DESCRIEREA SOLUTIILOR, OLIMPIADA NATIONALA DE INFORMATICA, BARAJ, ZIUA 1

Problema 1: Cactus

Propusa de: stud. Matei Tinca

Fie V mulțimea nodurilor lui G și E mulțimea muchiilor lui G.

Subproblemele 1 și 2. Observăm că în aceste subprobleme nu se elimină nicio muchie: ni se cere doar să calculăm suma distanțelor între toate muchiile.

Fie $\text{len}(x,y)$ lungimea muchiei de la x la y. Fie $\text{dist}(x,y)$ distanța (minimă) dintre x și y în graf (în viitor, dacă sunt mai multe grafuri G', G'', G''' atunci $\text{dist}_G(x,y)$ este distanța minimă între x și y in G). Fie S suma distanțelor între toate muchiile. Înrădăcinăm arborele în nodul 1; fie $s(x)$ mărimea subarborelui cu rădăcina la x, și $t(x)$ părintele nodului x.

Teoremă 1. $S = \sum_{u=2}^{N} s(u)(N - s(u))\text{len}(t(u), u)$.

Demonstraţie. Pentru fiecare pereche de noduri distincte (x,y), fie $x = p_1^{xy}, \ldots, p_{d(x,y)}^{xy} = y$ lanțul dintre ele.

$$(1) \qquad S = \sum_{x<y} \text{dist}(x,y) \; = \sum_{x<y} \text{len}\left(p_1^{xy}, p_2^{xy}\right) + \ldots + \text{len}\left(p_{d(x,y)-1}^{xy}, p_{d(x,y)}^{xy}\right) = \sum_{u=2}^{N} c(u)\text{len}(t(u), u),$$

unde coeficientul $c(u)$ reprezintă de câte ori apare muchia de la $t(u)$ la u în suma din ecuația (1). Ne întrebăm acum: de câte ori apare oare? Observăm că muchia apare odată pentru fiecare pereche neordonată de noduri distincte (x,y) pentru care lanțul de la x la y conține muchia de la u la $t(u)$. Acest lucru se poate întâmpla daca și numai dacă unul dintre nodurile x, y este un descendent al lui u — u având $s(u)$ descendenți — și celălalt nod este un non-descendent al lui u — u având $N - s(u)$ non-descendenți. Reiese deci că $c(u) = s(u)(N - s(u))$, de unde rezultă egalitatea cerută. □

Pentru a rezolva subproblema aceasta, deci, este necesar (i) să folosim un DFS pentru a calcula $t(x)$ și $s(x)$, și apoi (ii) să folosim formula de mai sus pentru a calcula soluția. Complexitatea finala este, așadar, $O(N)$.

Pentru toate subproblemele ce urmează, vom descrie o procedură pentru a găsi muchiile eliminate din graf. După eliminarea lor, se ruleaza procedura de mai sus pentru a calcula soluția efectivă.

Subproblema 2. Pentru acesată subproblemă, se aleg muchiile pe care le eliminăm în toate modurile posibile, apoi se aplică soluția precedentă pentru a calcula rezultatul în fiecare caz. Rezultatul final este minimul acestor rezultate parțiale. Ne interesează acum care este numărul de moduri în care putem să alegem muchiile eliminate.

Observație 1. $M \leq \frac{4N}{3} - 1$.

Demonstraţie. Observăm că $M = (N-1) + (\textit{numărul de ciclii})$. Cum fiecare ciclu conține cel puțin 3 noduri, și fiecare nod este în cel mult un ciclu, rezultă că $M \leq (N-1) + \frac{N}{3} = \frac{4N}{3} - 1$. □

În câte moduri putem să alegem $M - (N-1)$ muchii din M? În $\binom{M}{M-(N-1)} = \binom{M}{N-1} \leq \binom{4N/3-1}{N-1}$ moduri. Pentru $N = 15$, numărul acesta este 11 628, deci putem efectiv să iterăm prin toate modurile, apoi să aplicăm soluția precedentă.

Subproblema 4. Pentru subproblema aceasta, trebuie să observăm că *fiecare ciclu este independent*. Mai exact, din fiecare ciclu vom elimina exact o muchie, iar muchia ce ar trebui eliminată nu depinde de celelalte eliminări.

Observație 2. *Pentru orice muchie (x,y) din graf, indiferent de modul în care muchiile din afara ciclului ce conține muchia au fost eliminate (cât timp graful rămâne conex), greutatea grafului crește cu o constantă $\Delta(x,y)$ la eliminarea muchiei (x,y).*

Demonstrație. Fie G' starea curentă a grafului, din care nu a fost eliminat nicio muchie din ciclul lui (x,y). Observăm că diferența dintre greutatea lui G' și greutatea lui $G'-(x,y)$ este suma, pentru fiecare pereche de noduri u,v, dintre diferența din lungimea lanțului minim de la u la v in $G'-(x,y)$ și lungimea lanțului minim de la u la v în G'. Considerăm acum o pereche u,v oarecare. Considerând lanțul de la u la v, fie u' cel mai apropiat nod de u din ciclul lui (x,y), și v' cel mai apropiat nod de v din ciclul acesta. Dacă lanțul de lungime minimă de la u la v nu conține muchia (x,y) în G', atunci perechea u,v nu contribuie la diferența menționată anterior. Altfel, dacă lanțul îl conține pe (x,y), diferența este egală cu lungimea lanțului de la u' la v' ce *nu trece* prin (x,y), minus lungimea lanțului de la u la v ce *trece* prin (x,y).

Definim acum $g(u')$ ca fiind numărul de noduri pentru care cel mai apropiat nod din ciclul lui (x,y) este u'. Rearanjând suma de mai sus, și definind $d(u',v')$ ca fiind 0 dacă lanțul de lungime minimă de la u' la v' nu trece prin (x,y), și

(*lungimea lanțului de la u' la v' ce nu trece prin* (x,y))–
(*lungimea lanțului de la u' la v' ce trece prin* (x,y))),

altfel, ajungem la concluzia că diferența dintre greutatea lui G' și $G'-(x,y)$ este
$\sum_{u',v'} d(u',v')g(u')g(v')$, unde suma este pentru (u',v') orice muchie din ciclul lui (x,y) în afară de (x,y). Dar, $g(u'), g(v'), d(u',l')$ *nu depind de modul în care au fost eliminate muchiile din* G! Prin urmare, oricare ar fi starea curenta G' a grafului, diferența între greutatea sa înainte și după eliminarea muchiei (x,y) este $\sum_{u',v'} d(u',v')g(u')g(v')$, care este prin urmare fix valoarea $\Delta(x,y)$ pe care o căutăm. □

Așadar, am demonstrat că putem considera fiecare ciclu separat, găsim care este modul optim de a elimina o muchie din fiecare ciclu, și să combinăm toate soluțiile. Pentru a găsi modul optim de a elimina o muchie dintr-un ciclu, este necesar doar să aplicăm soluția din prima subproblemă pentru fiecare mod în care se poate elimina muchia din ciclu. (În afara ciclului, se pot elimina muchii arbitrar până când ne rezulta un arbore.) Complexitatea finala este $O(N^2)$.

Subproblema 3. Fie ciclul $x_1,\ldots,x_N$. Definim $x_{N+1}=x_1,\ldots,x_{2N}=x_N$. Fie $l_i=\text{len}(x_{i-1},x_i)$, $l_1=\text{len}(x_1,x_N)$. Dacă se elimină muchia (x_{u-1},x_u), pentru $2\le u\le N+1$, observăm că greutatea lanțului rezultat, notată cu S_u, este

$$S_u=\sum_{i=u}^{u+N-2}\sum_{j=i+1}^{u+N-1}\text{dist}(x_i,x_j)=\sum_{i=u}^{u+N-2}\sum_{j=i+1}^{u+N-1}\sum_{k=i+1}^{j}l_k$$

Observam acum că termenul l_k apare de exact $(k-u)(u+N-k)$ ori în suma de mai sus; așadar,

$$S_u=\sum_{k=u+1}^{u+N-1}(k-u)(u+N-k)l_k=\sum_{k=u+1}^{u+N-1}-k^2l_k+(2u+N)kl_k-u(u+N)=$$
$$-\sum_{k=u+1}^{u+N-1}k^2l_k+(2u+N)\sum_{k=u+1}^{u+N-1}kl_k-\sum_{k=u+1}^{u+N-1}u(u+N)$$

Pentru a calcula această formulă mai rapid, definim (și calculăm în $O(N)$)

$$s_i^{(0)} = l_1 + l_2 + \ldots + l_i$$
$$s_i^{(1)} = l_1 + 2l_2 + \ldots + il_i$$
$$s_i^{(2)} = l_1 + 4l_2 + \ldots + i^2 l_i$$

Acum, greutatea arborelui dacă este eliminată muchia (x_{u-1}, x_u) este

$$S_u = -\sum_{k=u+1}^{u+N-1} k^2 l_k + (2u+N)\sum_{k=u+1}^{u+N-1} k l_k - \sum_{k=u+1}^{u+N-1} u(u+N)$$
$$= -(s_{u+N-1}^{(2)} - s_u^{(2)}) + (2u+N)(s_{u+N-1}^{(1)} - s_u^{(1)}) - u(u+N)(s_{u+N-1}^{(0)} - s_u^{(0)}).$$

Soluția optimă se găsește iterând prin toate valorile posibile ale lui u, și găsind u-ul ce minimizează expresia de mai sus.

Soluție completă. Observăm, precum în cazul subproblemei 3, că este necesar doar să alegem, pentru fiecare ciclu, modul optim de a șterge o muchie din ciclu. Așadar, fie G' un subgraf conex de al lui G care conține exact un ciclu, $x_1, \ldots, x_n$. Vrem să alegem muchia din ciclul acesta care, dacă e eliminată, crește greutatea lui G' cât mai puțin. Acest lucru este echivalent cu a alege o muchie (x_i, x_j) care minimizează greutatea lui $G' - (x_i, x_j)$. Vom da acum o formulă care exprimă această greutate (minus o constantă); pentru a găsi muchia optima, se itereaza prin toate muchiile din ciclu și apoi se alege muchia ce minimizează formula dată. Formula *nu va depinde de* G', așadar se poate calcula fără a fixa un astfel de subarbore (subarborele G' apare doar in demonstrația corectitudinii formulei).

Definim $x_{n+1} = x_1, \ldots, x_{2n} = x_n$, $l_i = \text{len}(x_{i-1}, x_i)$, $l_1 = \text{len}(x_n, x_1)$. Definim $\text{dist}'(x_i, x_j)$, pentru $i < j$, ca fiind lungimea lanțului $x_i, x_{i+1}, \ldots, x_j$. Fie w_i numărul de noduri pentru care cel mai apropiat nod dintre $x_1, \ldots, x_n$ este x_i. Observăm că toate aceste valori se pot calcula doar în graful G, nu au nevoie de G'. Pentru a îl calcula pe w_i, observăm că cel mai apropiat nod dintr-un ciclu este neschimbat oricum am șterge noduri din G; așadar putem alege un arbore parțial $\widehat{G}$ al lui G în mod arbitrar. Facând un DFS al lui $\widehat{G}$, calculăm $s(i)$ = *mărimea subarborelui lui* i. Observăm că w_i este s_i minus s_j pentru toți fiii lui i care aparțin mulțimii $\{x_1, \ldots, x_n\}$. Toate acestea se pot calcula în timp linear.

Să zicem acum că eliminăm muchia (x_{u-1}, x_u) din ciclu. Notăm greutatea lui $G' - (x_{u-1}, x_u)$ cu S_u. Observăm că

$$S_u = \sum_{p,q \in V} \text{dist}_{G'-(x_{u-1},x_u)}(p,q) = \sum_{(r,s)\in E} c(r,s)\,\text{len}(r,s).$$

Acest fapt este pentru că $\text{dist}_{G'-(x_{u-1},x_u)}(p,q)$ este o sumă de lungimi; așadar $c(r,s)$ este numărul de ori de care apare $\text{len}(r,s)$ în suma de mai sus, sau, în mod echivalent, numărul de lanțuri pe care se află (r,s). Observăm că, dacă (r,s) nu aparține ciclului $x_1, \ldots, x_n$, atunci $c(r,s)$ depinde doar de G'. Acum, separând muchiile (r,s) ce sunt in ciclul $x_1, \ldots, x_n$ de celelalte,

$$S_u = \sum_{(r,s)\in E} c(r,s)\,\text{len}(r,s). = \sum_{(r,s)\in E - \{(x_1,x_2),\ldots,(x_N,x_1)\}} c(r,s)\,\text{len}(r,s) + \sum_{k=u+1}^{u+N-1} c(x_{k-1},x_k) l_k$$
$$= C + \sum_{k=u+1}^{u+N-1} c(x_{k-1},x_k) l_k,$$

unde C este prima sumă. Cum toate elementele sumei C sunt de forma $c(r,s)\text{len}(r,s)$ pentru (r,s) care nu e o muchie a ciclului $x_1, \ldots, x_n$ — și cum $c(r,s)\text{len}(r,s)$ nu depinde de u în cazul acesta, observăm că C *nu depinde de alegerea lui* u.

Cât este $c(x_{k-1}, x_k)$? Acesta reprezintă numărul de lanțuri din $G' - (x_{u-1}, x_u)$ care trec prin muchia (x_{k-1}, x_k). Echivalent, este numărul de lanțuri ce intră în ciclul $x_u, \ldots, x_{u+n-1}$ la nodul x_i, și ies din el la nodul x_j, astfel încât $i < k \leq j$. Numărul de lanțuri ce intră la nodul x_i și ies la nodul x_j este exact $w_i w_j$, așadar observăm că

$$S_u - C = \sum_{k=u+1}^{u+N-1} c(x_{k-1}, x_k) l_k = \sum_{k=u+1}^{u+N-1} l_k \sum_{i=u}^{i<k} \sum_{j=k}^{j<u+N} w_i w_j$$

Pentru a putea calcula mai rapid suma aceasta, definim (și calculăm în $O(n)$)

$$\begin{aligned} W_i &= w_1 + \ldots + w_{i-1} \\ s_i^{(0)} &= l_1 + \ldots + l_i \\ s_i^{(1)} &= W_1 l_1 + \ldots + W_i l_i \\ s_i^{(2)} &= W_1^2 l_1 + \ldots + W_i^2 l_i \end{aligned}$$

Acum,

$$\begin{aligned} S_u - C &= \sum_{k=u+1}^{u+N-1} l_k \sum_{i=u}^{i<k} \sum_{j=k}^{j<u+N} w_i w_j = \sum_{k=u+1}^{u+N-1} l_k \left(\sum_{i=u}^{i<k} w_i \right) \left(\sum_{j=k}^{j<u+N} w_j \right) = \sum_{k=u+1}^{u+N-1} l_k (W_k - W_u)(W_{u+N} - W_k) \\ &= \sum_{k=u+1}^{u+N-1} -l_k W_k^2 + l_k W_k (W_{u+N} + W_u) - l_k W_{u+N} W_u \\ &= -\sum_{k=u+1}^{u+N-1} -l_k W_k^2 + (W_{u+N} + W_u) \sum_{k=u+1}^{u+N-1} l_k W_k - W_{u+N} W_u \sum_{k=u+1}^{u+N-1} l_k \\ &= -(s_{u+N-1}^{(2)} - s_u^{(2)}) + (W_{u+N} + W_u)(s_{u+N-1}^{(1)} - s_u^{(1)}) - W_{u+N} W_u (s_{u+N-1}^{(0)} - s_u^{(0)}). \end{aligned}$$

Astfel, făcând precalculare $O(n)$, putem calcula în timp constant valoarea $S_u - C$, adică greutatea lui $G' - (x_{u-1}, x_u)$ minus o constantă ce nu depinde de u. Precalcularea noastră nu depinde de graful G'. Tot ce trebuie să facem pentru a putea găsi muchia cea mai buna de tăiat din ciclul $x_1, \ldots, x_n$ este să iteram prin toate muchiile (x_{u-1}, x_u) și să calculăm valoarea $S_u - C$, alegând muchia care minimizează această valoare. Cum C nu depinde de u, aceasta procedură este corecta. La final, eliminăm muchia aleasă pentru fiecare ciclu.

Notă asupra implementării: sumele din formulele de mai sus pot cu ușurință să depășească limita tipului întreg de 64 de biți. Totuși , rezultatul final al oricărui calcul trebuie să fie un număr non-negativ având valoarea cel mult 10^{18}*. Așadar, efectuând operațiile pe tipul de date* `unsigned long long` *va da rezultatul corect. (În practică, majoritatea compilatoarelor C/C++ folosesc aritmetica "two's complement", deci se poate folosi și tipul de date* `long long`*.)*

Problem 2: Şirbun

Propusa de: prof. Mihai Bunget

Subproblema 1. Se generează toate secvenţele din şirul A, iar pentru fiecare element A_i dintr-o secvenţă se verifică dacă numărul elementelor din secvenţă mai mici sau egale cu el este cel mult A_i, caz în care secvenţa este *bună*.

Complexitate $O(N^4)$

Subproblema 2. Se generează toate secvenţele din şirul A, iar pentru fiecare secvenţă se calculează frecvenţa elementelor din secvenţă. Dacă toate frecvenţele sunt mai mici sau egale cu valoarea elementului atunci secvenţa este *bună*.

Complexitate $O(N^3)$

Subproblema 3. Se parcurge şirul A calculând frecvenţa fiecărui element. La adăugarea unui nou element se calculează şi sumele parţiale ale frecvenţelor elementelor mai mari sau egale cu el. Dacă o frecvenţă a unui element depăşeste valoarea elementelului atunci se adaugă la soluţie numărul secvenţelor *bune* găsite. Se elimină din secvenţa curentă elemente din stânga până când orice frecvenţă devine cel mult egală cu valoarea elementului corespunzător. Apoi se reia procesul de adăugare la dreapta a noilor elemente.

Complexitate $O(N^2)$

Subproblema 4. Se aplică acelaşi algoritm ca la subproblema anterioară însă diferă modul de verificare a faptului că frecvenţa fiecărui element din secvenţă este cel mult egală cu valoarea elementului. Astfel se împarte şirul în secvenţe de lungime $[\sqrt{N}]$, reţinând pe fiecare secvenţă minimul diferenţelor dintre valoarea elementului şi suma frecvenţelor elementelor mai mici sau egale cu el. Când acest minim devine negativ secvenţa nu este *bună*, caz în care trebuie să eliminăm elemente din stânga secvenţei până aceasta devine *bună*.

Complexitate $O(N[\sqrt{N}])$

Subproblema 5. Se aplică acelaşi algoritm ca la subproblema anterioară însă diferenţele dintre valoarea elementului şi suma frecvenţelor elementelor mai mici sau egale cu el se va reţine într-un arbore de intervale cu "lazy propagation".

Complexitate $O(NlogN)$

Problema 3: Bt

Propusa de: stud. Vlad-Adrian Ulmeanu

Subproblema 1 (10p). Simulăm toate posibilitățile de a goli cutia. Putem să ne oprim dintr-o ramură a backtracking-ului în momentul în care toate elementele rămase sunt distincte, însă în practică această optimizare nu este de ajuns pentru $n = 20$.

Complexitate: $O(n!)$.

Subproblema 2 (10p). Rezolvăm problema folosind programare dinamică pe stări exponențiale. Fie d_{conf} = în câte moduri putem goli o cutie care mai are doar bucățile din mulțimea $conf$ în ea.

Dacă $conf$ are un singur element, $d_{conf} = 1$. Altfel:

$$d_{conf} = \sum_{x \in conf, v_{prev(x)} \neq v_x, v_{next(x)} \neq v_x} d_{conf \setminus x}$$

Unde $v_{prev(x)}$ si $v_{next(x)}$ sunt valorile bucăților rămase adiacente din cutie. Răspunsul este $d_{\{1,..,n\}}$.

Complexitate: $O(n2^n)$.

Subproblema 3 (30p). Încercăm să rezolvăm problema invers, pornind cu cutia goală și punem bucăți în ea, una câte una, având grijă să nu avem niciodată două bucăți adiacente de același tip. Trebuie să terminăm cu forma inițială a cutiei.

Această subproblemă se poate rezolva prin programare dinamică, dar permite mai multe dimensiuni decât următoarea subproblemă.

Considerând că avem un delimitator, vrem să calculăm $d_{i,j,l,r}$: care este numărul de moduri în care putem umple intervalul $[i, j]$, știind că cele mai apropiate două bucăți care au fost deja puse înapoi sunt pe pozițiile $l < i$ și $r > j$.

$$d_{i,j,l,r} = \sum_{k \in \{i,..j\}, v_k \neq v_l, v_k \neq v_r} d_{i,k-1,l,k} \cdot d_{k+1,j,k,r} \cdot C_{j-i}^{k-i}$$

Când umplem $[i, j]$, vedem ce bucată va fi pusă prima în interval. Presupunând că o vom pune pe poziția k, trebuie să avem $v_l \neq v_k$ și $v_k \neq v_r$. Trebuie să numărăm și în câte moduri putem intercala bucățile puse în $[i, k-1]$ cu cele puse în $[k+1, j]$: C_{j-i}^{k-i}.

Trebuie să avem grijă la calcularea unor valori: $d_{i,i-1,?,?} = d_{i+1,i,?,?} = 1 \ \forall i \in \{1,..,n\}$.

Soluția pentru delimitator se obține în $d_{1,n,0,n+1}$. Putem considera că $v_0 = v_{n+1} = 0$ (valori diferite de oricare altele din cutie).

Complexitate: $O(n^5)$.

Subproblema 4 (50p). Observăm că vom calcula întotdeauna valori în dinamica de la subproblema anterioară doar pentru $l = i - 1$ și $r = j + 1$. Astfel, trebuie să calculăm doar $d_{i,j}$, subînțelegându-se că valorile de pe pozițiile $i - 1$ și $j + 1$ au fost deja puse.

Pentru a rezolva cazul cu vectorul circular, simulăm punerea primului element în cutie pentru toate posibilitățile ($\forall i \in \{1,..,n\}$), iar rezultatul este:

$$\sum_{\forall i \in \{1,..,n\}} d_{next(i),prev(i)}$$

Unde funcția *next* se definește ca $next(i) = i \mod n + 1$ și dă indicele următorului element din cutie, iar $prev(i) = (i + n - 2) \mod n + 1$ dă indicele elementului precedent din cutie.

Altă soluție mai ușor de implementat presupune dublarea vectorului. $d_{i,j}$ = în câte moduri putem umple $[i + 1, j - 1]$, știind că am pus bucățile de pe pozițiile i și j. Aici $i < j < 2n$.

Pornim cu $d_{i,i+1} = 1 \ \forall \ 0 < i < 2n$ și calculăm într-un mod asemănător $d_{i,i+len} \ \forall \ 2 < len \leq n + 1$. Răspunsul este $\sum_{i \in \{1,..,n\}} d_{i,i+n}$ (unde i reprezintă poziția primei valori puse).

Complexitate: $O(n^3)$.