**算法训练 Don't fear, DravDe is kind**

时间限制：2.0s   内存限制：256.0MB

问题描述

　　这一天，有一列车子排起了一排长队，必经之路是一个被魔王笼罩的山洞。每辆车的司机害怕魔王程度不同，所以每个司机有一些要求。  
　　车子有n台，排成一条长队，每辆车有4个属性:  
　　V ——这辆车的总价值，价值就是比如它其中的乘客和货物的价值  
　　c ——这辆车里面的人数量（司机表示自己也算一个乘客，司机和乘客不用区分开来）  
　　l ——在这辆车的前面需要总量正好为多少乘客的车（不多也不少），这车才敢开  
　　r ——在这辆车的后面需要总量正好为多少乘客的车（不多也不少），这车才敢开  
　　“前面需要总量正好为多少乘客的车”指的是驶在这辆车前面所有的车的乘客总数。  
　　“后面需要总量正好为多少乘客的车”指的是驶在这辆车后面所有的车的乘客总数。  
　　你不能改变每辆车在车队的相对顺序，但你可以安排某些车退出车队，保证依然在车队的每辆车都敢开了，即满足上述条件，并且剩下车的v的总量最大。  
　　-----------------------------  
　　简单来说，给您按输入顺序排列的n辆车，您需要删去里面的一些车（剩下的车仍然按原相对顺序排列）。  
　　使得对于每辆车，若它没被删去，设其为输入的第i辆车，  
　　要满足  
　　l[i]= sigma{c[j] | j<i 且第j辆车没被删去}  
　　r[i]= sigma{c[j] | j>i 且第j辆车没被删去}  
　　在满足这些条件前提下，要求sigma{V[i] | i没被删去} 最大，  
　　请输出这个最大值，并且递增输出没有被删去的车的标号。

输入格式

　　输入的第一行为一个正整数n（1<=n<=10^5）——车的个数。  
　　接下来n行，每行四个整数，第i行的数字: vi, ci,li ,ri ,（1<=vi<=10^4 , 1<=ci<=10^5,0<=li,ri<=10^5），车子们从1开始编号，从车队的最前头开始算起。

输出格式

　　第一行输出一个数k：会继续在这车队里的车的总数（注意我们的目标是让价值最大）。  
　　第二行k个数，递增输出继续在车队里的车的编号。  
　　请留心你不允许改变车的次序。如果答案不唯一，输出任意一个。

样例输入

5  
1 1 0 3  
1 1 1 2  
1 1 2 1  
1 1 3 0  
2 1 3 0

样例输出

4  
1 2 3 5

样例输入

5  
1 1 0 3  
10 1 2 1  
2 2 1 1  
10 1 1 2  
3 1 3 0

样例输出

3  
1 3 5

数据规模和约定

　　对于20%的数据，n<=100  
　　对于50%的数据，n<=1000  
　　对于100%的数据，n<=100000  
　　对于100%的数据，1<=vi<=10^4 , 1<=ci<=10^5,0<=li,ri<=10^5

C++:

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <math.h>

#include <set>

#include <vector>

#include <map>

#define N 100005

using namespace std;

struct node{

int v,c,l,r;

int num;

}car[N];

int n;

vector<int>G[N], tmp;

set<int> myset;

set<int> ::iterator p;

map<int,int> mymap;

map<int,int> per;

map<int,int> sig;

int pos[N], endpos;

int work(int x){

if(G[x].size()==0)

return 0;

mymap.clear();

per.clear();

sig.clear();

mymap[0] = 0;

per[0] = -1;

sig[0] = -1;

int ans = 0;

endpos = -1;

int i;

for(i = 0; i < G[x].size(); i++) {

node now = car[G[x][i]];

if(mymap.find(now.l)==mymap.end())

continue;

if(mymap.find(now.l+now.c)==mymap.end() ||

mymap[now.l+now.c]<mymap[now.l]+now.v) {

mymap[now.l+now.c] = mymap[now.l]+now.v;

sig[now.l+now.c] = G[x][i];

per[G[x][i]] = sig[now.l];

}

if(now.r==0 && ans<mymap[now.l+now.c])

ans = mymap[now.l+now.c], endpos = G[x][i];

}

return ans;

}

int main(){

int i,j,u,v;

scanf("%d",&n);

for(i = 1; i <= n; i++)

G[i].clear();

myset.clear();

mymap.clear();

for(i = 1; i <= n; i++)

scanf("%d%d%d%d",&car[i].v,&car[i].c,&car[i].l,&car[i].r), car[i].num = i;

for(i = 1; i <= n; i++) {

myset.insert(car[i].c+car[i].l+car[i].r);

}

i = 1;

for(p = myset.begin(); p!=myset.end(); p++, i++)

mymap[\*p] = i, pos[i] = \*p;

for(i = 1; i <= n; i++)

G[mymap[car[i].c+car[i].l+car[i].r]].push\_back(i);

int ans = 0;

for(i = 1; i <= n; i++) {

int now = work(i);

if(now>ans) {

ans = now;

tmp.clear();

u = endpos;

while(u!=-1) {

tmp.push\_back(u);

u = per[u];

}

}

}

printf("%d\n",tmp.size());

for(i = tmp.size()-1; i>=0; i--)

printf("%d%c",tmp[i],i?' ':'\n');

system("pause");

return 0;

}