențează. Toate discurile introduse după acest disc sunt dispensabile.

Pentru determinarea coordonatei orizontale x_i a centrului unui disc *i* care atinge un disc *j* avem nevoie de coordonata orizontală x_j a centrului discului *j*, precum și de razele r_i și r_j ale celor două discuri.

Dacă aceste trei valori sunt cunoscute, se poate folosi următoarea formulă pentru a determina coordonata orizontală a centrului discului *j*:



$$x_j = x_i + \sqrt{(r_i + r_j)^2 - (r_i - r_j)^2}.$$

Analiza complexității

Pentru fiecare disc *i* care este introdus, se determină posibila coordonată x_i datorată atingerii cu toate cele *i* - 1 discuri inserate anterior. Pentru cele *N* discuri se vor determina, în total, $0+1+2+\cdots+N-1=\frac{N\cdot(N-1)}{2}$ coordonate, deci ordinul de complexitate al acestei operații este $O(N^2)$.

La fiecare pas, pot fi marcate ca dispensabile cel mult toate discurile inserate anterior, așadar ordinul de complexitate al acestei operații este tot $O(N^2)$.

Determinarea lățimii figurii și a cercurilor dispensabile de la sfârșitul secvenței necesită a singură parcurgere a șirului care păstrează coordonatele centrelor discurilor, ceea ce implică ordinul de complexitate $O(N)$.

Afișarea cercurilor dispensabile precum și citirea razelor cercurilor sunt operații care se efectuează în timp liniar ($O(N)$), necesitând o simplă parcurgere a unor șiruri.

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

Dificultate	10
Algoritmi	★☆☆☆☆
Structuri de date	★☆☆☆☆
Originalitate	★★☆☆☆
Cunoștințe	★☆☆☆☆
Implementare	★☆☆☆☆

www.ginfo.ro/revista/12_6/surse/discuri.cpp

P050206: Cod

Din condițiile precizate în enunț rezultă că, pe baza unei mulțimi de litere distincte, se poate construi un singur cuvânt care respectă condițiile date, și anume cel care conține literele ordonate lexicografic. Așadar, oricărei mulțimi de cel mult zece litere distincte îi corespunde un cuvânt care respectă condițiile din enunț. Se poate afirma că un cuvânt este o submulțime a mulțimii literelor; de aici rezultă că numărul cuvintelor formate din *k* litere este C_{26}^k. Mai mult, numărul cuvintelor format din *k* dintre ultimele *n* litere ale alfabetului este C_n^k.

Numărul de ordine al cuvântului dat este mai mare decât cel al cuvintelor formate din mai multe cifre. Din aceste motive, pentru un cunvânt format din *k* litere, vom avea un număr mai mare decât $\sum_{i=1}^{k} C_{26}^i$.

În continuare, pentru prima literă a cuvântului, va trebui să găsim numărul cuvintelor care încep cu o literă mai mică (din punct de vedere lexicografic). În cazul în care cuvântul are *k* litere, vor exista C_{25}^{k-1} cuvinte valide care încep cu `'a'`, C_{24}^{k-1} cuvinte valide care încep cu `'b'` etc. În general, vor exista C_{26-i}^{k-1} cuvinte valide care încep cu a *i*-a literă a alfabetului. Dacă prima literă a cuvântului este cea de-a *n*-a literă a alfabetului, vom avea $\sum_{i=1}^{n} C_{26-i}^{k-1}$ cuvinte valide care încep cu o literă mai mică.

În acest moment știm numărul de ordine minim al unui cuvânt care începe cu prima literă a cuvântului dat. Pentru a doua literă vom proceda într-o manieră asemănătoare. Singura diferență este dată de faptul că a doua literă trebuie să fie strict mai mare decât prima. Așadar, dacă prima literă este cea de-a *n*-a a alfabetului, iar a doua este cea de-a *m*-a, atunci vom avea $\sum_{i=n+1}^{m-1} C_{26-i}^{k-2}$ cuvinte care au pe prima poziția aceeași literă, iar pe cea de-a doua poziție o literă mai mică.

Procedeul va continua pentru fiecare literă în parte. În cazul în care litera curentă este cea de-a *p*-a a cuvântului, este a *m*-a a alfabetului, iar litera anterioară este a *n*-a a alfabetului, numărul de cuvite care au pe primele *p* - 1 poziții aceleași litere ca și cuvântul dat, iar pe cea de-a *p*-a o literă mai mică este dat de formula $\sum_{i=n+1}^{m-1} C_{26-i}^{k-p}$.

Adunând toate valorile obținute pe parcurs vom obține numărul cuvintelor care se află înaintea cuvântului dat. Adunând 1 la această valoare, vom obține numărul de ordine al cuvântului.

Analiza complexității

Pentru a analiza complexitatea acestui algoritm va trebui să precizăm faptul că numărul literelor din alfabet este

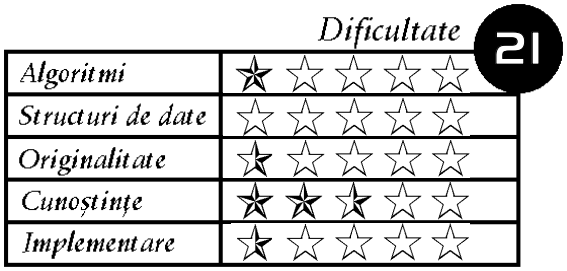
constant, deci nu poate interveni în exprimarea ordinului de complexitate. Acesta va fi stabilit doar în funcție de lungimea k a cuvântului.

Inițial se calculează suma $\sum_{i=1}^{k} C_{26}^{i}$, operație realizabilă în timp liniar având în vedere observația anterioară. Așadar, primul pas al algoritmului are ordinul de complexitate $O(k)$.

Pentru fiecare literă a cuvântului se calculează suma $\sum_{i=n+1}^{m-1} C_{26-i}^{k-2}$, unde variabilele au semnificația prezentată anterior. Numărul de litere este implicat în determinarea valorii combinării. Așadar, calculul combinării se realizează în timp liniar. Numărul de combinări calculate nu depinde de lungimea cuvântului, deci ordinul de complexitate al calculării acestei sume este $O(k)$. În total vor fi calculate k astfel de sume, deci ordinul de complexitate al celui de-al doilea pas al algoritmului este $O(k) \cdot O(k) = O(k^2)$.

Citirea datelor de intrare și scrierea celor de ieșire se realizează cu ajutorul unor operații elementare, deci putem considera că au ordinul de complexitate $O(1)$.

În concluzie, ordinul de complexitate al algoritmului de rezolvare a acestei probleme este $O(k^2) + O(k) + O(k) = O(k^2)$ având în vedere că numărul literelor din alfabetul englez este constant.



www.ginfo.ro/revista/12_6/surse/cod.cpp

P050207: Hotel

Pentru început, vom determina numărul de etaje la care există angajați și vom atribui fiecărui etaj câte o culoare. Pe măsură ce citim datele referitoare la angajați, vom verifica dacă etajul la care lucrează angajatul are asociată o culoare și, dacă este cazul, îi vom atribui o culoare și vom crește numărul etajelor distincte. Dacă ajungem în situația în care numărul culorilor disponibile este mai mic decât numărul etajelor distincte, atunci problema nu are soluție. Atribuind câte o culoare fiecărui etaj, practic, am rezolvat cea de-a doua parte a problemei.

Pentru prima parte vom considera că avem la dispoziție n culori și există k etaje distincte la care lucrează angajați. Va trebui să determinăm numărul de posibilități de a atribui celor k etaje câte o culoare astfel încât fiecare etaj să aibă propria sa culoare.

Practic, având la dispoziție o mulțime formată din n elemente va trebui să determinăm câte posibilități de alegere a k dintre aceste elemente există. Este foarte ușor de observat că, de fapt, propoziția anterioară reprezintă definiția noțiunii matematice de ***aranjamente***. Așadar, soluția primei părți a problemei este dată de formula:

$$A_n^k = \frac{n!}{(n-k)!} = (n-k+1)\cdot(n-k+2)\cdot\cdots\cdot(n-1)\cdot n.$$

Datorită faptului că se obțin numere foarte mari, nu se pot folosi tipurile de date puse la dispoziție de limbajele de programare pentru operații aritmetice. Se observă că, folosind formula anterioară, avem nevoie doar de înmulțiri dintre un număr mare și un număr cel mult egal cu 200. Ca urmare, va trebui să implementăm operația de înmulțire a unui număr mare cu un scalar. Pentru a mări viteza de execuție a programului și a utiliza mai eficient memoria, se poate folosi baza 10000 pentru efectuarea calculelor.

Analiza complexității

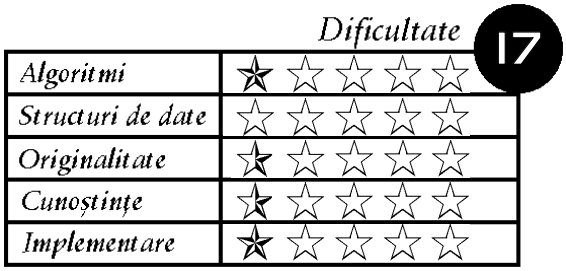
Citirea datelor de intrare se realizează în timp liniar, deci ordinul de complexitate al acestei operații este $O(n)$.

Verificarea faptului că unui etaj îi corespunde o culoare și eventuala atribuire a unei culori se realizează în timp constant. Deoarece această operație se realizează pentru etajul corespunzător fiecărui angajat, stabilirea corespondenței dintre etaje și culori se realizează într-un timp cu ordinul de complexitate $O(n) \cdot O(1) = O(n)$.

Calcularea numărului de modalități de alegere a culorilor implică folosirea numerelor mari. Se observă că numărul de cifre (în baza 10000) al rezultatului este mai mic decât 100, așadar putem considera că o înmulțire se realizează în timp liniar. Numărul înmulțirilor efectuate este k, deci ordinul de complexitate al operației de determinare a rezultatului este $O(k)$.

Afișarea numărului de modalități este realizată în timp constant, iar operația de afișare a culorilor corespunzătoare fiecărui angajat are ordinul de complexitate $O(n)$.

În concluzie, ordinul de complexitate al algoritmului de rezolvare a acestei probleme este $O(n) + O(n) + O(k) + O(1) + O(n) = O(k + n)$.



www.ginfo.ro/revista/12_6/surse/hotel.cpp

P050208: Lac

Pentru rezolvarea acestei probleme vom folosi o variantă puțin modificată a algoritmului lui ***Lee***. Pentru fiecare poziție a matricei care reprezintă o zonă mlăștinoasă vom determina numărul minim de pontoane care trebuie să fie amplasate pentru a se ajunge în poziția respectivă.

Pentru aceasta vom considera că pentru a ajunge în imediata vecinătate a primei linii a matricei numărul de pontoane necesare este 0. Pentru o anumită poziție, numă-

rul de pontoane necesare ajungerii în punctul respectiv este dat de cel mai mic număr corespunzător uneia dintre pozițiile învecinate la care se adaugă, eventual, un ponton dacă poziția nu corespunde unei porțiuni de uscat.

După o astfel de parcurgere a matricei, vom avea pentru fiecare poziție a matricei o anumită valoare, dar nu suntem siguri că aceasta este cea minimă. Vom parcurge succesiv matricea încercând să îmbunătățim valorile obținute. O valoare va fi modificată dacă se poate ajunge în poziția curentă dintr-o poziție învecinată și se obține un număr mai mic de pontoane pentru poziția curentă. În momentul în care nu va mai exista nici o parcurgere care să aducă îmbunătățiri, am obținut rezultatul final.

Practic cele două tipuri de parcurgeri pot fi "asimilate" în una singură dacă, inițial, se marchează toate pozițiile matricei cu o valoare suficient de mare. De fiecare dată când are loc o îmbunătățire, vom păstra direcția din care s-a ajuns în poziția curentă pentru a putea reconstitui drumul.

În final, numărul minim de pontoane va fi dat de cea mai mică valoare de pe ultima linie a matricei. Pentru reconstituirea drumului, se va porni în sens invers, de pe poziția de pe ultima linie în poziția din care s-a ajuns în ea și așa mai departe, până se ajunge pe prima linie. Dacă se ajunge într-o poziție care nu corespunde unei zone de uscat, atunci va trebui amplasat un ponton în acea poziție.

Analiza complexității

Citirea datelor de intrare implică o traversare a matricei, ordinul de complexitate al acestei operații fiind $O(m \cdot n)$.

O parcurgere a matricei în vederea realizării unei îmbunătățiri are același ordin de compelxitate. Datorită faptului că, după prima parcurgere, suntem siguri că se poate ajunge pe ultima linie folosind m pontoane (completarea integrală a coloanei), nu se vor realiza în nici o situație mai mult de m parcurgeri de îmbunătățire. Din aceste motive, ordinul de complexitate al operației de determinare a numărului de pontoane necesare pentru ajungerea în fiecare poziție este $O(m) \cdot O(m \cdot n) = O(m^2 \cdot n)$.

Determinarea valorii minime de pe ultima linie se realizează în timp liniar, ordinul de complexitate fiind $O(n)$.

Reconstituirea drumului ar putea, teoretic, să necesite traversarea întregii matrice. Astfel, în cel mai defavorabil caz, ordinul de complexitate al acestei operații este $O(m \cdot n)$.

În concluzie, ordinul de complexitate al algoritmului de rezolvare a acestei probleme este $O(m \cdot n) + O(m^2 \cdot n) + O(n) + O(m \cdot n) = O(m^2 \cdot n)$.

Dificultate 33	
Algoritmi	★★★☆☆
Structuri de date	★☆☆☆☆
Originalitate	★★☆☆☆
Cunoștințe	★★☆☆☆
Implementare	★☆☆☆☆

www.ginfo.ro/revista/12_6/surse/lac.cpp

P050209: Logic

Pentru început vom observa că, pentru ca trasarea să fie posibilă, fiecare zonă trebuie să fie delimitată de un număr par de segmente. Aceasta se datorează faptului că linia trebuie să intre și să iasă din fiecare zonă de un anumit număr de ori.

Așadar, pentru fiecare zonă va trebui să numărăm segmentele (așa cum sunt definite ele în enunț) care o delimitează și să verificăm dacă numărul obținut este par sau impar. În momentul în care găsim o zonă delimitată de un număr par de segmente, vom ști că trasarea nu este posibilă. Dacă toate zonele sunt delimitate de un număr par de segmente, atunci trasarea este posibilă.

Numărarea segmentelor care delimitează o zonă se realizează pe baza unui algoritm de umplere. În momentul în care ajungem la un punct de la marginea zonei, se pune problema creșterii numărului de segmente. Există mai multe situații în care numărul de segmente va crește:

- se ajunge în partea de sus a zonei, iar punctul aflat imediat deasupra și cel din dreapta acestuia fac parte din zone diferite;
- se ajunge în partea de jos a zonei, iar punctul aflat imediat dedesubt și cel din dreapta acestuia fac parte din zone diferite;
- se ajunge în partea din stânga a zonei, iar punctul aflat imediat la stânga și cel de deasupra sa fac parte din zone diferite;
- se ajunge în partea din dreapta a zonei, iar punctul aflat imediat la dreapta și cel de deasupra sa fac parte din zone diferite;
- se ajunge într-unul din colțurile zonei.

Dacă ultima condiție este îndeplinită simultan cu una dintre primele patru, numărul segmentelor va crește doar cu 1. Creșterile datorate primelor patru condiții sunt cumulative.

Analiza complexității

Vom studia acum complexitatea algoritmului pentru un desen. Citirea datelor de intrare implică o traversare a matricei, deci această operație se realizează într-un timp de ordinul $O(n^2)$.

La fiecare pas al algoritmului de umplere se numără segmentele care trebuie adăugate pentru poziția curentă. Se pot adăuga cel mult patru segmente, așadar ordinul de complexitate al operației este $O(1)$. Algoritmul de umplere implică parcurgerea integrală a matricei, motiv pentru care vor fi vizitate toate cele n^2 poziții. Așadar, algoritmul de verificare a posibilității de trasare a liniei are ordinul de complexitate $O(n^2) \cdot O(1) = O(n^2)$.

Datele de ieșire constau din scrierea unui singur mesaj, deci această operație se efectuează în timp *liniar*.

În concluzie, algoritmul de rezolvare a acestei probleme, pentru un desen, are ordinul de complexitate $O(n^2) + O(n^2) + O(1) = O(n^2)$.

Pentru a stabili ordinul de complexitate al algoritmului de rezolvare a întregii probleme, vom considera că un set

de date cuprinde t desene. Așadar, ordinul de complexitate este $O(t) \cdot O(n^2) = O(n^2 \cdot t)$.

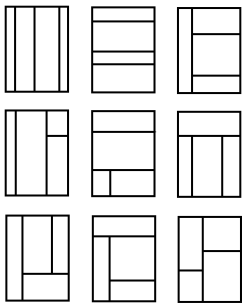
Dificultate 38

Algoritmi	★★☆☆☆
Structuri de date	★☆☆☆☆
Originalitate	★★★☆☆
Cunoștințe	★★★☆☆
Implementare	★☆☆☆☆

www.ginfo.ro/revista/12_6/surse/logic.cpp

P050210: Foto

Se observă foarte ușor că există doar nouă posibilități de lipire a patru fotografii pe o pagină, indiferent de dimensiunile acestora. Toate celelalte posibilități sunt echivalente cu una dintre cele nouă, ele obținându-se prin translatări sau oglindiri. Cele nouă aranjamente posibile pe care le vom lua în considerare sunt prezentate în figura alăturată.



Modalități de amplasare

Va trebui să alegem pozițiile celor patru fotografii pentru fiecare dintre cele nouă configurații. Teoretic, pentru fiecare dintre cele nouă configurații, există 4! = 24 posibilități. Practic, se observă că există și în acest caz configurații echivalente. De exemplu, pentru prima și a doua configurație, nu contează ordinea în care sunt așezate fotografiile, pentru a treia configurație nu contează ordinea în care sunt dispuse cele trei poze de pe a doua coloană etc. Practic, pentru fiecare configurație vom avea 1, 2, 4, 6 sau 12 posibilități.

Cu excepția ultimei configurații, în toate celelalte există o fotografie care ocupă o întreagă latură a paginii. Cunoscând raportul dintre lungimea și lățimea sa se determină laturile spațiului liber rămas pe foaie. În continuare, una dintre laturile spațiului este ocupat integral de o fotografie. Procedura continuă până determinăm dimensiunile tuturor fotografiilor. În final se verifică dacă foaia este acoperită integral. Pentru ultima configurație vom alege, pe rând, dimensiunile posibile ale fotografiei din colțul din stânga sus și dimensiunile celorlalte fotografii vor fi determinate în același mod ca și în situațiile anterioare.

Analiza complexității

Citirea datelor se realizează în timp ***constant***, deoarece numărul fotografiilor este întotdeauna 4.

Studierea primelor opt configurații se realizează de asemenea în timp constant deoarece numărul posibilităților de amplasare a fotografiilor respectând configurațiile considerate este constant.

Pentru cea de-a noua configurație putem avea cel mult $\min(X, Y)$ dimensiuni posibile ale primei fotografii, unde X și Y sunt dimensiunile paginii. Dacă notăm cu n acest minim, ordinul de complexitate al acestei operații va fi $O(n)$.

Datele de ieșire constau în exact opt numere, deci scrierea rezultatelor se realizează în timp ***constant***.

În concluzie, algoritmul de rezolvare a acestei probleme are ordinul de complexitate $O(1) + 8 \cdot O(1) + O(n) + O(1) = O(n)$.

Dificultate 30

Algoritmi	★★☆☆☆
Structuri de date	★☆☆☆☆
Originalitate	★★☆☆☆
Cunoștințe	★★☆☆☆
Implementare	★★★☆☆

www.ginfo.ro/revista/12_6/surse/foto.cpp

P050211: Balanță

Se observă că suma momentelor greutăților este cuprinsă între -6000 și 6000 (dacă avem 20 de greutăți cu valoarea 20 și acestea sunt amplasate pe cel mai îndepărtat cârlig față de centrul balanței, atunci modulul sumei momentelor forțelor este $15 \cdot 20 \cdot 20 = 6000$). Ca urmare, putem păstra un șir a ale cărui valori a_i vor conține numărul posibilităților ca suma momentelor greutăților să fie i. Indicii șirului vor varia între -6000 și 6000. Pentru a îmbunătăți viteza de execuție a programului, indicii vor varia între $-300 \cdot g$ și $300 \cdot g$, unde g este numărul greutăților. Pot fi realizate îmbunătățiri suplimentare dacă se determină distanțele maxime față de mijlocul balanței ale celor mai îndepărtate cârlige de pe cele două talere și suma totală a greutăților. Dacă distanțele sunt d_1 și d_2, iar suma este s, atunci indicii vor varia între $-d_1 \cdot s$ și $d_2 \cdot s$.

Inițial, pe balanță nu este agățată nici o greutate, așadar suma momentelor greutăților este 0. Ca urmare, inițial valorile a_i vor fi 0 pentru orice indice nenul și $a_0 = 1$ (există o posibilitate ca inițial suma momentelor greutăților să fie 0 și nu există nici o posibilitate ca ea să fie diferită de 0).

În continuare, vom încerca să amplasăm greutățile pe cârlige. Fiecare greutate poate fi amplasată pe oricare dintre cârlige. Să presupunem că la un moment dat există a_i posibilități de a obține suma i. Dacă vom amplasa o greutate de valoare g pe un cârlig aflat la distanța d față de centrul balanței, suma momentelor greutăților va crește sau va scădea cu $g \cdot d$ (în funcție de brațul pe care se află cârligul). Ca urmare, după amplasarea noii greutăți există a_i posibilități de a obține suma $i + g \cdot d$. Considerăm că șirul b va conține valori care reprezintă numărul posibilităților de a obține sume ale momentelor forțelor după amplasarea greutății curente. Înainte de a testa posibilitățile de plasare a greutății, șirul b va conține doar zerouri. Pentru fiecare pereche (i, d), valoarea $b_{i+g \cdot d}$ va crește cu a_i. După considerarea tuturor perechilor, vom putea trece la o nouă greutate. Valorile din șirul b vor fi salvate în șirul a, iar șirul b

va fi reinițializat cu 0. Rezultatul final va fi dat de valoarea a_0 obținută după considerarea tuturor greutăților disponibile.

Analiza complexității

Pentru studiul complexității vom nota numărul greutăților cu g, iar cel al cârligelor cu c.

Citirea datelor de intrare corespunzătoare cârligelor și greutaților se realizează în timp liniar, deci ordinul de complexitate al acestor operații este $O(g)$, respectiv $O(c)$.

Singura mărime care nu este considerată constantă și de care depinde numărul de posibilități de a obține sumele este numărul greutăților. Așadar, vom considera că ordinul de complexitate al traversării șirului a este $O(g)$. Numărul de traversări este dat tot de numărul greutăților disponibile, deci vom avea $O(g)$ traversări. În timpul traversării vom considera toate cârligele pentru fiecare element al șirului. Ca urmare, ordinul de complexitate al operațiilor efectuate asupra unui element într-o parcurgere este $O(c)$. Rezultă că ordinul de complexitate al unei parcurgeri este $O(g) \cdot O(c) = O(g \cdot c)$, în timp ce ordinul de complexitate al întregii operații care duce la obținerea rezultatului este $O(g) \cdot O(g \cdot c) = O(g^2 \cdot c)$.

Afișarea numărului de posibilități de a echilibra balanța se realizează în timp *constant*.

În concluzie, ordinul de complexitate al algoritmului de rezolvare a acestei probleme este $O(c) + O(g) + O(g^2 \cdot c) + O(1) = O(g^2 \cdot c)$.

Dificultate 23	
Algoritmi	★★☆☆☆
Structuri de date	★☆☆☆☆
Originalitate	★★☆☆☆
Cunoștințe	★★☆☆☆
Implementare	★☆☆☆☆

www.ginfo.ro/revista/12_6/surse/balanta.cpp

P050212: Aliniere

Pentru fiecare soldat vom determina cel mai lung subșir strict crescător (din punct de vedere al înălțimii) de soldați care se termină cu el, respectiv cel mai lung subșir strict descrescător de soldați care urmează după el. După această operație, vom determina soldatul pentru care suma lungimilor celor două șiruri este maximă.

Chiar dacă s-ar părea că în acest mod am găsit soluția problemei, mai există o posibilitate de a mări numărul soldaților care rămân în șir. Să considerăm soldatul cel mai înalt în șirul rămas (cel căruia îi corespunde suma maximă). Acesta poate privi fie spre stânga, fie spre dreapta șirului. Din aceste motive, la stânga sau la dreapta sa poate să se afle un soldat de aceeași înălțime; unul dintre cei doi va privi spre dreapta, iar celălalt spre stânga. Totuși, nu putem alege orice soldat cu aceeași înălțime, ci doar unul pentru care lungimea șirului strict crescător (dacă se află spre stânga) sau a celui strict descrescător (dacă se află spre dreapta) este aceeași cu lungimea corespunzătoare șirului strict crescător, respectiv strict descrescător, corespunzătoare celui mai înalt soldat dintre cei rămași în șir.

După identificarea celor doi soldați de înălțimi egale (sau demonstrarea faptului că nu există o pereche de acest gen care să respecte condițiile date) se marchează toți soldații din cele două subșiruri. Ceilalți soldați vor trebui să părăsească formația.

Analiza complexității

Citirea datelor de intrare se realizează în timp liniar, deci ordinul de complexitate al acestei operații este $O(n)$.

Chiar dacă există algoritmi eficienți (care rulează în timp liniar-logaritmic) de determinare a celui mai lung subșir ordonat, timpul de execuție admis ne permite folosirea unui algoritm simplu, cu ordinul de complexitate $O(n^2)$. Acesta va fi aplicat de două ori, după care se va căuta valoarea maximă a sumei lungimilor șirurilor corespunzătoare unui soldat; așadar identificarea soldatului care poate privi în ambele direcții este o operație cu ordinul de complexitate $O(n^2) + O(n^2) + O(n) = O(n^2)$.

Urmează eventuala identificare a unui alt soldat de aceeași înălțime care respectă condițiile referitoare la lungimile subșirurilor. Pentru aceasta se parcurge șirul soldaților de la soldatul identificat anterior spre extremități; operația necesită un timp *liniar*.

Deteminarea celor două subșiruri se realizează în timp *liniar* dacă, în momentul construirii celor două subșiruri, se păstrează predecesorul, respectiv succesorul fiecărui soldat. În timpul parcurgerii subșirurilor sunt marcați soldații care rămân în formație.

Pentru scrierea datelor de ieșire se parcurge șirul marcajelor și sunt identificați soldații care părăsesc formația. Ordinul de complexitate al acestei operații este $O(n)$.

În concluzie, ordinul de complexitate al algoritmului de rezolvare a acestei probleme este $O(n) + O(n^2) + O(n) + O(n) + O(n) = O(n^2)$.

Dificultate 33	
Algoritmi	★★☆☆☆
Structuri de date	★☆☆☆☆
Originalitate	★★☆☆☆
Cunoștințe	★★★☆☆
Implementare	★☆☆☆☆

www.ginfo.ro/revista/12_6/surse/aliniere.cpp

ONI 2002 BRĂILA