Olimpiada Societății pentru Excelență și Performanță în Informatică, Etapa Județeană
Descrierea Soluțiilor
Clasa a X-a

1 Problema Labirint

Propunător: prof. Gheorghe Manolache, stud. Ștefan-Cosmin Dăscălescu

Mai întâi vom afla, folosind algoritmul lui Lee, distanța minimă de la $(1,1)$ la (N,M), notată în enunț cu d_0. Acum, diferența dintre soluția optimă și soluția parțială va consta în modul în care verificăm pentru fiecare poziție egală cu 1 dacă distanța se modifică.

Soluție pentru 10 puncte Pentru primul grup de teste, deoarece $d_0 = N + M - 1$, se poate observa că nu va exista nicio zonă care să îmbunătățească răspunsul, deci se poate obține punctajul pe aceste teste afișând o matrice numai cu valori egale cu 0.

Soluție pentru 40 de puncte Pentru cel de-al doilea grup de teste, se poate simula cu ajutorul algoritmului lui Lee înlocuirea fiecărei valori egale cu 1 și se va verifica dacă distanța de la $(1,1)$ la (N,M) s-a micșorat, afișându-se 1 sau 0, după caz, complexitatea acestui algoritm fiind $O(N^2M^2)$. De notat că această soluție rezolvă corect și testele din primul grup 1.

Soluție pentru 100 de puncte Pentru a ajunge la soluția optimă, se va observa faptul că nu e nevoie să rulăm algoritmul lui Lee de fiecare dată când modificăm valoarea unei poziții, fiind de ajuns precalcularea distanțelor atât din $(1,1)$ cât și din (N,M) către toate celelalte poziții din matrice. Astfel, pentru fiecare zonă (i,j) în care se poate ajunge atât din $(1,1)$, cât și din (N,M), distanța pe care o vom avea de la $(1,1)$ la (N,M) trecând printr-o poziție (i,j) egală inițial cu 1 va fi egală cu următoarea valoare: $d(i,j) = dist_1(i,j) + dist_2(i,j) - 1$, unde $dist_1(i,j)$ reprezintă distanța de la $(1,1)$ la (i,j), iar $dist_2(i,j)$ reprezintă distanța de la (N,M) la (i,j), fiind necesară scăderea lui 1 deoarece poziția (i,j) apare în ambele distanțe.

Complexitatea algoritmului pentru soluția ce obține 100 de puncte devine astfel $O(NM)$.

2 Problema SDistanțe

Propunător: stud. Bogdan-Petru Pop

Vom nota cu N lungimea șirului s.

Solutie în $O(N^4)$ - 11-16 puncte în funcție de implementare Pentru această soluție, ajunge să găsim un algoritm simplu de calcul al distanței între două șiruri și să îl aplicăm pe toate perchile de subsecvențe. Observăm că $dist(a, b)$ este numărul de poziții i pentru care $a(i)$ este diferit de $b(i)$ $(0 \leq i < |a|)$. Cu această observație, obținem un algoritm liniar care va fi aplicat pe $O(N^3)$ perechi de stringuri (toate perechile $(i, i + lungime), (j, j + lungime)$, $0 < i < j < |a|, i + lungime < |a|$), obținând un algoritm de complexitate $O(N^4)$.

Algorithm 1 Soluție brute-force

$N \leftarrow lungime(s)$
$raspuns \leftarrow 0$
for $i \leftarrow 0...N-1$ **do**
  for $j \leftarrow i+1..N-1$ **do**
    for $lungime \leftarrow 1..N-j$ **do**
      for $indice \leftarrow 0..lungime$ **do**
        if $s(i+indice) \neq s(j+indice)$ **then** $raspuns \leftarrow raspuns + 1 \bmod 1.000.000.007$

Solutie in $O(N^3)$ - 31 de puncte Încercăm să optimizăm soluția anterioară. Observăm că un triplet $(i, j, indice)$ contribuie de mai multe ori la răspuns. Mai exact, observăm că dacă $s(i + indice) \neq s(j + indice)$, atunci adunăm 1 la răspuns pentru toate valorile lui $lungime$ mai mari sau egale cu indice. Astfel, putem renunța la a itera cu variabila lungime, iar în locul acestei iterații să adunăm la răspuns numărul de valori ale lui $indice$ pentru care se adună 1 la verificarea $s(i + indice) \neq s(j + indice)$. Calculând exact, vom adăuga la răspuns $N - (j + indice)$ pentru fiecare triplet care verifică proprietatea anterioară. Obținem astfel un algoritm de complexitate $O(N^3)$.

Solutie in $O(N^2)$ - 67 de puncte Asemănător cu pasul anterior, vom încerca să fixăm o anumită pereche de indici cu caractere diferite și să observăm cu cât contribuie la rezultat. Pentru asta, vom rescrie în funcție de pozițiile pe care le comparăm condițiile ca restul variabilelor să fie valide. Notăm cu a și b, $a < b$, pozițiile pe care le comparăm la fiecare pas și scriem condițiile pentru ca $i, j, lungime, indice$ sa fie valide.

$$\begin{cases} a = i + indice \\ b = j + indice \\ i \geq 0 \\ j + lungime < N \end{cases}$$

Vom rescrie indicii pentru indexa în funcție de pozițiile pe care le comparăm.

$$\begin{cases} i = a - indice \\ j = b - indice \\ a - indice \geq 0 \\ N - lungime < b - indice \end{cases}$$

Observați ca i și j sunt unic determinați dacă știm toate valorile $a, b, lungime, indice$, deci nu este necesar să păstram primele două ecuații pentru a verifica validitatea unei soluții. Astfel, dacă avem a și b fixat, condițiile pe care trebuie să le îndeplinească $lungime$ și $indice$ sunt pentru a fi valide sunt:

$$\begin{cases} indice \leq a \\ lungime - indice < N - b \end{cases}$$

Acest sistem are $(a+1) * (N-b)$ soluții care sunt perechi de numere naturale (orice valoare de la 0 la a este valabilă pentru $indice$, iar pentru $lungime - indice$ putem alege orice valoare de la 0 la $N - b - 1$, determinând unic o valoare validă a variabilei $lungime$ pentru orice valoare fixată a variabilei $indice$). Astfel, perechea (a, b) contribuie cu $(a+1) * (N-b)$ la răspuns.

Solutie in $O(N * sigma)$ unde $sigma$ este mărimea alfabetului - 82 de puncte Vom fixa doar poziția b și vom încerca să găsim contribuția acesteia la răspuns. Aceasta va fi $(a_1+1) * (N-b) + (a_2+1) * (N-b) + .. + (a_k+1) * (N-b)$, unde $a_1, a_2, ...a_k$ sunt indicii pentru care $s(b) \neq s(a_i)$ și $a_i < b$, Dacă dăm factor comun $(N-b)$, obținem:

$$[(a_1+1) + (a_2+1) + ... + (a_k+1))] * (N-b)$$

Astfel, problema se reduce la calculul eficient al sumei $(a_1+1) + (a_2+1) + \ldots + (a_k+1)$. Pentru a obține suma acestor indici, ne vom folosi de proprietatea lor: conțin o altă valoare decât $s(b)$. Putem ține un vector de sume $sum(c)$ care conține suma de $i+1$ pentru indicii i mai mici decât b-ul curent pentru care $s(i) = c$. Acest vector poate fi actualizat în $O(1)$ la fiecare pas, adăugând $b+1$ la $sum(s(b))$. Atunci când vrem să obținem suma a-urilor din expresia de mai sus, vom însuma pur și simplu toate valorile $sum(c), c \neq s(b)$, ceea ce poate fi realizat în $O(sigma)$.

Solutie in $O(N)$ - 100 de puncte Observăm că suma $a_1+a_2+\ldots+a_k$ este de fapt $1+2+3+\ldots+(b-1)-(b_1+b_2+\ldots+b_l)$, unde $b_1, b_2, \ldots, b_l$ sunt toate pozițiile cu proprietatea $b_i < b, s(b_i) = s(b)$ (practic putem să scădem din suma tuturor pozițiilor anterioare suma pozițiilor cu un caracter egal cu $s(b)$ și rămâne suma celor care nu sunt egale cu b. Analog $(a_1+1) + (a_2+1) + ... + (a_k+1) = (0+1) + (1+1) + (2+1) + \ldots + ((b-1)+1) - ((b_1+1) + (b_2+1) + \ldots + (b_l+1))$. Suma anterioară poate fi rescrisă, folosind vectorul sum, ca:

$$\frac{b * (b+1)}{2} - sum(b)$$

Această expresie poate fi calculată în $O(1)$, fapt ce ne duce la complexitatea finală $O(N)$.

Alternativ, putem calcula de la început cât ar fi răspunsul dacă toate caracterele din șir ar fi diferite, iar apoi să scădem numărul de "potriviri" (caractere egale pe aceeași poziție în 2 șiruri) între perechile de subsecvențe, cu o implementare asemănătoare ce folosește același vector sum.

3 Problema Tort

Propunător: prof. Marius Nicoli

Soluție pentru $N \leq 20$. O abordare prin care se generează toate modurile de a descompune în secvențe șirul dat obține punctele la aceste teste. De exemplu, se pot genera toate șirurile de 0 și de 1 care încep cu 1, mai conțin încă un 1 cel puțin și care au lungimea n. Pozițiile pe care se află valoarea 1 sunt începuturile de secvențe ale unei descompuneri. Timpul de executare este de ordinul $2^n * n$.

Soluție pentru $N \leq 100, a_1 = \ldots = a_N = 1$. Toate elementele fiind egale cu 1, odată ce fixăm prima secvență între indicii 1 și i, la soluție trebuie adunat numărul de divizori ai valorii $n - i$ (suma numerelor de la poziția i până la poziția n). Timpul de executare este deci $n\sqrt{n}$ dar se poate obține unul mai bun dacă ne gândim că este vorba de suma divizorilor pentru fiecare număr de la 1 la $n - 1$, iar aceste valori se pot calcula folosind ciurul lui Eratostene.

Soluție pentru $N \leq 100$. Fixăm de asemenea prima secvență iar pentru restul descompunerii fixăm mai întâi suma comună pe care o dorim în restul secvențelor și apoi facem verificarea dacă descompunerea cu elementele fixate înainte este posibilă. Timpul de calcul este de ordinul $n^2 \times$ (*suma valorilor din șirul dat*).

Soluție pentru $N \leq 2000, a_1 + \ldots + a_N \leq 4000$. Odată ce fixăm prima secvență (până la poziția $i-1$) ne dăm seama că suma comună din celelalte secvențe nu poate fi decât un divizor al valorii S_i ($S_i =$ suma elementelor de la poziția i până la poziția n). Astfel este necesară verificarea doar pentru valorile acestor divizori. Avem timp de ordinul n^2 de la fixarea secvenței inițiale și de la verificare și se adaugă costul obținerii divizorilor lui S_i (fie obținem divizorii la întâlnirea elementului curent cu timp de ordin $\sqrt{S_i}$, fie îi precalculăm folosind Ciurul lui Eratostene).

Soluție pentru 100 de puncte. Facem o parcurgere de la final și acumulăm la o sumă s valorile din șir pe măsură ce le întâlnim, marcând într-un vector de frecvență în dreptul sumelor obținute. Pe acest vector, pentru valori naturale i, începând cu 1, verificăm cât putem merge plecând de la indicele i și avansând din i în i pe elemente marcate, fiecare pas făcut reprezentând de altfel o soluție. Este de fapt o simulare a algoritmului de la Ciurul lui Eratostene, aceasta fiind și cel care dă complexitatea în timp a unei soluții care se încadrează în timp pe toate testele.

Echipa. Setul de probleme pentru această rundă a fost propus și pregătit de:

- stud. doctorand Lucian Bicsi, Universitatea din București,
- prof. Mihai Bunget, Colegiul Naţional "Tudor Vladimirescu" Târgu Jiu,
- stud. Ștefan-Cosmin Dăscălescu, Universitatea din București,
- prof. Gheorghe Manolache, Colegiul Național de Informatică Piatra Neamț,
- stud. masterand Tamio-Vesa Nakajima, Universitatea Oxford,
- prof. Marius Nicoli, Colegiul Național "Frații Buzești" Craiova,
- prof. Eugen Nodea, Colegiul Naţional "Tudor Vladimirescu" Târgu Jiu,
- stud. Costin-Andrei Oncescu, Universitatea Oxford,
- stud. Bogdan-Petru Pop, Universitatea "Babeș-Bolyai" Cluj Napoca,
- conf.univ.dr. Doru Anastasiu Popescu, Universitatea din Pitești, Departamentul de Matematica-Informatica,
- stud. Mihai-Cristian Popescu, Universitatea Babeș-Bolyai Cluj Napoca.