9. feladat: Puffer (50 pont)

Számítógépes hálózatokban az üzeneteket csomagokra darabolva továbbítják. A csomagok nem feltétlenül az elküldés sorrendjében érkeznek a címzetthez. Hogy össze lehessen állítani az eredeti csomagsorrendet, a beérkező csomagokat ideiglenes tárolóban, pufferben kell tárolni. Tegyük fel, hogy olyan csomag érkezik, amely az üzenetnek az ***a*** sorszámú byte-jától a ***b*** sorszámú byte-jáig terjedő részét tartalmazza. Ekkor két lehetőség van. Ha már megérkezett és továbbításra került az üzenetnek az ***a-1***-sorszámú byte-ja, akkor a csomagot nem kell pufferbe rakni, hanem közvetlenül outputra tehetjük, és a pufferből kivesszük a csatlakozó csomagokat. Ha még nem érkezett meg az üzenet ***a-1***-sorszámú byte-ja, akkor a csomagot pufferbe kell rakni.

Készíts programot (PUFFER.PAS, PUFFER.C vagy PUFFER.CPP), amely kiszámítja, hogy minimálisan mekkora méretű puffer kell!

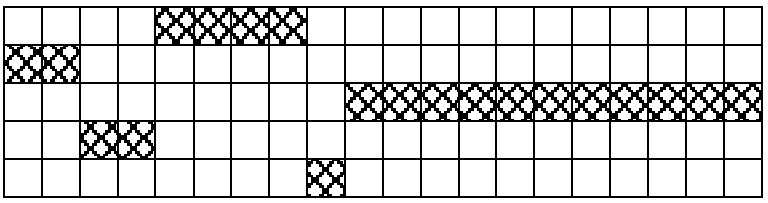
A PUFFER.BE állomány első sorában a csomagok ***N*** száma (***1≤N≤300000***) van. A következő ***N*** sor mindegyikében két egész szám van, ***a*** és ***b,*** ami egy csomag első és utolsó byte-jának sorszáma (***1≤u≤v≤100 000 000)***.

A PUFFER.KI állomány első sorába azt a legkisebb ***K*** számát kell írni, amelyre teljesül, hogy a csomagokat össze lehet rakni ***K***-méretű pufferrel!

Példa:

PUFFER.BE
```
5
5 8
1 2
10 20
3 4
9 9
```

PUFFER.KI
```
15
```



2. feladat: Robotverseny (30 pont)

Robotokat versenyeztetnek egy olyan pályán, amely kijelölt pontokból és bizonyos pontpárokat összekötő egyenes pályaszakaszokból áll. Ha a ***p*** és ***q*** pontot összeköti közvetlenül pályaszakasz, akkor azt mondjuk, hogy ***p*** és ***q*** szomszédok. Minden robot egy lépésben egy időegység alatt szomszédos pontba léphet, betartva a következő szabályokat. A ***p*** pontból csak akkor léphet a ***q*** szomszédos pontba, ha

1) a ***q*** pontban még nem járt egyetlen robot sem,
2) ugyanazon pontba nem léphet egyszerre több robot, azaz csak akkor léphet ***q***-ba, ha **q**-nak nincs olyan ***r*** szomszédja, ahol éppen robot tartózkodik,
3) ha egy robot valamikor nem tud lépni, akkor utána már sohasem léphet.

A verseny során egyik robot sem tudja, hogy a többi éppen hol tartózkodik és milyen útvonalon érkezett oda. Ezért a robotokat úgy programozták, hogy biztonságos útvonalon haladjanak, tehát bárhogy is haladt a többi robot az adott időpontig, az aktuális lépése biztosan szabályos lesz. Minden robot arra törekszik, hogy a kijelölt célpontba érjen a lehető legrövidebb idő alatt.

Készíts programot (ROBOT.PAS, ROBOT.C vagy ROBOT.CPP), amely kiszámítja, hogy melyik robot ér először a célba, és mennyi idő alatt!

A ROBOT.BE állomány első sorában a pályapontok ***N*** száma (***1≤N≤10000***), a robotok ***K*** (***1≤K≤1000***) száma, valamint a célpont ***C*** azonosítója van. A pályapontokat az ***1,…N*** számokkal azonosítjuk. A második sorban pontosan ***K*** különböző egész szám van (egy-egy szóközzel elválasztva), a ***K*** darab robot kezdeti pályapontja. A következő ***N*** sor írja le a a pályát. Az állomány ***i+2***-edik sorában azok a csomópontok vannak felsorolva egy-egy szóközzel elválasztva és ***0***-val zárva, amelyek az ***i*** pályapont közvetlen szomszédjai A közvetlen pályaszakaszok száma legfeljebb ***100000***.

A ROBOT.KI állomány első sorába azt a ***D*** legkisebb időt kell írni, amennyi idő alatt valamelyik robot célba ér. A második sorba egy legrövidebb idő alatt célba érkező robot útvonalát kell kiírni. Ha egyik robot sem tud célba jutni a szabályok betartásával, akkor az első és egyetlen sorba a ***0*** számot kell kiírni!

Példa:

ROBOT.BE
```
10 3 10
1 2 3
4 5 0
5 6 7 3 0
7 2 0
5 1 10 0
8 1 2 9 0
2 9 10 0
2 3 10 0
5 10 0
5 6 0
8 4 6 0
```

ROBOT.KI
```
4
1 4 5 8 10
```