Netafræði, vika 2

Dæmatími

Atli Fannar Franklín

9. mars 2022

Dominodæmi

Þú hefur n dominokubba og gefið að þegar sumir þeirra detta rekast þeir á aðra dominokubba. Hver er fæsti fjöldi kubba sem þú þart að velta handvirkt til að allir detti?

Inntakið byrjar á t, fjölda prófunartilfella. Hvert tilfelli byrjar á n, m. Svo koma m línur með tölum x, y sem merkja að ef kubbur x dettur gerir kubbur y það einnig.

SCC

Þetta er s.s. stefnt net. Ef kubbur x veldur því að y detti og öfugt getum við allt eins litið á þá sem sama kubbinn. Því getum við tekið herpingu netsins og unnið með hana!

Notum því reiknirit Tarjans til að fá herpta netið og grannröðun á því. Ef enginn kubbur getur orðið til þess að x detti verðum við að fella hann handvirkt. Því fjarlægjum við þannig kubba endurtekið og hendum út öllu sem þeir fella. En þetta er svipað því sem við gerum í Kahn, svo þetta mun á endanum klára alla hnúta og gefa okkur svar!

Forrit

```
#include <bits/stdc++.h>
using namespace std;
typedef vector<int> vi; typedef vector<vi> vvi;
#define rep(i,a,b) for(auto i = (a); i < (b); ++i)
// SCC forrit hér, hefur komið fram áður í glærum
int main() { int t, n, m, x, y;
    cin >> t; while(t--) {
        cin >> n >> m; vvi g(n);
        rep(i, 0, m) {
            cin >> x >> y; x--; y--;
            g[x].push_back(y); }
        auto res = scc(g); vi indeg(n, 0);
        rep(i, 0, n) { for(int z : g[i]) {
                if(res.first.united(i, z)) continue;
                indeg[res.first.find(z)]++;
            } }
        int cnt = 0; for(int z : indeg) if(z != 0) cnt++;
        cout << res.first.c - cnt << '\n'; } }
```

Dæmið

Þú ert staddur í Zürich og vilt komast út í ETH. Til þess notarðu sporvagnakerfið, en þú vilt leggja eins seint af stað og auðið er án þess að mæta of seint. Hversu seint geturðu laft af stað?

Inntakið byrjar á n, m, s, fjöldi stoppistöðva, fjöldi sporvagnalína og eftir hve margar sekúndur þú verður að vera mættur. Næstu m línur innihalda hver 5 tölur u, v, t_0, p, d. Þetta merkir að þessi sporvagnalína fer frá stoppistöð u til stoppistöðvar v. Hún leggur fyrst af stað eftir t_0 sekúndur og eftir það leggur annar eins sporvagn af stað hverjar p sekúndur. Ferðin tekur d sekúndur. Stöðvarnar eru númeraðar frá 0 til $n-1$ og þú vilt komast frá stöð 0 til $n-1$. Gert er ráð fyrir að það taki engan tíma að skipta um sporvagn, svo ef hún leggur af stað við tíma t og þú mætir þangað á tíma t kemstu í vagninn.

Einhalla

Það að leggja af stað seinna getur stundum breytt engu, stundum látið þig mæta seinna en það getur aldrei látið þig mæta fyrr. Því er eitthvað gildi x þannig að ef þú leggur af stað á tíma $\leq x$ nærðu tímanum en fyrir öll $> x$ ertu of seinn.

Því er þetta einhalla skilyrði og getum við því helmingunarleitað að gildinu x. Því erum við búin að smækka verkefnið niður í 'Ef við leggjum af stað við tíma t, verðum við of sein?'.

Dijkstra

Til þess að finna stystu leið frá 0 til $n-1$ að byrjunartíma gefnum getum við bara notað Dijkstra! Við geymum þá alltaf hvenær við getum fyrst verið komin í einhvern hnút frekar en fjarlægðina þangað.

Þetta er pínu erfitt að gera rétt, en dugar til að leysa dæmið.

Forrit, hluti 1

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef pair<ll,ll> lll;
typedef vector<ll> vll;
typedef vector<vll> vvll;
ll n, s; vvll g;

struct tram {
    ll from, to, start, freq, dur;
    tram(ll fr, ll t, ll s, ll f, ll d) :
        from(fr), to(t), start(s), freq(f), dur(d) { }
};

vector<tram> trams;
```

Forrit, hluti 2

```
bool dkstr(ll st) {
    priority_queue<lll> q; vll dist(n, LLONG_MAX);
    dist[0] = st; q.push(lll(-st, 0));
    while(!q.empty()) {
        if(dist[q.top().second] != -q.top().first) {
            q.pop(); continue; }
        ll cur = q.top().second; q.pop();
        for(ll x : g[cur]) {
            tram t = trams[x];
            ll tm = dist[cur];
            tm += ((t.start % t.freq)
                - (dist[cur] % t.freq) + t.freq) % t.freq;
            tm = max(tm, t.start);
            tm += t.dur;
            if(dist[t.to] > tm) {
                dist[t.to] = tm;
                q.push(lll(-dist[t.to], t.to)); } } }
    return dist[n - 1] <= s; }
```

Forrit, hluti 3

```
int main() {
    ll m, u, v, t, p, d;
    cin >> n >> m >> s;
    g.resize(n);
    for(ll i = 0; i < m; ++i) {
        cin >> u >> v >> t >> p >> d;
        trams.push_back(tram(u, v, t, p, d));
        g[u].push_back(i); }
    if(!dkstr(0)) {
        cout << "impossible" << endl;
        return 0; }
    int lo = 0, hi = s, mid;
    while(hi - lo > 0) {
        mid = hi - (hi - lo) / 2;
        if(dkstr(mid)) lo = mid;
        else hi = mid - 1;
    }
    cout << hi << endl; }
```

Dæmið

Gefnir eru C gjaldmiðlar. Svo eru R skiptimöguleikar gefnir þar sem hægt er að skipta x einingar af gjaldmiðli út fyrir y einingar af öðrum gjaldmiðli. Að þessu gefnu, segðu hvort hægt sé að græða pening á að skipta fram og til baka með einhverjum hætti eða ekki.

Prófunartilfelli er gefið þannig að $C \leq 200$ er á fyrstu línu. Svo koma nöfn gjaldmiðlanna á næstu línu með bili á milli. Næsta lína inniheldur töluna R. Næstu R línur innihalda skiptingar þar sem fyrst eru gefin nöfnin á gjaldmiðlinum og svo skiptihlutfallið á forminu `x:y`. Mörg prófunartilfelli geta verið í inntaki en $C = 0$ í því síðasta og ekki á að svara því. Prenta á `Arbitrage` ef hægt er að græða en `Ok` annars.

Umbreyting vigta

Vigt leggja er þá í raun hlutfall gildanna sem gefin eru. Við viljum þá finna hvort til sé rás þar sem margfeldi allra vigtanna er > 1. En við vinnum lítið með margfeldi vigta, viljum mun frekar summu. Tökum því mínus logrann af öllum vigtum og þá verður verkefnið bara að leita að rás með neikvæða summu á vigtum. Til þess getum við einfaldlega notað Floyd Warshall reikniritið því C er svo lítið.

Floyd-Warshall

```
import math

def floyd_warshall(g):
    n = len(g)
    for k in range(n):
        for i in range(n):
            for j in range(n):
                g[i][j] = min(g[i][j], g[i][k] + g[k][j])
    return g
```

Restin af lausn

```
while True:
    c = int(input())
    if c == 0:
        break
    nm, ind = input().strip().split(), dict()
    for (i, s) in enumerate(nm):
        ind[s] = i
    r, g = int(input()), [[10 ** 20 for j in range(c)] for i in range(c)]
    for i in range(c):
        g[i][i] = 0
    for i in range(r):
        ln = input().strip().split()
        i1, i2 = ind[ln[0]], ind[ln[1]]
        ln = ln[2].split(':')
        g[i1][i2] = math.log(float(ln[0])) - math.log(float(ln[1]))
    floyd_warshall(g)
    arb = False
    for i in range(c):
        if g[i][i] < -1e-9:
            arb = True
    if arb:
        print("Arbitrage")
    else:
        print("Ok")
```