ACM/ICPC Template

AsZ VincentLDL

August 15, 2015

Contents

Chapter 1

java

1.1 读写

1.2 高精

Chapter 2

dp 优化

2.1 决策单调性优化

- 形式：f[i] = f[j] + w[j, i] 形式决策单调。
- 一般打表找规律看决策是否单调。
- 四边形不等式：w[i,j]+w[i + 1, j + 1] <= w[i + 1, j] + w[i + 1, j], 则满足决策单调性。
- 有时候不满足决策单调性，但是去掉完全不合法状态之后却可以满足。

```
#include <bits/stdc++.h>
#define MAXN 51234

using namespace std;
typedef long long arrayN[MAXN];

deque < pair< pair<int, int> , int> > deq;
arrayN f, sum, c;
long long L;

long long sqr(long long x)
{
    return x * x;
}

long long trans(int l, int r)
{
    return sqr(1LL * r - (l + 1) - L + sum[r] - sum[l]) + f[l];
}

int myLowBound(pair <int, int> pr, int ori, int now)
{
    int l = pr.first, r = pr.second;
    for (; l < r; )
    {
        int mid = l + r >> 1;
        if (trans(ori, mid) <= trans(now, mid)) l = mid + 1;
        else r = mid;
    }
    return l;
}

int main()
{
    int n;
    freopen("toys.in", "r", stdin);
    cin >> n >> L;
    for (int i = 1; i <= n; ++i)
    {
        cin >> c[i];
        sum[i] = sum[i - 1] + c[i];
    }
    deq.push_back(make_pair(make_pair(1, n), 0));
    for (int i = 1; i <= n; ++i)
    {
        for (; deq.front().first.second < i; deq.pop_front());
        f[i] = trans(deq.front().second, i);
        if (i == n) break;
        deq.front().first.first = i + 1;
        if (deq.front().first.second < i + 1) deq.pop_front();
        for (;!deq.empty() && trans(deq.back().second, deq.back().
    first.first) >= trans(i, deq.back().first.first); deq.pop_back
    ());
        if (deq.empty()) deq.push_back(make_pair(make_pair(i + 1, n
    ), i));

        else
        {
            int x = myLowBound(deq.back().first, deq.back().second,
     i);
            if (trans(i, x) >= (trans(deq.back().second, x))) x++;
            deq.back().first.second = x - 1;
            if (x <= n) deq.push_back(make_pair(make_pair(x, n), i)
    );
        }
    }
```

```
    cout << f[n] << endl;
    return 0;
}
```

Listing 2.1: hnoi2008toys.cpp

2.2 单调队列优化以及写仙人掌图

- 题目背景：仙人掌图上最长链
- 形式：f[i] = max(g[j]) + w[i]，w[i] 单调，可见，如果 j<k, g[j]<g[k], 则 j 可以直接不考虑，所以此时维护 g 单调减的队列即可。
- 仙人掌图找环：首先形成 bfs 树，发现有环，记 pt，ph，然后选 pt 沿着 pre 走到跟，一路打时间戳；再从 ph 沿着 pre 走，就可以找到 lca。pt，ph 到 lca 的路径，加上 pt->ph 就是基环了。

```
#include <bits/stdc++.h>
#define MAXN 1123456
#define MAXM 2123456

typedef int arrayN[MAXN], arrayM[MAXM];

using namespace std;

arrayN fir, cost, t, pre, vis;
arrayM e, nxt, c;
long long ans, dst[MAXN];
int num, now, visNow;

void link(int u, int v, int w)
{
    e[++num] = v, nxt[num] = fir[u];
    fir[u] = num, c[num] = w;
}

vector <int> bfsFindCycle(int x)
{
    ++now;
    vector <int> cyc;
    deque <int> deq;
    int pt = 0, ph = 0, last;
    deq.push_back(x);
    t[x] = now;
    for (; !deq.empty() && !pt;)
    {
        int u = deq.front();
        deq.pop_front();
        for (int p = fir[u]; p && !pt; p = nxt[p])
            if (e[p] != pre[u])
                if (t[e[p]] == now)
                {
                    pt = u, ph = e[p];
                    last = c[p];
                }
                else
                {
                    t[e[p]] = now;
                    pre[e[p]] = u;
                    cost[e[p]] = c[p];
                    deq.push_back(e[p]);
                }
    }
    vector <int> cycTmp;
    if (pt)
    {
        ++now;
        int tmp = pt;
        for (; tmp != x; tmp = pre[tmp])
            t[tmp] = now;
        t[x] = now;
        int lca = ph;
        for (; t[lca] != now; lca = pre[lca]);
        for (tmp = pt; tmp != lca; tmp = pre[tmp])
        {
            swap(last, cost[tmp]);
            cyc.push_back(tmp);
        }
        cyc.push_back(lca);
        cost[lca] = last;

        for (tmp = ph; tmp != lca; tmp = pre[tmp])
            cycTmp.push_back(tmp);
        for (; !cycTmp.empty(); cycTmp.pop_back())
            cyc.push_back(cycTmp.back());
    } else cyc.push_back(x);

    ++now;
    for (int i = 0; i < cyc.size(); ++i)
        t[cyc[i]] = now;
    return cyc;
```

```
}

struct node
{
    long long w;
    long long lst, f;}g[MAXN * 2];

long long bfsLongest(int rt, int &nrt)
{
    long long lst = 0;
    deque <int> deq;
    deq.push_back(rt);
    nrt = rt;
    vis[rt] = ++visNow;
    dst[rt] = 0;
    for (; !deq.empty(); )
    {
        int u = deq.front();
        deq.pop_front();
        for (int p = fir[u]; p; p = nxt[p])
            if (vis[e[p]] != visNow && t[e[p]] != now)
            {
                vis[e[p]] = visNow;
                dst[e[p]] = dst[u] + c[p];
                if (dst[e[p]] > lst)
                {
                    lst = dst[e[p]];
                    nrt = e[p];
                }
                deq.push_back(e[p]);
            }
    }
    return lst;
}

long long solve(int x)
{
    long long ans = 0;
    vector <int> cyc = bfsFindCycle(x);
    int n = cyc.size();
    for (int i = 0; i < n; ++i)
    {
        int pt, pp;
        t[cyc[i]] = 0;
        g[i].lst = bfsLongest(cyc[i], pt);
        ans = max(ans, bfsLongest(pt, pp));
        t[cyc[i]] = now;
        if (n == 1) return g[i].lst;
        g[i].w = cost[cyc[i]];
        g[i].f = 0;
        g[i + n] = g[i];
    }
    g[0].w = 0;
    for (int i = 1; i < 2 * n; ++i)
    g[i].w += g[i - 1].w;
    g[0].f = g[0].lst;
    deque <int> deq;
    deq.push_back(0);
    for (int i = 1; i < 2 * n; ++i)
    {
        for (; deq.front() + n <= i; deq.pop_front());
        g[i].f = g[i].lst + g[i].w + g[deq.front()].lst - g[deq.
    front()].w;
        for (; !deq.empty() && g[deq.back()].lst - g[deq.back()].w
    <= g[i].lst - g[i].w; deq.pop_back());
        deq.push_back(i);
    }
    for (int i = 0; i < 2 * n; ++i)
        ans = max(ans, g[i].f);
    return ans;
}

int main()
{
    freopen("island.in", "r", stdin);
    int n;
    num = 1;
    scanf("%d", &n);
    for (int i = 1; i <= n; ++i)
    {
        int v, len;
        scanf("%d%d", &v, &len);
        link(i, v, len);
        link(v, i, len);
    }
    long long ans = 0;
    for (int i = 1; i <= n; ++i)
        if (!vis[i])
            ans += solve(i);
    printf("%lld\n", ans);
    return 0;
```

```
}
```

Listing 2.2: ioi2008Island.cpp

2.3 斜率优化

- f[i] = min(a[i] * x[j] + b[i] * y[j])
- 更好的理解：设 P=f[i], 则 y = (-a/b)x + P/b. 求满足要求的最小截距。或者通过各种转化，最优决策就是从无穷远朝原点移动，第一个碰上的点为最优决策点。
- 很好的性质：所有最优决策一定在当前所有点构成的凸包上。（例如，在最优决策点划一条相应斜率的线，其余点均在该线上方，）

2.3.1 斜率以及 x 维都单调

想像斜率越来越大的直线往 y 正方向移动，第 i 次移动首次碰上 k。对于以后的决策，因为斜率更大，那么在 k 之前，第 i 次移动没有碰上的点必然再也用不上了，所以可以维护一个单调队列。下面例题是：把一个序列切开，每个部分权值是和平方加常数，求权值和最小值

```cpp
#include <deque>
#include <cstdio>
#include <cstring>
#include <iostream>
#include <cstdlib>

#define MAXN 512345

using namespace std;
typedef long long arrayN[MAXN];

struct node
{
    long long x, y, f;
    node (long long tx = 0, long long ty = 0, long long tf = 0)
    {
        x = tx, y = ty, f = tf;
    }
    //y = f + sum^2, x = sum
}g[MAXN];

long long sqr(long long x)
{
    return x * x;
}

long long cross(long long x1, long long y1, long long x2, long long y2)
{
    return x1 * y2 - x2 * y1;
}


deque < int > deq;

int main()
{
    freopen("hdu3507.in", "r", stdin);
    int N, M;
    for (; scanf("%d%d", &N, &M) != EOF; )
    {
        deq.clear();
        g[0] = node(0, 0, 0);
        deq.push_back(0);
        for (int i = 1; i <= N; ++i)
        {
            int x;
            scanf("%d", &x);
            g[i].x = g[i - 1].x + x;
            long long lim = g[i].x << 1;
            for (; deq.size() > 1; deq.pop_front())
            {
                node u = g[deq[0]];
                node v = g[deq[1]];
                if ((v.y - u.y) > lim * (v.x - u.x))
                    break;
            }
            node pt = g[deq.front()];
            g[i].f = pt.f + sqr(g[i].x - pt.x) + M;
            g[i].y = sqr(g[i].x) + g[i].f;
            for (; deq.size() >= 2; deq.pop_back())
            {
                node A = g[deq[deq.size() - 2]];
                node B = g[deq[deq.size() - 1]];
                node C = g[i];
                if (cross(B.x - A.x, B.y - A.y, C.x - B.x, C.y - B.y) > 0) break;
            }
            deq.push_back(i);
```

```
        }
        cout << g[N].f << endl;
    }
    return 0;
}
```

Listing 2.3: hdu3507.cpp

2.3.2 随便什么情况：cdq 分治优化

- 排序的顺序，凸壳的方向写之前一定要画清楚。
- 这里归并排一维的序可以节省一个 log 的复杂度
- cdq 分治的顺序至关重要，千万不能乱。
- f[i] 表示第 i 天手上的券全换成现金最多多少, 其中 x[j],y[j] 分别表示用 f[j] 的钱换成 A，B 券分别能有多少。
- f[i] = max(max(A[i] * x[j] + B[i] * x[j], f[j]))
- 就是经典的斜率优化问题咯。不用平衡树的话可以离线用 cdq 分治。先按照 -A[i]/B[i] 排序（具体大小顺序画一画就知道了）。solve(l, r) 时需要按照下标 lab 大小分为两部分。然后 solve(l,mid), 同时主义归并把递散维 x 排好序。l ~ mid 至 mid + 1 r 转移. 最后 solve(mid + 1, r)，接着归并排好 x 就行了。

```
#include <bits/stdc++.h>
#define MST(a, b) memset((a), (b), sizeof(a))
#define MAXN 112345
#define esp 1e-8

using namespace std;

struct node
{
    double A, B, rate; //A/B
    double x, y;
    double f;
}g[MAXN];

int lab[MAXN], a[MAXN];

int cmp(double x)
{
    if (x < -esp) return -1;
    if (x > esp) return 1;
    return 0;
}

int smaller(int u, int v)
{
    int tx = cmp(g[u].x - g[v].x);
    int ty = cmp(g[u].y - g[v].y);
    return tx < 0 || (tx == 0 && ty < 0);
}

void mergeSortX(int al, int ar, int bl, int br)
{
    int Na = 0;
    for (int i = al; i <= ar; ++i)
    {
        while (bl <= br && smaller(lab[bl], lab[i]))
            a[++Na] = lab[bl++];
        a[++Na] = lab[i];
    }
    for (; bl <= br; ++bl)
        a[++Na] = lab[bl];
    for (int i = 1; i <= Na; ++i)
        lab[al + i - 1] = a[i];
}


double cross(int A, int B, int C)
{
    return (g[B].x - g[A].x) * (g[C].y - g[B].y) - (g[B].y - g[A].y) * (g[C].x - g[B].x);
}

double comRate(int A, int B, int C)
{
    return (g[B].y - g[A].y) * g[C].B + g[C].A * (g[B].x - g[A].x);
}

void getRightPartF(int al, int ar, int bl, int br)
{
    int Na = 0;
    double lim = 0;
    for (int i = al; i <= ar; ++i)
    {
        lim = max(lim, g[lab[i]].f);
        while (Na >= 2 && cmp(cross(a[Na - 1], a[Na], lab[i])) >= 0)
```

```
            --Na;
        a[++Na] = lab[i];
    }
    int La = 1;
    for (int i = bl; i <= br; ++i)
    {
        int p = lab[i];        g[p].f = max(g[p].f, lim);
        for (; La + 1 <= Na && cmp(comRate(a[La], a[La + 1], p)) >= 0; ++La);
        g[p].f = max(g[p].f, g[a[La]].x * g[p].A + g[a[La]].y * g[p].B);
    }
}

void solve(int l, int r)
{
    if (l == r)
    {
        int p = lab[l];
        //g[p].f = max(g[p].f, g[p - 1].f);
        g[p].x *= g[p].f;
        g[p].y *= g[p].f;
        return ;
    }
    int Na = r - l + 1;
    int upLim = 0, downLim = MAXN;
    for (int i = l; i <= r; ++i)
    {
        upLim = max(upLim, lab[i]);
        downLim = min(downLim, lab[i]);
    }
    int midLim = (upLim + downLim) >> 1;
    int pLow = 0;
    for (int i = l; i <= r; ++i)
        if (lab[i] <= midLim)
            a[++pLow] = lab[i];
    int pHigh = pLow;
    for (int i = l; i <= r; ++i)
        if (lab[i] > midLim)
            a[++pHigh] = lab[i];
    for (int i = 1; i <= Na; ++i)
        lab[i + l - 1] = a[i];
    pLow += l - 1;
    solve(l, pLow);
    getRightPartF(l, pLow, pLow + 1, r);
    solve(pLow + 1, r);
    mergeSortX(l, pLow, pLow + 1, r);
}

int com(int u, int v)
{
    node tu = g[u];
    node tv = g[v];
    return tu.A * tv.B < tv.A * tu.B;
}

int main()
{
    //  freopen("cash4.in", "r", stdin);
    int N, S;
    scanf("%d%d", &N, &S);
    for (int i = 1; i <= N; ++i)
    {
        scanf("%lf%lf%lf", &g[i].A, &g[i].B, &g[i].rate);
        g[i].y = 1.0 / (g[i].B + g[i].A * g[i].rate);
        g[i].x = g[i].y * g[i].rate;
        g[i].f = S;
        lab[i] = i;
    }
    g[1].f = S;
    sort(lab + 1, lab + N + 1, com);
    solve(1, N);
    double ans = 0;
    for (int i = 1; i <= N; ++i)
        ans = max(ans, g[i].f);
    printf("%.3f\n", ans);
    return 0;
}
```

Listing 2.4: cash.cpp

Chapter 3

图论

3.1 tarjan

3.1.1 2-sat

3.1.2 割顶，点双联通分量

3.1.3 桥，边双联通分量

3.2 平面图

farmland 那道题

3.3 网络流

3.3.1 dinic

3.3.2 费用流

3.3.3 常见模型

3.4 弦图

3.5 最小树形图

Chapter 4

数据结构

Chapter 5

字符串

5.1 后缀数组

5.2 后缀自动机

Chapter 6

其他算法

6.1 cdq 分治与读入优化

- 不要排结构体，因为排结构体到时候还要排回来。
- 线段树打时间戳不要 memsize();
- 在严格小的限制下，第二维排序的时候一定要双关键字排序
- 这题是三维空间中，三个坐标都不减的最长链

```
#include <iostream>
#include <cstring>
#include <cstdlib>
#include <cstdio>
#include <algorithm>
#define REP(i, n) for(int i = 0; i < (int) (n); ++i)
#define REPP(i, a, b) for(int i = (int) (a); i <= (int) (b); ++i)
#define REDD(i, a, b) for(int i = (int) (a); i >= (int) (b); --i)
#define MST(a, b) memset((a), (b), sizeof(a))
#define MAXN 111111
#include <vector>

using namespace std;
int zLim;

long long gTot[MAXN *4];
int t, g[MAXN *4], n, ti[MAXN *4], now;
struct node
{
    int x, y, z, f;
    long long tot;
} a[MAXN];

int comx(node A, node B)
{
    return (A.x < B.x) || ((A.x == B.x) && (A.y < B.y)) || ((A.x == B.x) && (A.y == B.y) && A.z < B.z);
}

int comy(node A, node B)
{
    return A.y < B.y;
}

void change(int pos, int x, long long cnt)
{
    pos += t;
    if (ti[pos] != now) g[pos] = gTot[pos] = 0;
    if (x < g[pos]) return ;
    if (x == g[pos]) gTot[pos] += cnt;
    else gTot[pos] = cnt, g[pos] = x;
    ti[pos] = now;

    for(pos >>= 1; pos; pos >>= 1)
    {
        if (ti[pos <<1] != now) g[pos <<1] = gTot[pos <<1] = 0;
        if (ti[pos <<1 ^1] != now) g[pos <<1 ^ 1] = gTot[pos <<1 ^1] = 0;
        ti[pos] = now;
        g[pos] = max(g[pos <<1], g[pos << 1 ^1]);
        gTot[pos] = 0;
        if (g[pos] == g[pos <<1]) gTot[pos] += gTot[pos <<1];
        if (g[pos] == g[pos <<1 ^1]) gTot[pos] += gTot[pos <<1 ^1];
    }
}

int ask(int l, int r, long long &cnt)
{
    if (l > r) return 0;
    int tmp = 0;
    cnt = 0;
    l += t - 1, r += t + 1;
    for (;(l ^ r) != 1; l >>= 1, r >>= 1)
    {
        if (!(l &1))
        {
```

```
            if (ti[l +1] == now)
        {
            if (tmp == g[l +1]) cnt += gTot[l +1];
            else if (tmp <g[l +1])
            {
                tmp = g[l +1];                  cnt = gTot[l +1];
            }
        }
          }
          if (r &1)
          {
            if (ti[r - 1] == now)
        {
            if (tmp == g[r - 1]) cnt += gTot[r - 1];
            else if (tmp < g[r - 1])
            {
                tmp = g[r - 1];
                cnt = gTot[r - 1];
            }
        }
          }
    }
    return tmp;
}


void solve(int l, int r)
{
    if (l == r) return ;
    int mid = (l +r) >>1;
    solve(mid +1, r);

    sort(a +mid +1, a +r +1, comy);
    sort(a +l, a +mid +1, comy);

    // MST(g, 0);
    //MST(gTot, 0);
    ++now;
    int pos = r +1;
    REDD(i, mid, l)
    {
        for (;pos > mid +1 && a[pos - 1].y >= a[i].y; --pos)
        {
            change(a[pos - 1].z, a[pos - 1].f, a[pos - 1].tot);
        }

        long long tmpTot;
        int tmp = ask(a[i].z, zLim, tmpTot) +1;
        if (a[i].f == tmp) a[i].tot += tmpTot;
        else if (a[i].f < tmp)
        {
            a[i].f = tmp;
            a[i].tot = tmpTot;
        }
    }

    sort(a + l, a +r +1, comx);
    solve(l, mid);
}

int INT()
{
    int res;
    char ch;
    while (ch = getchar(), !isdigit(ch));
    for (res = ch - '0'; ch = getchar(), isdigit(ch);)
        res = res * 10 + ch - '0';
    return res;
}

int main()
{
    int task;
    freopen("in.txt", "r", stdin);
    now = 0;
    for (task = INT(); task; --task)
    {
        n = INT();
        vector <int> dataZ;
        REPP(i, 1, n)
        {
            a[i].x = INT();
            a[i].y = INT();
            a[i].z = INT();
            a[i].f = 1;
            a[i].tot = 1;
            dataZ.push_back(a[i].z);
        }
        sort(dataZ.begin(), dataZ.end());
        dataZ.resize(unique(dataZ.begin(), dataZ.end()) - dataZ.
    begin());
        REPP(i, 1, n)
```

```
156         {
157             a[i].z = (lower_bound(dataZ.begin(), dataZ.end(), a[i].
        z) - dataZ.begin()) +1;
158         }
159         zLim = dataZ.size();        for (t = 1; t <= zLim + 1; t
        <<= 1);
160         sort(a +1, a + n +1, comx);
161         solve(1, n);
162         int ans = 0;
163         long long cnt = 0;
164         REPP(i, 1, n)
165         {
166             if (ans == a[i].f) cnt += a[i].tot;
167             else if (ans < a[i].f) cnt = a[i].tot, ans = a[i].f;
168         }
169         printf("%d %lld\n", ans, cnt);
170     }
171     return 0;
172 }
```

Listing 6.1: hdu4742.cpp