Soluţie ratina

Problema are mai multe solutii. Daca raspundem la fiecare intrebare parcurgand cuvintele pe toata lungimea lor pana la gasirea unei diferente (solutie de complexitate O(M*L*C) (L=lungimea maxima,C numarul de cuvinte din intrebare)) obtinem 30 de puncte. Daca preprocesam pentru oricare 2 cuvinte gradul de asemanare obtinem 50 de puncte.Pentru a obtine punctajul maxim este necesar sa raspundem mai repede la o astfel de intrebare.Solutia comisiei foloseste un arbore de prefixe pentru a retine dictionarul si totodata pentru fiecare caracter din cuvant ce nod din arbore ii corespunde. Avand aceste date preprocesate putem raspunde la o intrebare in O(C*logL) folosind cautarea binara.Mai multe cuvinte au prefixul de lungime k egala in cazul in care cuvintelor le corespunde acelasi nod din arbore pentru acea pozitie.

Un arbore de prefixe este un arbore in care fiecare nod are S fii unde S este dimesiunea alfabetului, in cazul nostru 26.Nodul radacina este special il putem nota cu caracterul # ,acesta are 26 de fii notati cu ’a’,’b’,…,’z’ acesti fii avand la randul lor alti 26 de fii .Cand introducem un cuvant in arbore incepem cu radacina si mergem in fiu care corespunde primului caracter din acesta in fiul care corespunde celui de-al doilea caracter s.a.m.d in acest mod gasim ce nod in arbore ii corespunde unei anumite pozitii dintr-un cuvant.

Matrice – soluţie

Se parcurg în ordine coloanele matricei. Pentru o coloană `j` se calculează pentru fiecare linie `i` o valoare `Left[i][j]` ca fiind lungimea maximă a unui şir valid de pe linia i care are ultimul element pe coloana `j`.
`Left[i][j]` se poate determina din `Left[i][j - 1]` memorând ultima poziţie (coloană) de pe fiecare linie unde şirul şi-a schimbat monotonia (deoarce două elemente consecutive de pe o linie sunt distincte, şirurile nu vor putea fi decât strict crescătoare/descrescătoare). Ca o observaţie, este suficient să memorăm doar ultima coloană a acestei matrice `Left`.

Problema revine la a afla submatricea de arie maximă care are coloana din dreapta `j`, ştiind pentru fiecare linie lungimea maximă a unui şir valid de pe linia `i` care se termină pe coloana `j` (`Left[i][j]`).

Acest lucru se poate face în O(N) pentru fiecare coloană, ţinând o stivă de elemente crescătoare. De exemplu, pentru o matrice cu 4 linii considerăm că pentru o coloană `j` am calculat următoarele valori `Left[i][j]`: 3, 4, 1, 2. Introducem 3 şi 4 în stivă, iar când introducem 1 va trebui să scoatem din stivă pe 4 şi apoi pe 3. De fiecare dată când scoatem un număr din stivă vedem ce dreptunghiuri pot apărea care să îl conţină.
Această soluţie are complexitatea O(M*N) şi obţine 100 puncte.

O soluţie care pentru fiecare coloană, având calculate valorile `Left[i][j]`, rezolvă problema în $O(N^2)$ (deci are complexitate totală $O(M*N^2)$) va obţine 70 puncte.

O soluţie care pentru fiecare coloană calculează de fiecare dată valorile `Left[i][j]` (mergând în stânga atât cât este nevoie), deci având complexitatea totală $O(N^2*M^2)$, va obţine 40 puncte.

Petrom – soluţie

Vom precalcula mai întâi o matrice `cost`:
`cost[a][b]` = costul de transport pentru cazul în care un depozit alimentează benzinăriile din intervalul `[a,b], a<b`
`cost[a][a]=0`
Costul optim se obţine amplasând un depozit în benzinăria situată în mijlocul intervalului.

Soluţia problemei se obţine prin programare dinamică.
Subprobleme:
Să se determine o amplasare optimă a `i` depozite în benzinăriile `1,..., j`
Vom memora soluţiile subproblemelor în matricea `cmin`:
`cmin[i][j]`= costul minim de transport în condiţiile în care construim `i` depozite în benzinăriile `1, ..., j`

Relaţia de recurenţă:

```
  cmin[1][j]=cost[1][j], pentru j=1, ..., n
  cmin[i][j]=min {cmin[i-1][p-
1]+cost[p][j]| p=i,i+1, ..., j}
```

Pentru reconstituirea soluţiei vom utiliza o matrice `back`
`back[i][j]`= poziţia `p` pentru care se obţine costul minim `cmin[i][j]`