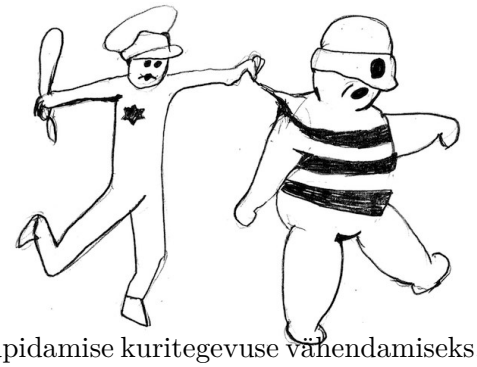
Politseinik ja röövel



Bytemore'i linnas on kuritegevus tõusnud kõigi aegade kõrgeimale tasemele. Muude kuritegude hulgas pannakse igapäevaselt toime röövimisi. Kui kuritegu on sooritatud, peab üksik patrullpolitseinik röövli kinni püüdma, joostes läbi kitsaste tänavate, mis ühendavad tänavanurki. Kahjuks pääsevad röövlid enamasti minema, sest nad tunnevad linna oluliselt paremini kui politseinikud.

Bytemore City Police Department (BCPD) korraldas nõupidamise kuritegevuse vähendamiseks. Üks vastuvõetud otsustest on kasutada röövlite tabamiseks arvutite abi. Selleks on BCPD loonud linna täpse kaardi, aga nüüd vajatakse tarkvara jälitamisstrateegiate leidmiseks.

Jälitamisstrateegia, kus üks politseinik jälitab üht röövlit, saab modelleerida järgmiselt:

1. Politseinik valib, millisel tänavanurgal ta patrullib.
2. Röövel valib, millisel tänavanurgal ta röövib (teades ette, kus politseinik on). Sellest hetkest alates saame eeldada, et nii politseinik kui ka röövel teavad alati, kus vastane asub.
3. Politseiniku käik võib olla kas liikumine naabernurgale (sellisele tänavanurgale, millele praeguselt nurgalt, läbides ühe tänava) või ootamine (jäädes paigale).
4. Röövel liigub oma käigul alati mõnele naabernurgale. Erinevalt politseinikest ei suuda röövlid paigal püsida, nende instinkt sunnib neid alati jooksma.
5. Politseinik ja röövel teevad kordamööda käike (politseinik alustab), kuni juhtub üks kahest tulemusest:
   (a) Seis kordab mõnda eelnevat (seis on defineeritud kombinatsioonina mõlema isiku asukohast ning sellest, kelle käigukord on). Kordus tähendab, et röövel suudab politseinikku lõputult vältida, nii et ta pääseb põgenema.
   (b) Politseinik ja röövel kohtuvad samal nurgal (pärast ükskõik kumma käiku). Sel juhul saab politseinik röövli kätte.

Task

Kirjutada programm, mis otsustab linnaplaani põhjal, kas röövli tabamine on võimalik, ja kui on, siis püüab röövli kinni, tehes politseiniku eest käike.

Programm peab eeldama, et röövel teeb optimaalseid käike.

Implementation

Realiseerida kaks funktsiooni:

- `start(N, A)`, mis saab järgmised parameetrid:

- N — tänavanurkade arv (nurgad on märgistatud arvudega 0 kuni $N-1$);
- A — kahemõõtmeline massiiv, mis kirjeldab tänavaid: iga $0 \le i, j \le N-1$ korral

$$A[i,j] \text{ on } \begin{cases} \texttt{false}, & \text{kui } i \text{ ja } j \text{ ei ole tänavaga ühendatud} \\ \texttt{true}, & \text{kui } i \text{ ja } j \text{ on tänavaga ühendatud} \end{cases}$$

Kõik tänavad on kahesuunalised (s.t iga i ja j korral $A[i,j] = A[j,i]$) ja ükski tänav ei ühenda tänavanurka iseendaga (s.t iga i korral $A[i,i]$ on `false`). Saab ka eeldada, et igalt tänavanurgalt on mööda tänavaid liikudes alati võimalik jõuda igale teisele nurgale.

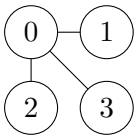
Kui röövlit on võimalik parameetritega kirjeldatud kaardil tabada, siis funktsioon `start` peab tagastama tänavanurga numbri, millel politseinik otsustab patrullida. Vastasel korral tagastada -1.

- Funktsioon `nextMove(R)` saab parameetrina tänavanurga numbri R, millel röövel parajasti asub, ja peab tagastama selle nurga numbri, millel politseinik pärast oma järgmist käiku asub.

Funktsiooni `start` kutsutakse välja täpselt üks kord, enne `nextMove` väljakutseid. Kui `start` tagastab -1, siis `nextMove` välja ei kutsuta. Vastasel korral kutsutakse `nextMove` välja kuni jälitamine lõpeb. Täpsemalt, programm lõpetab töö, kui juhtub üks järgmistest asjaoludest:

- `nextMove` tagastab ebakorrektse käigu;
- tekib korduv seis;
- röövel saadakse kätte.

Example



Vaatame parempoolsel joonisel toodud näidet. Antud juhul võib politseinik alustada suvaliselt nurgalt. Kui ta alustab nurgalt 0, võib ta oma esimesel käigul oodata ja röövel jookseb ise tema juurde. Teise võimalusena võib ta alustada suvaliselt teiselt nurgalt, oodata, kuni röövel jookseb nurka 0, ning siis ise ka sinna minna.

Näidissessioon näeks välja järgmine:

Funktsiooni väljakutse	**Tagastab**
`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

Märkus: Lühiduse mõttes tähistab `start` funktsiooni väljakutses `0` `false` ning `1` tähistab `true`.

Scoring

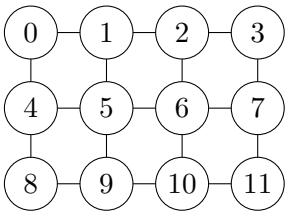
Maksimaalsete punktide saamiseks peab lahendus:

1. Õigesti leidma, kas politseinikul on võimalik röövel kätte saada;
2. Politseiniku eest käike tehes röövli edukalt kinni püüdma.

Lahendused, mis täidavad ainult esimese nõude, saavad alamülesannetes 3 ja 4 30% vastava alamülesande punktidest.

Alamülesanne 1 (16 punkti): $2 \leq N \leq 500$. Iga tänavanurkade paari vahel on täpselt üks võimalik tee.

Alamülesanne 2 (14 punkti): $2 \leq N \leq 500$. Tänavanurkade ja tänavate kaart moodustab ruudustiku. Ruudustikul on vähemalt kaks rida ja veergu ning tänavanurgad on märgistatud alltoodud joonisel näidatud viisil.



Alamülesanne 3 (30 punkti): $2 \leq N \leq 100$.

Alamülesanne 4 (40 punkti): $2 \leq N \leq 500$.

Constraints

Ajapiirang: 1 s.

Mälupiirang: 256 MB.

Experimentation

Näidishindaja teie arvutis loeb andmeid standardsisendist. Sisendi esimesel real on täisarv N — tänavanurkade arv. Järgmised N rida sisaldavad naabrusmaatriksi A ridu. Igal neist ridadest on N arvu väärtustega 0 või 1. Maatriks peab olema sümmeetriline ja selle peadiagonaali väärtused peavad olema nullid.

Järgmine rida sisaldab arvu 1, kui politseinik saab röövli kätte ja 0 vastasel juhul.

Lõpuks, kui politseinik saab röövli kätte, järgnevad N rida, mis kirjeldavad röövli strateegiat. Igaüks neist ridadest sisaldab $N+1$ täisarvu 0 ja $N-1$ vahel. Väärtus reas r ja veerus c, kus $c < N$, vastab seisule, kus röövli kord on käia, politseinik on nurgal r ja röövel on nurgal c. Väärtus ise tähistab tänavanurka, millele röövel liigub. Peadiagonaali väärtusi ignoreeritakse, kuna nad vastavad seisudele, kus röövel ja politseinik on juba samal nurgal. Rea r viimane arv tähistab röövli stardinurka, mis vastab politseiniku stardinurgale r.

Järgnevalt on toodud näidissisend näidishindajale, mis tähistab kolme tänavanurka, mis on omavahel ühendatud:

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

Järgnevalt on toodud näidissisend, mis vastab eelpool toodud näitele:

```
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
```