# АНАЛИЗ НА РЕШЕНИЕТО НА ЗАДАЧА
# ИЗГУБЕН ПЪТ

Има три възможности за липсващото картонче – да съдържа името на началния град, да е вътрешно за маршрута или да съдържа името на крайния град. Да означим тези три случая с 0, 1 и 2 и да запомним в променливата `int c` кой от тях имаме в данните. Нека функцията `from_start()` строи в масива `int path[]` маршрута от град `0` докато достигне до липсващото картонче, а функцията `from_end()` строи в масива `int path[]` маршрута от град `K` назад, докато достигне до липсващото картонче. В случай 1, след изпълнението на двете функции маршрутът ще бъде построен изцяло. В случай 0 не е необходимо да се изпълнява първата функция, а в случай 2 – втората:

```
#define MAXN 1000001
int path[MAXN],K;
int main()
{  int a,b,c;
   cin>>K;
   <въвеждане на данните и определяне на случая>
   if(c!=0) from_start();
   if(c!=2) from_end();
   for(int i=0;i<=K;i++)
      cout<<path[i]<<" ";
   cout<<endl;

}
```

Трудното в тази задача е да си организираме данните. Ако постъпим по елементарния начин – в масив `cards` с $K-1$ реда и два стълба запишем във всеки ред двете числа от едно от останалите картончета, тогава за възстановяване на всяка стъпка от маршрута ще трябва да претърсим по-голяма или по-малка част от масива `cards`, което ще означава сложност $O(K^2)$, недопустима при входове с 100000 града.

Затова постъпваме така – в `i`-тия ред на масива `cards[MAXN][2]` записваме тези градове (0, 1 или 2), които участват в едно и също картонче с града `i`. Ако в картончетата сме намерили само едно картонче с `i`, тогава на второто място в реда поставяме -1, а ако градът изобщо не се среща в картонче – и двете стойности в реда са -1.

```
int cards[MAXN];
...
for(i=0;i<=K;i++)cards[i][0]=cards[i][1]=-1;
for(int i=1;i<K;i++)
{  cin >> a>>b;
   if(cards[a][0]==-1) cards[a][0]=b;
   else cards[a][1]=b;
   if(cards[b][0]==-1) cards[b][0]=a;
   else cards[b][1]=a;
}
```

Началото и краят на пътя са фиксирани:

```
path[0]=0; path[K]=K;
```

а в кой от трите случая сме определяме като разгледаме какво е записано в реда на града 0 и града $K$:

```
c=1;
if(cards[0][0]==-1) c=0;
else if(cards[K][0]==-1) c=2;
```

Остава да напишем функциите `from_start()` и `from_end()`:

```
void from_start()
{  path[1]=cards[0][0];
   for(int i=2;i<=K;i++)
   { if(cards[path[i-1]][0]!=path[i-2])
       path[i]=cards[path[i-1]][0];
     else
       if(cards[path[i-1]][1]!=-1)
          path[i]=cards[path[i-1]][1];
       else break;
    }
}
```

и подобно

```
void from_end()
{  path[K-1]=cards[K][0];
   for(int i=K-2;i>=0;i--)
   { if(cards[path[i+1]][0]!=path[i+2])
       path[i]=cards[path[i+1]][0];
     else
       if(cards[path[i+1]][1]!=-1)
          path[i]=cards[path[i+1]][1];
       else break;
    }
}
```

Така получаваме програма, която решава задачата с линейна сложност.

*Автор: Красимир Манев*