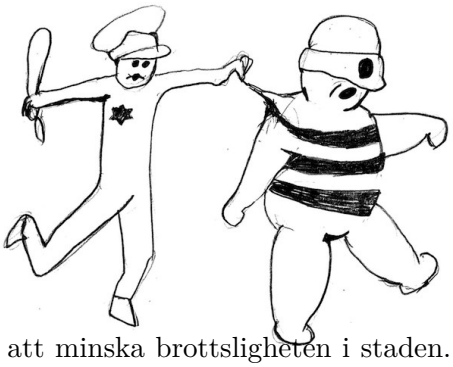
Tjuv och polis



I staden Bitköping så är kriminalitet ett stort problem. Varje dag sker allt från småbrott till väpnade rån. När ett brott har skett så är det alltid upp till en ensam patrullerande polis att jaga tjuven genom de smala gator som binder ihop gatukorsningarna i staden. Tyvärr så lyckas tjuvarna dock fly större delen av tiden, eftersom de känner till staden så mycket bättre än vad poliserna gör.

Bitköpings Poliskår (BP) organiserar nu en kampanj för att minska brottsligheten i staden. Ett av initiativen är att använda sig av datorhjälp vid jakten på tjuvarna. För detta ändamål så har en detaljerad karta av staden skapats; nu saknas bara mjukvara för att hitta jaktstrategier.

Jaktproblemet (där 1 polis jagar 1 tjuv) kan modelleras på följande sätt:

1. Polisen väljer en korsning att patrullera på.
2. Tjuven väljer sedan en korsning där han utför ett rån (han vet var polisen befinner sig). Från den här tiden och framåt vet både polisen och tjuven var den andra befinner sig.
3. Polisen gör ett drag som består i att antingen förflytta sig till en närliggande korsning (en som är ihopkopplad till den nuvarande korsningen via en gata) eller stå stilla.
4. Tjuven gör också ett drag som består i att förflytta sig till en närliggande korsning. Tjuven har dock inte möjligheten att stå stilla (eftersom det skulle gå emot tjuv-instinkten att alltid fortsätta springa).
5. Polisen och tjuven fortsätter att göra sina drag tills någon av följande saker händer:
   (a) En situation upprepar sig själv (en situation består av polisens och tjuvens positioner samt vem som ska göra nästa drag). Detta betyder att tjuven lyckas undvika polisen i all oändlighet, och lyckas därför fly.
   (b) Polisen och tjuven möts vid en korsning efter ett drag av någon av dem. I det här fallet lyckas polisen fånga tjuven.

Uppgift

Din uppgift är att skriva ett program som, givet en karta av staden, kan avgöra om det är möjligt att fånga tjuven. Om detta är möjligt ska du även styra polisen på så sätt att tjuven åker fast.

Ditt program måste anta att tjuven förflyttar sig optimalt.

Implementation

Du behöver implementera två funktioner:

- `start(N, A)` som tar följande 2 parametrar:

- N — antalet korsningar i staden (korsningar indexeras från 0 till $N-1$)
- A — en två-dimensionell array som beskriver stadens gator: för $0 \leq i, j \leq N-1$,

$$A[i,j] \text{ är } \begin{cases} \texttt{true} & \text{om } i \text{ och } j \text{ är ihopkopplade med en gata} \\ \texttt{false} & \text{annars} \end{cases}$$

Alla gator går att passera åt båda hållen ($A[i,j] = A[j,i]$ för alla värden på i och j) och det finns inga vägar som kopplar ihop en korsning med sig själv ($A[i,i]$ kommer vara $false$ för alla värden på i). Du kan också anta att det alltid är möjligt att nå alla andra korsningar från varje given korsning.

Om det är möjligt att fånga tjuven på kartan beskriven av parametrarna, så ska funktionen `start` returnera index för den gata som polisen borde börja patrullera på. Om det är omöjligt att fånga tjuven, oavsett vilken korsning polisen väljer, så ska -1 returneras.

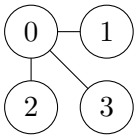
- `nextMove(R)` som tar som parameter indexet R på korsningen där tjuven just nu befinner sig, och returnerar korsningen som polisen ska flytta till.

Funktionen `start` kommer att anropas precis en gång innan alla anrop till `nextMove` görs. Om `start` returnerar -1 så anropas aldrig `nextMove`. I annat fall så kommer upprepade anrop till `nextMove` att ske tills jakten är över. Mer specifikt så kommer programmet att terminera så fort något av följande inträffar:

- `nextMove` returnerar ett ogiltigt drag
- en upprepad situation uppstår
- tjuven blir fångad

Exempel

Låt oss ta en titt på exemplet illustrerat till höger. I det här fallet så är alla korsningar bra startpositioner för polisen. Om hon startar i korsning 0, så kan hon stå stilla under första draget och låta tjuven springa in i henne. Å andra sidan, om hon börjar i någon annan korsning så kan hon vänta tills tjuven når korsning 0 och sedan flytta dit.



Här är hur en exempel-session skulle kunna se ut:

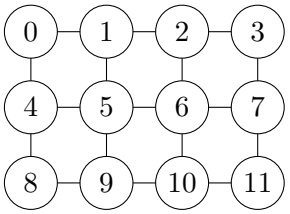
Funktionsanrop	**Returvärde**
`start(4, [[0, 1, 1, 1], [1, 0, 0, 0], [1, 0, 0, 0], [1, 0, 0, 0]])`	3
`nextMove(1)`	3
`nextMove(0)`	0

Anmärkning: i anropet till `start` ovanför så har `true` och `false` bytts ut mot `1` och `0` för att ta mindre plats.

Poängsättning

Deluppgift 1 (16 poäng): $2 \leq N \leq 500$. För varje par av korsningar kommer det kommer att finnas exakt en väg som leder mellan dem.

Deluppgift 2 (14 poäng): $2 \leq N \leq 500$. Nätverket av korsningar och gator kommer att bilda en rutnätsformad struktur. Rutnätet kommer att ha minst två rader och kolumner och korsningsnumreringen kommer att följa mönstret som illustreras i figuren nedan.



Deluppgift 3 (30 poäng): $2 \leq N \leq 100$.

Deluppgift 4 (40 poäng): $2 \leq N \leq 500$.

Din lösning skall uppfylla två krav:

1. korrekt bestämma huruvida polisen kan fånga tjuven;
2. om detta är möjligt, fånga tjuven genom att göra drag åt polisen.

I deluppgifter 1 och 2 så måste din lösning uppfylla båda kraven för att få några poäng. I deluppgifter 3 och 4 så kommer lösningar som bara implementerar det första kravet att få 30% av deluppgiftens poäng. Om din lösning enbart satser på delpoäng så kan du terminera programmet genom att utföra vilket ogiltigt drag som helst (t.ex. kan du returnera -1 från `nextMove`).

Notera att standardkraven (tids- och minnesgränser, inga körtidsfel) fortfarande måste uppfyllas för att få några poäng.

Begränsningar

Tidsgräns: 1.5 s.

Minnesgräns: 256 MB.

Experimentation

Exempelgradern på din dator kommer att läsa data från standard input. Första raden av indata ska innehålla ett heltal N — antalet korsningar. Följande N rader ska innehålla närhetsmatrisen A. Varje rad ska innehålla N tal, vardera antingen 0 eller 1. Matrisen måste vara symmetrisk, och värdena på huvuddiagonalen måste alla vara nollor.

Nästa rad ska innehålla talet 1 om polisen kan fånga tjuven, och 0 annars.

Till slut, om polisen kan fånga rånaren, ska det följa N rader, som beskriver tjuvens flyktstrategi. Varje rad ska innehålla $N + 1$ heltal mellan 0 och $N - 1$. Värdet på rad r och kolumn c, där $c < N$, motsvarar en situation där det är tjuvens tur, polisen befinner sig vid korsning r och tjuven vid korsning c, och representerar korsningen som tjuven då ska röra

sig till. Värdena på huvuddiagonalen kommer att ignoreras, då de motsvarar situationer då polis och tjuv befinner sig i samma korsning. Det sista värdet på rad r beskriver tjuvens startkorsning om polisen startar i korsning r.

Här är exempeldata till exempelgradern som representerar tre sammankopplade korsningar:

```
3
0 1 1
1 0 1
1 1 0
1
0 2 1 2
2 0 0 2
1 0 0 1
```

Och här är indatat som matchar exemplet givet ovan i problembeskrivningen:

```
4
0 1 1 1
1 0 0 0
1 0 0 0
1 0 0 0
1
0 0 0 0 1
2 0 0 0 2
3 0 0 0 3
1 0 0 0 1
```