CEOI'2012 Day2, Task: wagons

First consider the special case when each setting contains only one type. For simplicity we write the type identifier instead of the setting.
Let

$$x_1, \ldots, x_i, \ldots, x_n$$

be the input sequence of types. We will describe an algorithm that computes the longest initial subsequence $x_1, \ldots, x_i$ for given types a,b and c if the setting for the first, second and third day is a, b and c respectively.
The possible configurations that can appear during the process can be described by triples as follows:

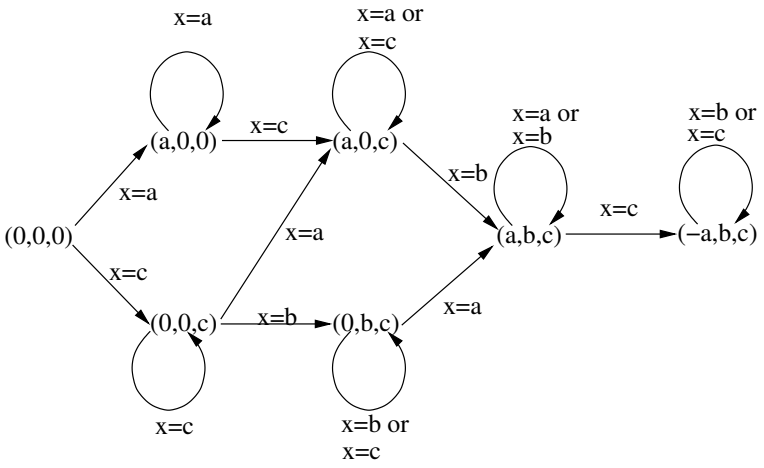

Figure 1: Transition diagram for the configurations of triples of types.

(0,0,0) : The initial configuration; no wagon moved from the incoming track.

(a,0,0) : All items so far was of type a and processed by the first day's setting a.

(0,0,c) : All items so far was of type c and moved to the auxiliary track.

(0,b,c) : Items of type c followed by items of type b moved to the auxiliary track.

(a,b,c) : Items of type a processed by the first day's setting a, items of type b and c moved to the auxiliary track such that the auxiliary track contains items of type c followed by items of type b from bottom to top.

(-a,b,c) : The current setting is the setting for the second day, that is b, the auxiliary track contains items of type c and items of type a already processed.

For each configuration and current item x ($x = a$ or $x = b$ or $x = c$) we can give the next possible configuration. The transition diagram below shows all possibilities. Note that a move from (0,b,c) to (b,a,c) by x=a is also possible, but it is covered by the move (0,0,0) -c-> (0,0,c) -b-> ((b,0,c) -a-> (b,a,c).
The sequence of items $x_1, \ldots, x_i$, can be processed by setting of first day = a, second day = b and third day = c if and only if the sequence leads from the initial configuration to some configuration. Moreover, if the final configuration has two 0-s, then the input can be processed in a single day, if it has two 0-s then it can be processed in two days, and conversely.
The general case when each setting can contain several types is similar, the difference is that test $x \in A$ stands instead of $x = a$. Therefore for given settings A, B and C we can compute the longest initial subsequence of the input which can be processed by settings A, B and C. The running time of this algorithm will be $\mathbf{O}(n \cdot s^3)$ where s is the number of the settings.
We can do it faster. Remember that for each type there are at most 10 settings containing that type. The idea is the following. Instead of configuration triples we maintain the set of triples (A, B, C) of settings that can be obtained from the initial configuration by all possible transitions for the input subsequence. The initial set is

$$\{(A, 0, 0) : x_1 \in A\} \cup \{(0, 0, C) : x_1 \in C\}$$

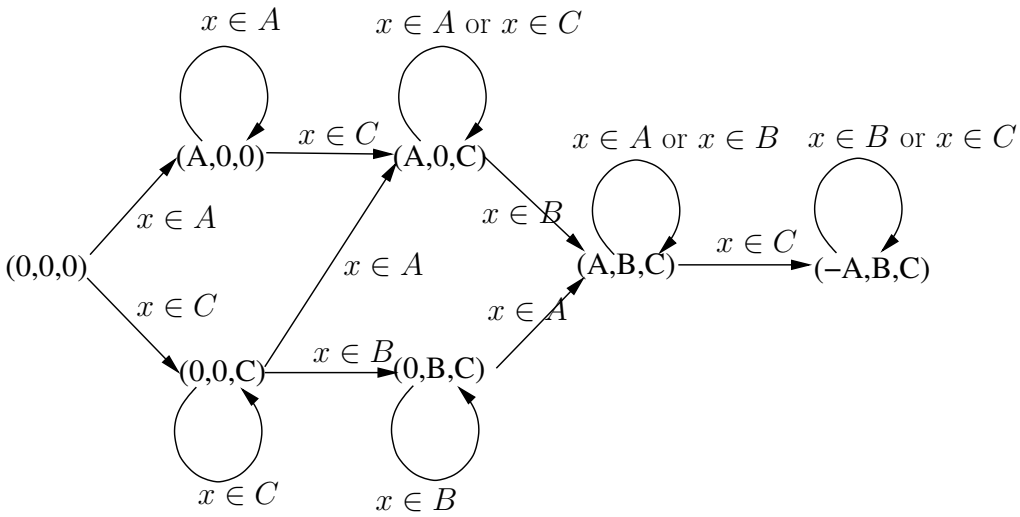

Figure 2: Transition diagram for the configurations of triples of settings.

If S is the current set of configurations and x is the current input item (type) then the next set is:

$$\{(A, B, C) : \exists(\overline{A}, \overline{B}, \overline{C}) \in S \text{ such that there is a transition from } (\overline{A}, \overline{B}, \overline{C}) \text{ to } (A, B, C) \text{ by } x\}$$

The worst case running time of this algorithm is $\mathbf{O}(n \cdot l^3)$, where for each type there are at most l settings containing that type.

Implementation

```
#include <stdio.h>
#include <stdlib.h>

#define maxK 1001
#define maxS 1001
#define maxQ 10000
using namespace std;
typedef struct triple{
    int a,b,c;
}triple;
triple Q[maxQ];

bool P[maxS][maxK];
int KS[maxS][maxK];
int qfirst=0,qlast=0,qnext=0;

void put(int x,int y,int z){
   triple t;
   t.a=x; t.b=y; t.c=z;
   Q[qnext]=t;
   qnext=(++qnext)%maxQ;
}
triple get(){
   triple t=Q[qfirst];
   qfirst=(++qfirst)%maxQ;
   return t;
}
```

```
28 int main(){
29     int n,kn,sn,X,y,sx;
30     int a=0,b=0,c=0,qfirst0=0;
31     triple t;
32
33     scanf("%d %d %d",&n, &kn, &sn);
34     for (int kx=1;kx<=kn;kx++) KS[kx][0]=0;
35     for (sx=1;sx<=sn;sx++)
36         for (int kx=1;kx<=kn;kx++) P[sx][kx]=false;
37     for (sx=1;sx<=sn;sx++)
38         while (true){
39             scanf("%d",&y);
40             if (y==0) break;
41             P[sx][y]=true;
42             KS[y][++KS[y][0]]=sx;
43         }
44     scanf("%d",&X);
45     for (int j=1;j<=sn;j++)
46         if (P[j][X]){
47             put(j,0,0); put(0,0,j);
48         }
49     qlast=qnext;
50     int i=2;
51     while (i<=n){
52         scanf("%d",&X);
53         qfirst0=qfirst;
54         while (qfirst!=qlast){
55             t=get();
56             if (t.b==0 && t.c==0){//1.
57                 if (P[t.a][X])
58                     put(t.a,0,0);
59                 else
60                     for (int j=1;j<=KS[X][0];j++)
61                         put(t.a,0,KS[X][j]);
62             }else if (t.a==0 && t.b==0){//2.
63                 if (P[t.c][X])
64                     put(0,0,t.c);
65                 else{
66                     for (int j=1;j<=KS[X][0];j++){
67                         put(KS[X][j],0,t.c);
68                     }
69                 }
70             }else if (t.a!=0 && t.b==0 && t.c!=0) {//3.
71                 if (P[t.a][X] || P[t.c][X]){
72                     put(t.a,t.b,t.c);
73                 }else{
74                     for (int j=1;j<=KS[X][0];j++)
75                         put(t.a, KS[X][j], t.c);
76                 }
77             }else if(t.a>0 && t.b!=0 && t.c!=0){//4.
78                 if (P[t.a][X] || P[t.b][X])
79                     put(t.a,t.b,t.c);
80                 else if (P[t.c][X]){
81                     put(-t.a,t.b,t.c);
82                 }
83             }else if (t.a<0 && t.b!=0 && t.c!=0){//5.
84                 if (P[t.b][X] || P[t.c][X])
85                     put(t.a,t.b,t.c);
86             }
```

```
87          }//while
88          if (qlast==qnext){
89              break;
90          }else{
91              qfirst=qlast;
92              qlast=qnext;
93          }
94          i++;
95      }//for i
96      qfirst=qfirst0;
97      t=get();
98      a=t.a;b=t.b;c=t.c;
99      while(qfirst!=qlast){
100         t=get();
101         if (t.b==0 && t.c==0){
102             a=t.a;b=t.b;c=t.c;
103             break;
104         }else if (t.b==0 || t.c==0){
105             a=t.a;b=t.b;c=t.c;
106         }
107     }
108     printf("%d\n",i-1);
109     if (b==0){b=c; c=0;}
110     printf("%d %d %d\n",abs(a),abs(b),abs(c));
111
112     return 0;
113 }
```