

ПРИМЕР

Вход

```
6 10
3 2 2 4
5 0
3 0
2 2 3 5
15 1 6
20 0
```

Изход

```
24
```

Решение

Складът за който става дума може да бъде представен като неориентиран граф – залите са върховете, а тунелите – ребрата. От условието е ясно, че този граф е дърво – съществуването на път от върха, съответен на входа, до всеки друг връх означава свързаност, а единствеността на този път – отсъствие на цикли. Обявявайки входния връх за корен, превръщаме дървото в кореново. Така, съседите на всеки връх, зададени във входа, са синовете му в кореновото дърво. Като използваме наредбата на върховете зададена във входа, можем да въведем наредба на синовете в кореновото дърво.

За решаването на задачата ще използваме едно от по-малко популярните, но много удобно за случая, представяне на кореновите дървета с наредба на синовете – „най-ляв син – десен брат“. Това представяне е толкова компактно, колкото и популярното представяне на двоични коренови дървета с наредба на синовете – „ляв син – десен син“, защото за всеки връх помним два други върха – най-левия от синовете му (ако изобщо има синове) и следващият в наредбата, т.е. братът отдясно (ако има такъв). При въведената номерация на върховете, за означаване на отсъствието на най-ляв син или десен брат използваме 0. Първата част на решението построява исканото представяне.

За намиране на минималното време (или на минималния път, което е същото, е достатъчно да обходим полученото кореново дърво в „постордер“, т.е. обхождането на всеки връх ще извършим тогава, когато са обходени всички върхове на поддървото, на което той е корен. Така кашоните от върховете на всяко поддървото ще се съберат в корена му и могат да бъдат изнесени от там при оптимално натоварване на електрокара. Обхождането започваме от корена с дължина на пътя 0. Под обхождане на текущия връх v ще разбираме следното:

- ако v е с необходим най-ляв син – отлагаме обхождането на v за по-късно, преместваме се в най-левия му син и добавяме единица в дължината на търсения път;
- ако v няма синове или най-левият му син е обходен – изнасяме всички кашони в бащата на v , като добавяме $2 \cdot \lceil K_v/M \rceil - 1$ към дължината на изминатия път и обявяваме v за обходен. Ако v има десен брат, той става текущ връх и добавяме единица в дължината на търсения път, а ако няма – нов текущ е бащата на v ;
- когато се окажем отново в корена на дървото, обхождането е завършено – т.е. всички кашони са пренесени до корена и с това задачата е решена.

Един недостатък на представянето „най-ляв син – десен брат“ е, че в него трудно се намира бащата на зададен връх. Това може да се преодолее с още едно поле за всеки възел, в което е записан неговият баща. В представената програма се използва стек, в който държим всички върхове по пътя от корена до текущия връх и с негова помощ намираме бащата елементарно. Тъй като за броя M на ребрата на всяко дърво е в сила $M = N - 1$, сложността на този алгоритъм е $O(N)$.

```
#include <stdio.h>
#define MAXN 10001
int main()
{
    int N,M,L=0;
    int lson[MAXN],rbro[MAXN],q[MAXN],st[MAXN],sp;
    int i,j,s,x,y,a,b;
    scanf("%d %d",&N,&M);
    for(i=1;i<=N;i++)
    {
        scanf("%d %d",&q[i],&s);
        if(s==0)
            lson[i]=0;
        else
        {
            if(s==1)
            {
                scanf("%d",&x);lson[i]=x;rbro[x]=0;
            }
            else
            {
                scanf("%d",&x);lson[i]=x;y=x;
                for(j=2;j<=s;j++)
                {
                    scanf("%d",&x);rbro[y]=x;y=x;
                }
                rbro[y]=0;
            }
        }
    }
    //postorder na "lson-rbro"
    st[0]=1;sp=0;
    while(1)
    {
        x=st[sp]; //x e varhat na steka
        y=lson[x]; // y e leviat mu sin
        if(y!=0) //ako ima lyav sin y
        {
            st[++sp]=y;L++;
        }
        else // x niama liav sin
        {
            //iznasiane do bashtata
            a=q[x]/M; b=q[x]%M;
            q[st[sp-1]]+=q[x];
            L+=2*a-1;if(b>0) L+=2;
            y=rbro[x];
            if(y!=0) //x ima desen brat
            {
                st[sp]=y;L++;
            }
        }
    }
}
```