empowersoft

Concurs online de informatică
Categoria PROGRAMARE

Secţiunea 11-12

PROBLEMA 1 **Autor: Iulia Tamaș**
graduated "Cum laude" Yale University

CUVINTE ASCUNSE
Se construieste un Trie, in care se insereaza fiecare cuvant din dictionar, cu literele mari transformate in litere mici. In fiecare nod din trie, o variabila denoteaza daca suntem la finalul unui cuvant din dictionar. O alta variabila, counter, initial 0, denoteaza de cate ori am gasit cuvantul respectiv in tabel.

Se porneste of functie de tip DFS din fiecare celula din tabel, apeland-o recursiv pe fiecare celula vecina. Se parcurge Trie-ul in acelasi timp, si pentru fiecare nod care denota un final de cuvant, se incrementeaza counterul.

Pentru a evita apeluri recursive inutile, nu apelam DFS daca nu exista nodul corespunzator in Trie (daca nodul nu exista in Trie, stim deja ca orice cuvant cu prefixul curent nu exista in dictionar).

Parcurgand Trie-ul, obtinem lista de cuvinte in ordine crescatoare, si counterele lor. Parola se obtine dupa regula mentionata in enunt.

PROBLEMA 2 **Autor: Iulia Tamaș**
graduated "Cum laude" Yale University

PEOPLEBOOK
Problema ne cere ca pentru fiecare punct i, sa aflam cele mai apropiate k puncte si sa le afisam in ordine crescatoare a distantei de punctul i.

Solutia de 100p

O solutie eficienta foloseste quadtrees. Construim quadtreeul astfel:

- fiecare nod din quadtree are asociata o zona din planul 2d al punctelor; alegem zona nodului radacina astfel incat sa acopere toate cele n puncte (de ex. este suficient ca zona sa fie un patrat cu diagonala (0, 0), (10000, 10000)).
- fiecare nod din quadtree este ori un nod frunza, ori un nod interior cu pointeri catre cei 4 quadnodes copii
- fiecare nod frunza contine l puncte, unde l <=k si cele l puncte au coordonatele in interiorul regiunii din plan de care e raspunzator nodul frunza
- fiecare nod interior este un nod a carui regiune din plan contine minim k+1 puncte. Impartim regiunea din plan a nodului in 4 regiuni egale, si le asociam ca regiuni celor 4 copii ai nodului interior.

Pentru a obtine solutia ne bazam pe urmatoarea observatie: pentru fiecare punct i, aflat intr-un nod frunza f, cele mai apropiate k puncte, se afla la o distanta mai mica sau egala cu distanta de la i la cel mai indepartat colt al nodului parinte(f) (Demonstratia mai jos). Un colt al unui nod este unul din cele patru colturi ale regiunii din plan de care este raspunzator nodul.

Ploiești, 28 mai 2016

Pe baza observatiei de mai sus, avem urmatoarea solutie: pentru fiecare punct i, aflat intr-un nod frunza f, calculam distanta r pana la cel mai indeparatat colt al nodului parinte(f). Cream un cerc C, cu centrul in punctul i si raza r. Cream un vector numit 'suspecti' in care vom pune punctele care au potential de a fi intre cele mai aproapiate k puncte de i. Traversam quadtreeul incepand cu nodul radacina, trecand prin fiecare nod a carui regiune de plan intersecteaza cercul C. Cand traversand quadtreeul, daca trecem printr-un nod frunza, copiem punctele din nodul frunza, in nodul de suspecti. Cand trecem printr-un nod interior, decidem pe baza testului de intersectie cu cercul C, care din cei 4 copii trebuie vizitati. Dupa ce terminam traversarea, sortam vectorul de suspecti si alegem primele k puncte ca distanta fata de punctul i.

Demonstratie. Dat modul in care am decis sa construim quadtreeul, fiecare nod interior acopera cel putin k+1 puncte. Astfel, pentru orice punct i, aflat in frunza f, nodul parinte(f) are cel putin k puncte diferite de i. Toate aceste puncte se afla la o distanta fata de i mai mica sau egala de distanta de la i pana la cel mai indepartat colt al parinte(f). Numim aceasta distanta `r.' Data fiind aceasta solutie potential neoptima (unde alegem ca prieteni pentru i k din punctele din nodul parinte(f)), stim ca punctele din solutia optima sunt la o distanta <= r de punctul i.

Solutii partiale

Punctaje partiale se pot obtine calculand distantele dintre fiecare pereche de puncte (i, j). Diferenta dintre solutiile partiale prezentate mai jos este modul in care se decid (sorteaza si aleg) cele mai apropiate k puncte pe baza calcularii tuturor celor n^2 distante.

Solutia naiva-1, complexitate teoretica: $O(n^2*k)$
Pentru fiecare punct i, cele k puncte se afla astfel:

Pentru fiecare punct j, calculam distanta dist(i,j). Mentinem un vector de k elemente, d, pe care il actualizam pe masura ce calculam distante dist(i,j), astfel incat d sa contina cele mai apropiate k puncte de i pe care le-am vazut pana acum.

Actulizarea vectorului d e similara cu operatia de insert pentru insert-sort. Pentru ca solutia sa fie mai rapida, mentinem mereu si maxj, punctul cel mai departat de i. Inainte de a actualiza d cu un punct j, verificam mai intai ca punctul j este mai aproape de i ca punctul maxj. Daca j este mai departe atunci, j nu are cum sa faca parte din solutie. Astfel reusim ca in practica sa evitam efectuarea operatiei de actualizare, si timpul de rulare al solutiei este mult sub complexitatea teoretica de $O(n^2*k)$.

Solutia este data de cele k elemente din d.

Solutia naiva-2, complexitate teoretica: $O(n^2*\log(k))$
Pentru fiecare punct i, cele k puncte se afla astfel:

Pentru fiecare punct j, calculam distanta dist(i,j) si o plasam intr-un vector v. Efectuam o sortare partiala a lui v (folosind partial_sort), astfel incat cele mai mici k puncte (ca distanta fata de i) din v sa fie sortate.

Solutia este data de primele k elemente din v.

Ploiești, 28 mai 2016

Solutia naiva-3, complexitate teoretica: $O(n^2*\log(n)$
Pentru fiecare punct i, cele k puncte se afla astfel:

Pentru fiecare punct j, calculam distanta dist(i,j) si o plasam intr-un vector v. Efectuam o sortare completa a lui v (folosind sort).

Solutia este data de primele k elemente din v.

PROBLEMA 3 **Autor: elev Cristi Dospra**
Colegiul Național "Grigore Moisil", București

SEC
Soluție O(N * Q) – 30p
Începem rezolvarea prin a face o parcurgere Euler a arborelui dat. Se poate observa că fii unui nod sunt identici dacă secvențele din parcurgerea Euler corespunzătoare acestora sunt identice. Astfel putem verifica la fiecare interogare simetria celor doi fii în timp liniar.

Soluție O(N + M) – 100p
Pentru obținerea punctajului maxim vom optimiza soluția precedentă: Pe parcurgerea Euler obținută aplicăm algoritmul lui Manacher pentru aflarea celui mai lung palindrom centrat într-o anumită poziție. La fiecare interogare putem răspunde în timp constant dacă cei doi fii sunt sau nu identici. O alternativă pentru rezolvarea acestei probleme este de a folosi în loc de algoritmul lui Manacher algoritmul Z, de două ori.

PROBLEMA 4 **student Vlad Costin**
Universitatea Politehnica Bucuresti

UNICORN
Solutia ($O(n^2)$ – 30 puncte) se bazeaza pe programare dinamica. Fie vectorul sol, unde sol[i] = greutatea si lungimea turnului corn ce reprezinta solutia pentru prefixul 1 … i al clatitelor initiale, in care se gaseste clatita cu indicele i. Astfel , pentru a calcula sol[i], trebuie sa ne uitam la toate clatitele din spate cu raza mai mare decat raza clatitei i si alegem cea mai buna solutie (max(s[j]), j < i si raza[j] > raza[i]), apoi adunam la greutatea ei , greutatea clatitei cu indicele i si crestam cu 1 lungimea. Solutia nostraa va fi maximul din fiecare sol[i], 1<= i <= n Aceasta metoda ar iesi din timp pentru 70% din teste.

Solutia optima (O(n*log(n)) – 100 puncte) se foloseste de dinamica anterioara , modificand algoritmul astfel incat sa putem gasi clatita din spate cu raza mai mare care ne ofera cea mai buna solutie in O(log n). Astfel vom folosi un arbore de interval in care vom calcula maximul solutiei pe un anumit interval de lungimi de raze. Deoarece razele clatitelor variaza intre 1 si 10^9, trebuie mai intai sa normalizam valorile razelor (sortam clatitele dupa raza si inlocuim fiecare raza cu indicele clatitei in vectorul sortat, avand grija la razele egale). Dupa normalizare , domeniul valorilor razelor a devenit [1, n] , pastrand ordinea din sirul initial(relative la raza), deci nu am modificat cu nimic gradul de generalizare al problemei. Acum construim arborele de intervale, iar la fiecare pas din dinamica , cautam solutia din arbore pe intervalul [R[i]+1, raza_maxima] in log(n), actualizam

maximul global, iar apoi updatam pe intrevalul [R[i], R[i]] cu solutia prefixului 1,…,I . Complexitate totala(O(n*log(n))