TABĂRA DE PREGĂTIRE A LOTULUI NAȚIONAL DE INFORMATICĂ
CLUJ, 10-15 MAI 2024
BARAJ 1
DESCRIEREA SOLUȚIILOR

COMISIA ȘTIINȚIFICĂ

Problema 1: Bug

Propusă de: prof.Emanuela Cerchez, Colegiul Național "Emil Racoviță", Iași

Soluția 1 - prof. Emanuela Cerchez

Pasul 1. Extragem cifrele numărului X şi le plasăm în vectorul cx, de lungime lgx, începând cu unităţile la poziţia 1.

Pasul 2. Precalculăm următoarele valori:

- $s[i]$=numărul de numere de exact i cifre care nu conţin ca subsecvenţă pe X (unde i variază de la 1 la $LGMAX = 17$);
- $sum[i] = s[1] + s[2] + \cdots + s[i]$=numărul de numere de cel mult i cifre care nu conţin ca subsecvenţă pe X.

```
s[0]=1; s[1]=9;
for (i=2; i <= lgx; i++) s[i]=s[i-1]*10;
s[lgx]--;
for (i=lgx+1; i < LGMAX; i++)
      {s[i]=10*s[i-1]-s[i-lgx];}
sum[0]=1;
for (i=1; i < LGMAX; i++) sum[i]=sum[i-1]+s[i];
```

Pasul 3. Rezolvăm problema prin căutare binară pe rezultat.

```
st=0; dr=N*10+1;
while (dr-st > 1)
        {
         mij = (st + dr) / 2;
         nr = numar(mij);
         if (nr < N)
              st = mij;
              else
              dr=mij;
         }
```

Funcţia $numar(n)$ determină numărul de valori $\leq n$ care nu conţin ca subsecvenţă pe X.

Pasul 4. Funcţia $numar(n)$ extrage cifrele numărului n şi le plasează în vectorul $cifre$ de lungime lg; parcurge cifrele lui n începând cu cifra dominantă (de la dreapta către stânga) şi analizează dacă secvenţa de cifre de lungime lgx care începe la poziţia curentă este mai mare, mai mică sau egală cu X. În cazul în care este mai mică, pentru poziţia curentă orice cifră mai mică decât cea curentă va conduce la o soluţie corectă, deci actualizăm rezultatul astfel:

```
 rez=rez+cifre[i]*sum[i-1];
```

În cazul în care este mai mare, trebuie să analizăm următoarele lgx cifre şi pentru fiecare să actualizăm rezultatul adunând numărul de soluţii care se obţin cu cifre mai mici decât cifra curentă, dar să scădem soluţiile generate cu prefixul lui X care începe la poziţia curentă, astfel:

```
for (j=lgx; j>0; j--)
    {scad=egal(j-lgx+i);
     if (scad>0)
        {rez=rez+(cifre[j-lgx+i])* sum[j-lgx+i-1]-sum[j-lgx+i-lgx];}
        else
        rez=rez+cifre[j-lgx+i]* sum[j-lgx+i -1];
    }
    i=i-lgx+1;
```

În caz de egalitate procedăm similar ca în cazul mai mare, numai că încheiem numărarea la parcurgerea celor lgx cifre.

Pasul 5. Din păcate, căutarea binară pe rezultat va genera cel mai mic număr natural pentru care numărul de numere mai mici sau egale cu el care nu conţin ca subsecvenţă pe X este N, dar nu este garantat că acest număr nu va avea ca subsecvenţă pe X. Vom corecta, eventual, rezultatul obţinut în urma căutării binare, parcurgându-l începând cu unităţile şi, atunci când identificăm o apariţie a lui X, adunăm 1 la cifra unităţilor acestei apariţii a lui X.

Soluția 2 - prof. Adrian Panaete

Se calculează răspunsul cifră cu cifră. În acest scop se precalculează:

(1) $cnt[L][P]$, $1 \leq L \leq |N|+1$; $0 \leq P < |X|$, cu semnificația: câte numere valide au L cifre și au prefix de lungime P comun cu numărul invalid X.

Observație: Dacă un număr valid are prefix P atunci el se număra și pentru prefixele $P-1$, $P-2$, ..., 1, 0.

Astfel $cnt[L][0]$ poate fi interpretat drept numărul tuturor valorilor valide de L cifre.

(2) $CNT[L][P][C]$, $0 \leq C \leq 9$, cu semnificația: câte dintre cele $cnt[L][P]$ numere valide descrise mai sus se obțin prin adăugarea la prefixul de lungime P a cifrei C

Se observă ca valorile $cnt[L][P]$ depind doar de $|X|$, nu și de valoarea cifrelor lui X. Astfel se obțin următoarele formule de calcul:

- pentru $L < |X|$:
  - $cnt[L][L] = 1$
  - $cnt[L][P] = 10 \cdot cnt[L][P+1]$
- pentru $L \geq |X|$ și $P > 0$:
  - $cnt[L][P] = 8 \cdot cnt[L-P-1][0] + cnt[L-P][1] + cnt[L][P+1]$
- pentru $L \geq |X|$ și $P = 0$:
  - $cnt[L][0] = 9 \cdot cnt[L-1][0] + cnt[L][1]$

Termenul $8 \cdot cnt[L-P-1][0]$ se referă la situația când după prefixul de lungime P adăugăm o cifră care nu este nici prima cifră din X și nici cifra care prelungește prefixul de lungime P la prefixul de lungime $P+1$.

Termenul $cnt[L-P][1]$ se referă la situația când după prefixul de lungime P adăugăm prima cifră din X.

Termenul $cnt[L][P+1]$ se referă la situația când după prefixul de lungime P adaugăm a $(P+1)$-a cifră din X.

Termenul $9 \cdot cnt[L-1][0]$ se referă la situația când numărul de lungime L începe cu orice cifră cu excepția primei cifre din X.

Termenul $cnt[L][1]$ se referă la situația când numărul de lungime L începe cu prima cifră din X.

Valorile $CNT[L][P][C]$ reprezintă termenii din descompunerea lui $cnt[L][P]$ în funcție de cifra C care se va adăuga prefixului de lungime P, considerat deja cunoscut. Aceste valori depind de cifrele numărului X și se vor calcula în funcție de P astfel:

- dacă $cnt[L][P] = 0$:
  - $CNT[L][P][C] = 0$ pentru orice cifră C
- dacă $P = 0$:
  - $CNT[L][P][C] = cnt[L][1]$ dacă C este prima cifră din X

  - $CNT[L][P][C] = cnt[L-1][0]$ dacă C nu este prima cifră din X
- dacă $P = |X| - 1$:
  - $CNT[L][P][C] = 0$ dacă C este ultima cifră din X
  - $CNT[L][P][C] = cnt[L-P][1]$ dacă C este prima cifră din X
  - $CNT[L][P][C] = cnt[L-P-1][0]$ pentru celelalte cifre
- dacă $0 < P < |X| - 1$:
  - $CNT[L][P][C] = cnt[L][P+1]$ dacă C este cifra de pe poziția $P+1$ din X
  - $CNT[L][P][C] = cnt[L-P][1]$ dacă C este prima cifra din X
  - $CNT[L][P][C] = cnt[L-P-1][0]$ pentru celelalte cifre

Odată ce am făcut precalculările vom determina soluția cifră cu cifră începând cu cifra de ordin maxim folosind următoarea strategie:

Detectăm pe rând cele cel mult 18 cifre. Între aceste cifre pot avea eventual și cifre 0 nesemnificative.

Să notăm cu R numărul de cifre care au mai rămas de detectat.

Știm deja prefixul PR de lungime $18 - R$ al soluției și vrem să detectăm următoarea cifră.

Pentru PR știm și numărul de cifre P din sufixul lui PR și care sunt prefix pentru X.

Rezultatul va fi un număr situat între două valori:

(1) Cea inferioară $LO =$ cele $18 - R$ cifre determinate urmate de R cifre de 0
(2) Cea superioară $HI =$ cele $18 - R$ cifre determinate urmate de R cifre de 9

Pentru acest context știm:

(1) 1. $offset < n$, unde $offset =$ numărul de valori valide strict mai mici decât LO
(2) 2. numărul căutat are valoarea intre LO și HI

Se consideră pe rând cele 10 cifre în ordine crescătoare și prelungim prefixul cu fiecare.

Numerele valide intre LO și HI vor fi distribuite astfel:

- primele $NR0 = CNT[R+P][P][0]$ au prefixul PR continuat cu cifra 0
- următoarele $NR1 = CNT[R+P][P][1]$ au prefixul PR continuat cu cifra 1
- următoarele $NR2 = CNT[R+P][P][2]$ au prefixul PR continuat cu cifra 2
- ...
- ultimele $NR9 = CNT[R+P][P][9]$ au prefixul PR continuat cu cifra 9

Astfel putem poziționa valorile valide cu prefixul PR în funcție de cifra c pe care o adăugam la PR pe intervale $[STc, DRc]$:

- $c = 0 : [ST0, DR0]$, unde $ST0 = offset + 1$ și $DR0 = offset + NR0$
- $c = 1 : [ST1, DR1]$, unde $ST1 = DR0 + 1$ și $DR1 = DR0 + NR1$
- $c = 2 : [ST2, DR2]$, unde $ST2 = DR1 + 1$ și $DR2 = DR1 + NR2$
- ...
- $c = 9 : [ST9, DR9]$, unde $ST9 = DR8 + 1$ și $DR9 = DR8 + NR9$

Cifra corecta c este cea pentru care n este situat în intervalul $[STc, Drc]$. După detectarea fiecărei cifre c:

- Adaugăm cifra c la PR
- Actualizăm $offset$ la $STc - 1$
- Actualizăm valoarea lui P în funcție de cifra c adăugată.

Soluția 3 - prof. Cătălin Frîncu, Nerdvana București

În arhiva cu surse se află o sursă comentată denumită bug-automaton.cpp care implementează o variantă mai generală pentru problema bug, care funcționează și pentru cazul în care cifrele lui X nu sunt distincte.

Problema 2: NoOverlap

Propusă de: Prof. Dan Pracsiu, Liceul Teoretic "Emil Racoviță", Vaslui

Să presupunem că în matricea a de dimensiuni $5 \times n$ de mai jos am ales patru submatrici care sunt marcate cu 1, iar restul componentelor sunt zero:

$$a = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Observăm din acest exemplu că pe orice coloană submatricile 2×2 pot ocupa zero, două sau patru elemente. Avem deci următoarele posibilități de ocupare a unor elemente pe o coloană:

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

Mai sus, prima coloană (să o numim configurație de tip 0) este plină de 0, deci nu este utilizat niciun element de pe ea. A doua coloană (să o numim configurație de tip 1) are primele două elemente ocupate. Ultima coloană (o numim configurație de tip 7) are ocupate ultimele patru elemente.

În continuare, construim rezolvarea cu programare dinamică. Vom utiliza o matrice dp tridimensională $k \times n \times 8$, în care:

$dp[i][j][p]$ = suma maximă obținută din primele j coloane, s-au ales deja i submatrici 2×2, iar pe ultima linie este configurația dată de valoarea lui p (p=0..7 fiind tipul de coloană)

Relațiile de recurență

Cum calculăm $dp[i][j][0]$? Pentru că $p = 0$, atunci de pe coloana curentă j a matricii a nu alegem niciun element, deci $dp[i][j][0]$ va depinde de valorile obținute deja până pe coloana $j-1$, deci:

$dp[i][j][0] = \max(dp[i][j-1][p],\ p \in \{0,1,2,3,4,5,6,7\})$

Cum calculăm pe $dp[i][j][1]$? Pentru că $p = 1$, atunci înseamnă că primele două elemente de pe coloana j sunt ocupate de o jumătate de matrice 2×2, iar cealaltă jumătate de submatrice se află pe coloana anterioară $j-1$. Acest lucru înseamnă că această submatrice 2×2 ocupă componentele $a[0][j-1]$, $a[1][j-1]$, $a[0][j]$, $a[1][j]$, deci $dp[i][j][1]$ depinde de maximul dintre $dp[i-1][j-1][0]$, $dp[i-1][j-1][3]$, $dp[i-1][j-1][4]$, la care se adaugă suma submatricii 2×2.

Similar se determină și $dp[i][j][2]$, $dp[i][j][3]$, $dp[i][j][4]$.

Să vedem cum se determină $dp[i][j][5]$, când avem primele patru valori de pe coloana j ocupate de două submatrice 2×2, care ocupă în consecință și primele patru poziții de pe coloana $j-1$. Deci $dp[i][j][5]$ depinde doar de $dp[i-2][j-1][0]$ (pentru că pentru a putea alege cele două submatrici 2×2 este necesar să fie disponibilă coloana $j-1$), la care se adaugă suma elementelor din cele două submatrici 2×2.

Asemănător cu $dp[i][j][5]$ se tratează și $dp[i][j][6]$ și $dp[i][j][7]$.

Inițializarea matricei dp se poate face cu o valoare foarte mică, sau, crescând toate valorile din matricea inițială a cu 1000, dp se poate inițializa cu 0.

Soluția problemei va fi în $\max(dp[k][n][p],\ p \in \{0,1,2,3,4,5,6,7\})$

Complexitatea este $O(n \times k \times \Sigma)$, unde $\Sigma = 8$.

O soluție alternativă care nu obține decât punctaj parțial este un algoritm de tip Greedy. Luăm toate submatricile 2×2, fiecare submatrice fiind dată prin coordonatele colțului stânga-sus și suma elementelor submatricii. Sortăm descrescător submatricile după sumă, apoi luăm primele k submatrici care nu se suprapun. Când alegem o submatrice, într-o matrice marcăm cu 1 cele 2×2 poziții ocupate.

Soluția Greedy nu furnizează rezultatul corect în multe situații. De exemplu dacă avem $k = 4$ și matricea $5 \times n$ este de forma:

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 4 & 4 & 4 & 4 & 1 & 1 & 1 \\ 1 & 1 & 4 & 5 & 5 & 4 & 1 & 1 & 1 \\ 1 & 1 & 4 & 5 & 5 & 4 & 1 & 1 & 1 \\ 1 & 1 & 4 & 4 & 4 & 4 & 1 & 1 & 1 \end{bmatrix}$$

atunci dacă alegem submatricea formată din cele patru valori de 5, atunci nu mai putem alege cele patru submatrice care au câte trei de 4 și un 5.

Pentru 15 puncte se poate implementa un algoritm de tip backtracking.

O altă soluție de punctaj maxim fructifică observația că valoarea k este foarte mică față de valoarea n. Astfel, vom păstra din matricea inițială doar coloanele din care putem selecta cele k configurații care asigură alegerea sumei maxime. Vor fi selectate cel mult câte $2 \times k$ coloane pentru fiecare tip de configurație. Pentru a face această selecție se poate construi câte o listă ordonată descrescător care conține $n - 1$ perechi de forma (coloană, sumă configurație) din care selectăm primele $2 \times k$ perechi sau putem utiliza Σ cozi cu prioritate. Apoi reconstruim o parte din matricea inițială folosind doar coloanele selectate și pe noua construcție aplicăm algoritmul de programare dinamică prezentat anterior. Complexitatea va fi acum $O((n \times log + k^2) \times \Sigma)$

Problema 3: Pase

Propusă de: prof. Ciprian Cheșcă, Liceul Tehnologic "Grigore C. Moisil", Buzău

Cuvinte cheie:

- Programare dinamică
- Puncte laticiale
- Cel mai mare divizor comun
- Asemănarea figurilor geometrice
- Sume parțiale în matrice

Soluția I (sume parțiale în matrice) $O(L * C * min(L, C))$

• În prima parte a rezolvării problemei se analizează fiecare pasă și într-o matrice cu L linii și C coloane, pe care s-o denumim matricea paselor, având toate valorile inițiale nule, se marchează cu 1 toate punctele laticiale prin care trece o pasă. Pentru a rezolva această subproblemă se calculează numărul de puncte laticiale prin care trece o pasă cu expresia $np = cmmdc(|l2 - l1|, |c2 - c1|) + 1$, unde $(l1, c1)$ și $(l2, c2)$ sunt coordonatele între care se trimite pasa. Atenție că există și un caz particular și anume situația când pasa se dă „pe loc” adică $l1 = l2$ și $c1 = c2$. Apoi se determină punctele laticiale prin care trece pasa rezolvând mai multe cazuri diferite în funcție de așezarea punctelor $(l1, c1)$ și $(l2, c2)$, cu alte cuvinte direcția pasei poate fi către unul dintre cele patru puncte cardinale sau pe diagonale. Problema revine acum la a afla care este cea mai mare suprafață pătratică de elemente de zero din matricea paselor.

• În a doua etapă se calculează matricea sumelor parțiale a matricei paselor.

• În a treia etapă se parcurge matricea sumelor parțiale și pentru fiecare punct laticial de coordonate (l,c) se calculează suma elementelor dintr-o suprafață pătratică. Dacă această sumă este zero, adică zona pătratică este formată doar din elemente de 0, atunci se actualizează lungimea laturii maxime dacă este cazul. În funcție de implementare această etapă poate fi realizată în $O(LC)^2)$ sau $O(LCmin(L, C))$

• În ultima etapă se parcurge încă odată matricea sumelor parțiale, de această dată cunoscând lungimea laturii maxime se memorează într-o structură corespunzătoare toate zonele cu lungimea laturii maxime determinate anterior, ale cărei elemente, la final se afișează în fișierul de ieșire.

Observații:

• Are mare importanță cum se construiește matricea sumelor parțiale. Acest subalgoritm se poate realiza în $O(L^2C^2)$ sau $O(LC(L+C))$ sau $O(LC)$.

• O modalitate eficientă de construcție a acestei matrice se realizează pe baza valorilor din pozițiile $(i-1,j)$,$(i,j-1)$ și $(i-1,j-1)$ cât și a valorii din matricea inițială de pe poziția (i,j).

Soluția II (programare dinamică) O(L*C)

• La fel ca la soluția anterioară se analizează fiecare pasă și într-o matrice având L linii și C coloane, pe care s-o denumim matricea paselor (MP), având toate valorile inițiale egale cu 1, se marchează cu 0 toate punctele laticiale prin care trece o pasă. Se ține desigur cont de direcția din care vine pasa și de cazul particular al „paselor pe loc”.

Pentru a determina cea mai mare zonă pătratică având doar elemente de 1 se procedează astfel:

• Se utilizează o matrice auxiliară aux[L][C] care se construiește folosind algoritmul:

```
pentru i = 1, L
 pentru j = 1, C
  daca (MP[i][j] == 1) atunci
        aux[i][j] = min(MP[i][j-1], MP[i-1][j], MP[i-1][j-1]) + 1;
                                  altfel aux[i][j] = 0;
```

• Se determină maximul dintre elementele lui aux.

• Se parcurge încă odată matricea aux și cu maximul determinat anterior se rețin toate zonele pătratice de arie maxima și coordonate lor într-o structură de date, care ulterior se afișează.

Soluție alternativă - prof. Ionel Vasile Piț Rada

Se construiește matricea cu valoarea 1 la pozițiile pasabile și valoarea zero în rest. Se construiește matricea sumelor parțiale. Apoi se caută binar cea mai mare latură pentru care avem o zonă pătrată cu suma valorilor egală cu zero, adică să nu conțină poziții pasabile.

Echipa

Problemele pentru acest baraj au fost pregătite de:

- Prof. Emanuela Cerchez, Colegiul Național ”Emil Racoviță” Iași
- Prof. Adrian Panaete, Colegiul Naţional “August Treboniu Laurian”, Botoşani
- Prof. Ciprian Chesçă, Liceul Tehnologic ”Grigore C. Moisil”, Buzău
- Prof. Ionel-Vasile Piț-Rada, Colegiul Naţional “Traian”, Drobeta Turnu Severin
- Prof. Mihai Bunget, Colegiul Național ”Tudor Vladimirescu”, Târgu Jiu
- lector dr. Paul Diac, Facultatea de Informatică, Universitatea “Alexandru Ioan Cuza” Iaşi
- Prof. Dan Pracsiu, Liceul Teoretic “Emil Racoviță”, Vaslui
- Prof. Gheorghe Eugen Nodea, Centrul Județean de Excelență Gorj, Târgu Jiu
- Prof. Daniela Elena Lica, Centrul Județean de Excelență Prahova, Ploiești
- Stud. Ioan-Cristian Pop, Universitatea Politehnica București
- Stud. Dumitru Ilie, Facultatea de Matematică-Informatică, Universitatea București
- Stud. Giulian Buzatu, Facultatea de Matematică-Informatică, Universitatea București
- Stud. Mircea Măierean, Facultatea de Matematică și Informatică, Universitatea ”Babeș-Bolyai”, Cluj-Napoca