《第六次上机实验》解题报告

1. 稀疏矩阵之差

1.1 题目描述

分数 100
作者 谷方明
单位 吉林大学

矩阵 A 和 B 都是稀疏矩阵。请计算矩阵的差 A-B.如果 A、B 不能计算差值，输出"Illegal!"

输入格式:

矩阵的输入采用三元组表示，先 A 后 B。对每个矩阵：
第 1 行，3 个整数 N、M、t，用空格分隔，分别表示矩阵的行数、列数和非 0 数据项数，10≤N、M≤50000，t≤min(N,M).
第 2 至 t+1 行，每行 3 个整数 r、c、v，用空格分隔，表示矩阵 r 行 c 列的位置是非 0 数据项 v, v 在 32 位有符号整型范围内。三元组默认按行列排序。

输出格式:

矩阵 A-B，采用三元组表示，默认按行列排序，非零项也在 32 位有符号整型范围内

1.2 思路

题目要求两点 1.作差；2.按行列序输出；
因为输入为按行列序的三元组表，所以只需设置两个指针顺序后移，按行列序依次作差即可；当差值为零时不存入；

1.3 代码：

```
#include<iostream>
using namespace std;
const int maxn = 50005;
```

```
int r1 = 0, c1 = 0, n1 = 0;
int r2 = 0, c2 = 0, n2 = 0;

typedef struct node {
    int r, c, v;
}Node;
Node mt1[maxn], mt2[maxn], dif[maxn];
int diff(Node Mat1[], Node Mat2[], Node diff[], int n1, int n2) {
    int p1 = 0, p2 = 0, pn = 0;
    while (p1 != n1 && p2 != n2) {
        if (Mat1[p1].r == Mat2[p2].r && Mat1[p1].c == Mat2[p2].c) {
            diff[pn] = mt1[p1];
            diff[pn].v = mt1[p1].v - mt2[p2].v;
            p1++; p2++;
            if (0!=diff[pn].v)pn++;
        }
        else if (Mat1[p1].r < Mat2[p2].r || (Mat1[p1].r == Mat2[p2].r &&
Mat1[p1].c < Mat2[p2].c)) {
            dif[pn] = mt1[p1];
            p1++,pn++;
        }
        else{
            dif[pn] = mt2[p2];
            dif[pn].v *= -1;
            p2++, pn++;
        }
    }
    for (; p1 != n1;p1++, pn++)
        diff[pn] = mt1[p1];

    for (; p2 != n2; p2++, pn++)
        dif[pn] = mt2[p2];

    return pn;
}

int main() {
    scanf("%d%d%d", &r1, &c1, &n1);

    for (int i = 0; i < n1; i++) {
        scanf("%d%d%d", &mt1[i].r, &mt1[i].c, &mt1[i].v);
    }
    scanf("%d%d%d", &r2, &c2, &n2);
```

```
    for (int i = 0; i < n2; i++) {
        scanf("%d%d%d", &mt2[i].r, &mt2[i].c, &mt2[i].v);
    }
    if (r1 != r2 || c1 != c2) {
        printf("Illegal!"); return 0;
    }
    int nn = diff(mt1, mt2, dif, n1, n2);
    printf("%d %d %d\n", r1, c1, nn);
    for (int i = 0; i < nn; i++) {
        printf("%d %d %d\n", dif[i].r, dif[i].c, dif[i].v);
    }
}
```

2. 二叉树最短路径长度

2.1 题目描述

分数 100
作者 谷方明
单位 吉林大学

给定一棵二叉树 T，每个结点赋一个权值。计算从根结点到所有结点的最短路径长度。路径长度定义为：路径上的每个顶点的权值和。

输入格式:

第 1 行，1 个整数 n，表示二叉树 T 的结点数，结点编号 1..n，1≤n≤20000。
第 2 行，n 个整数，空格分隔，表示 T 的先根序列，序列中结点用编号表示。
第 3 行，n 个整数，空格分隔，表示 T 的中根序列，序列中结点用编号表示。
第 4 行，n 个整数 Wi，空格分隔，表示 T 中结点的权值，-10000≤Wi≤10000，1≤i≤n。

输出格式:

1 行，n 个整数，表示根结点到其它所有结点的最短路径长度。

输入样例:

在这里给出一组输入。例如：
4
1 2 4 3

4 2 1 3
1 -1 2 3

输出样例:

在这里给出相应的输出。例如:
1 0 3 3

2.2 思路

本题关键在于利用中根、先根序建树;利用先根序的根节点将中根序分为两部分,此即左右子树,递归建树即可;
找最短路采用 dfs,累加即可;

2.3 代码

```
#include<iostream>

using namespace std;

typedef struct node {
    int key;
    struct node* ls, * rs;
}Node;
const int maxn = 20005;
int n;
int pre[maxn], in[maxn], w[maxn];
bool vis[maxn];
Node* build(int s, int e, int pi) {
    if (s > e)return NULL;
    Node* ret = new Node;
    ret->key = pre[pi];
    int r;
    for (r = s; r <= e && in[r] != pre[pi]; r++);
    ret->ls = build(s, r - 1, pi + 1);
    ret->rs = build(r + 1, e, pi + 1 + r - s);
    return ret;
}


void dfs(Node* root, int rw) {
    if (!root)return;
```

```
    int k = root->key;
    w[k] += rw;
    if (root->ls)dfs(root->ls, w[k]);
    if (root->rs)dfs(root->rs, w[k]);
}
int main() {
    scanf("%d", &n);
    for (int i = 0; i < n; i++)
        scanf("%d", &pre[i]);
    for (int i = 0; i < n; i++)
        scanf("%d", &in[i]);
    for (int i = 1; i <= n; i++)
        scanf("%d", &w[i]);
    Node* r = build(0, n - 1, 0);
    dfs(r, 0);
    for (int i = 1; i <= n; i++) {
        printf("%d", w[i]);
        if (i != n)printf(" ");
    }

}
```

3.文字编辑

3.1 题目描述

分数 100
作者 谷方明
单位 吉林大学

一篇文章由 n 个汉字构成，汉字从前到后依次编号为 1，2，……，n。
有四种操作：
A i j 表示把编号为 i 的汉字移动编号为 j 的汉字之前；
B i j 表示把编号为 i 的汉字移动编号为 j 的汉字之后；
Q 0 i 为询问编号为 i 的汉字之前的汉字的编号；
Q 1 i 为询问编号为 i 的汉字之后的汉字的编号。
规定：1 号汉字之前是 n 号汉字，n 号汉字之后是 1 号汉字。

输入格式:
第 1 行，1 个整数 T，表示有 T 组测试数据， 1≤T≤9999.
随后的每一组测试数据中，第 1 行两个整数 n 和 m，用空格分隔，分别代表汉字数和操作数，2≤n≤9999，1≤m≤9999；第 2 至 m+1 行，每行包含 3 个常量 s、i 和 j，用空格分隔，s 代表操作的类型，若 s 为 A 或 B，则 i 和 j 表示汉字的编号，若 s 为 Q，i 代表 0 或 1，j 代表汉字的编号。

输出格式:
若干行，每行 1 个整数，对应每个询问的结果汉字编号。

3.2 思路

有大量通过下标插入、删除操作，考虑使用静态链表，需要查找某一位的前后项、且首尾相接，因此使用双向循环链表。

3.3 代码

```
#include<iostream>

using namespace std;
const int maxn = 10005;
 struct node {
    int t,f,n;
}f[maxn];

int main() {
    int siz = 0, opnum = 0,n = 0;
    scanf("%d",&n);
    char oper[10];
    int x = 0, y = 0;
    for (int i = 0; i < n; i++) {
        scanf("%d%d", &siz, &opnum);
        for (int i = 1; i <= siz; i++)
            f[i] = { i,i-1,i + 1 };

        f[1].f= siz,f[siz].n = 1;
        for (int i = 0; i < opnum; i++) {
            scanf("%s %d %d", oper, &x, &y);
            switch (oper[0]) {
            case 'A': {
                f[f[x].f].n = f[x].n;
                f[f[x].n].f = f[x].f;
```

```
                f[x].f = f[y].f;
                f[x].n = y;

                f[f[y].f].n = x;
                f[y].f = x;
                break;
            }
            case 'B': {
                f[f[x].f].n = f[x].n;
                f[f[x].n].f = f[x].f;

                f[x].f = y;
                f[x].n = f[y].n;

                f[f[y].n].f = x;
                f[y].n = x;
                break;
            }
            case'Q': {
                if (0 == x)printf("%d\n", f[f[y].f].t);
                else   printf("%d\n", f[f[y].n].t);
                break;
            }
            }
        }
    }
}
```

4.方案计数

4.1 题目描述

分数 100
作者 谷方明
单位 吉林大学

组装一个产品需要 n 个零件。生产每个零件都需花费一定的时间。零件的生产可以并行进行。有些零件的生产有先后关系，只有一个零件的之前的所有零件都生产完毕，才能开始生产这个零件。如何合理安排工序，才能在最少的时间内完成所有零件的生产。在保证最少时间情况下，关键方案有多少种，关键方案是指

从生产开始时间到结束时间的一个零件生产序列，序列中相邻两个零件的关系属于事先给出的零件间先后关系的集合，序列中的每一个零件的生产都不能延期。

输入格式:
第 1 行，2 个整数 n 和 m，用空格分隔，分别表示零件数和关系数，零件编号 1..n，1≤n≤10000, 0≤m≤100000 。
第 2 行，n 个整数 Ti，用空格分隔，表示零件 i 的生产时间，1≤i≤n，1≤Ti≤100 。
第 3 到 m+2 行，每行两个整数 i 和 j，用空格分隔，表示零件 i 要在零件 j 之前生产。

输出格式:
第 1 行，1 个整数，完成生产的最少时间。
第 2 行，1 个整数，关键方案数，最多 100 位。
如果生产不能完成，只输出 1 行，包含 1 个整数 0.

4.2 思路

引入虚源虚汇，转为边权；在拓扑排序的中求出最大距离，同时求出到每个结点的方案数；某一点方案数，等于关键路径中所有到该点的点的方案数之和，若无则为 0，采用高精度计算。

4.3 代码

```
#include<iostream>
#include<queue>
#define cit(x,y,z)  for(int x=y;x<=z;x++)

using namespace std;
const int maxn = 1e4 + 5;
const int maxe = 1e5 + 5;
const int maxl = 103;

int n, e, ecnt = 0;
int head[maxn], in[maxn], out[maxn];
int ve[maxn], vl[maxn], cost[maxn];
int bn[maxn][maxl] = { 1,1 };
struct node {
    int to, w, next;
}edges[maxe];

inline void addedge(int x, int y, int w) {
    edges[ecnt] = { y,w,head[x] };
```

```
    head[x] = ecnt++;
}

void add(int x, int y) {
    int v = 0, i = 1;
    for (; i <= max(bn[x][0], bn[y][0]) + 1; i++) {
        bn[x][i] = bn[y][i] + bn[x][i] + v;
       (bn[x][i] >= 10) ? v = 1 ,bn[x][i] -= 10: v=0;
    }
    i--;
    bn[x][0] = (bn[x][i] > 0) ? i : i - 1;
}

bool topo() {
    queue<int>Q;
    Q.push(0);
    cit(j, 0, n) {
        if (Q.empty())return 0;
        int tmp = Q.front(); Q.pop();
        for (int i = head[tmp]; ~i; i = edges[i].next) {
            int near = edges[i].to; in[near]--;
            if (in[near] == 0)Q.push(near);
            if (ve[near] < ve[tmp] + edges[i].w) {
                ve[near] = ve[tmp] + edges[i].w;
                memset(bn[near],0,sizeof(bn[near]));
                add(near, tmp);
            }
            else if (ve[near] == ve[tmp] + edges[i].w)add(near, tmp);
        }
    }
    return 1;
}

int main() {
    scanf("%d%d", &n, &e);
    cit(i, 0, n + 1) head[i] = -1;
    cit(i, 1, n)scanf("%d", &cost[i]);
    int x, y;
    cit(i, 1, e) {
        scanf("%d%d", &x, &y);
        addedge(x, y, cost[x]);好像不用转更简便
        in[y]++, out[x]++;
    }
    cit(i, 1, n) {
```

```
        if (out[i] == 0)addedge(i, n + 1, cost[i]);
        if (in[i] == 0)addedge(0, i, 0), in[i]++;//虚源
    }
    if (!topo())printf("0");
    else {
        printf("%d\n", ve[n + 1]);
        for (int i = bn[n + 1][0]; i >= 1; i--)
            printf("%d", bn[n + 1][i]);
    }
    return 0;
}
```