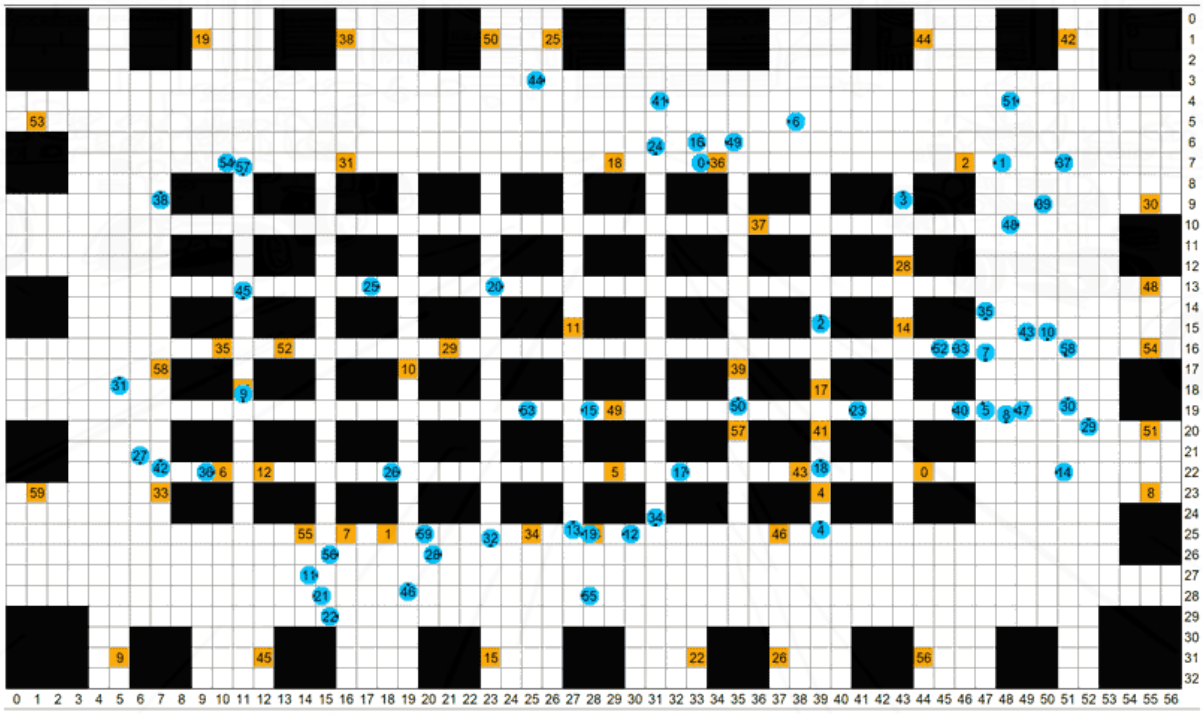
Szoftvertechnológia gyakorlat

Feladat

Ügyfelünk automatizált raktári rendszert üzemeltet, és szeretné optimalizálni a raktárban dolgozó robotok útvonalát. Ehhez egy szimulációs programot szeretne, ahol le lehet játszani a robotok mozgását, különböző útvonalkeresési megoldásokat lehet kipróbálni és elemezni. A kipróbálásnál valós időben le lehet futtatni egy futamot, a kielemzésnél pedig vissza lehet játszani a futás eredményét lassítva, gyorsítva, egy adott részre ugorva és ott akár előre hátra léptetve.

Egy raktár tipikus alaprajza:



A raktár négyzethálóba van szervezve. Vannak benne robotok, célállomások, és raktári állványok. A robotok feladata, hogy folyamatosan a számukra kijelölt célállomásokhoz menjenek el.

A robotok állapotát a pozíciójuk (a cella koordinátája) és az irányultságuk (észak, kelet, dél, nyugat) határozza meg. A robotok diszkrét időpontonként lépnek egyszerre, mindegyik úgy, ahogy a központi ütemező utasítja. A robot lépni úgy tud, hogy az irányultsága szerinti szomszéd cellába lép, a cellájában 90 fokot fordul óramutató járás irányába vagy azzal ellentétesen, illetve nem csinál semmit és helyben marad. A raktári állványok akadályok, ahova nem tudnak a robotok lépni.

A robotoknak el kell kerülniük a konfliktusokat: több robot nem léphet egyszerre ugyanabba a cellába, és két robot nem léphet egymással szemben (nem cserélhetnek helyet egymást átugorva). Az megengedett, hogy egy robot olyan cellába lép amelyikből egy másik éppen ellép, vagyis egymást szorosan követhetik, illetve derékszögű mozgásuknál a sarkuk nem akad össze.

A szimuláció diszkrét lépésekben történik. A központi vezérlő mondja meg minden lépésben, hogy melyik robot milyen műveletet végezzen. Minden lépésben minden robot végezhet egy műveletet.

A központi vezérlő a raktár állapotáról a következő információkat kapja meg:

- robotok száma
- raktár mérete (sor, oszlop)

- raktár neve
- raktár térképe
- aktuális szimulációs lépés száma
- minden robotra a kijelölt célállomás
- minden robot állapota

A központi vezérlő a szimuláció elején egy adott ideig előfeldolgozást végezhet, amihez az időkorlátot és a térkép adatait kapja meg. A szimuláció futása alatt mindig megkapja, hogy mennyi időn belül kell a robotok lépéseit megterveznie. Ha elkészült, akkor válaszként az összes robot lépését egyszerre elküldi. Ha mégsem tud adott időn belül válaszolni, akkor abban a szimulációs lépésben egyik robot sem mozdul, csak majd abban a szimulációs lépésben, amikor a központi ütemező válaszol.

A szimuláció indításához meg kell adni egy konfigurációs fájlban

- a raktár elrendezését (méret, állványok)
- a robotok számát
- a robotok kiindulási helyét
- a célállomások helyét időrendi sorrendben
- hány új célállomást lát előre az ütemező
- a célállomások kiosztási stratégiáját (ez egyelőre „roundrobin”, vagyis folyamatosan az éppen felszabaduló robotnak, több felszabaduló robot esetén abban a sorrendben, ahogy a kiindulási helyek fájljában szerepelnek)

A szimuláció indításakor meg lehet adni, hogy hány másodpercig tart egy szimulációs lépés, és hogy hány szimulációs lépést akarunk végrehajtani. A szimuláció során szeretnénk látni a raktár alaprajzát, a robotok mozgását irányultságukkal együtt, a robotok sorszámát, és a célállomások sorszámát. Nagy méretű raktár esetén tetszőleges részre rá tudunk nagyítani.

A szimuláció végén egy naplófájlban meg akarjuk kapni:

- robotmozgás szabályainak neve
- minden lépés ütközésmentes volt-e
- a robotok száma
- a robotok kiindulási helye
- összesen hány feladatot hajtottak végre
- összesen hány műveletet hajtottak végre
- hány lépésig tartott a szimuláció
- minden robotra, hogy ténylegesen milyen műveleteket hajtott végre (vesszővel elválasztva: “F” előre lépett, “R” órajárás szerint fordult, “C” órajárással szembe fordult, “W” várt)
- minden robotra, hogy az ütemező mit jelölt ki számára (ugyanolyan formátumban, mint az előző pontban, illetve “T” időtúllépés, azaz nem kapott műveletet)
- minden ütemező válaszra, hogy az ütemező mennyi idő után adta ki a következő műveletet (másodpercben, törtszám)
- hibák listája ([robot1, robot2, melyik lépésben, milyen hiba], ahol ha nincs robot, mert pl. falnak menne, akkor a robot azonosító -1)
- feladat események listája ([melyik feladat, melyik lépésben, esemény], ahol az esemény „assigned” vagy „finished”)
- feladatok listája ([melyik feladat, honnan, hova])

A naplófájlt a szimulációs programba betöltve újra le tudjuk futtatni a szimulációt, de itt már vizsgálódni is tudunk: tudjuk gyorsítva vagy lassítva lejátszani, meg tudjuk állítani, előre-hátra tudjuk léptetni, tetszőleges szimulációs lépésre rá tudunk állni. Látjuk a robotok mozgását irányultságukkal együtt, a robotok sorszámát, és az éppen aktuális célállomások sorszámát. Meg tudjuk állapítani, hogy melyik robot melyik célállomásra megy.

A fájlok formátumára minták alább vannak. A koordináták sokszor integer-ként vannak megadva, ahol a cellák számozása balról jobbra, fentről lefelé haladva mennek.

Konfigurációs fájl:

```
{
    "mapFile": "maps/random.map",
    "agentFile": "agents/random.agents",
    "teamSize": 20,
    "taskFile": "tasks/random.tasks",
    "numTasksReveal": 1,
    "taskAssignmentStrategy": "roundrobin"
}
```

Térkép, ahol a szabad cella „.”, illetve az akadály a „@” vagy bármilyen más karakter.

```
type octile
height 9
width 32
map
@@....@@@@....@@@@....@@@@....@@
................................
......@@@@....@@@@....@@@@......
......@@@@....@@@@....@@@@......
................................
......@@@@....@@@@....@@@@......
......@@@@....@@@@....@@@@......
................................
@@....@@@@....@@@@....@@@@....@@
```

Robot fájl (az első szám a robotok száma, utána az egyes robotok integer koordinátája):

```
5
134
489
602
983
484
```

Feladat fájl (az első szám a feladatok száma, utána az egyes célállomások integer koordinátája):

```
6
678
654
118
609
299
115
```

Naplófájl:


```
{
    "actionModel": "MAPF_T",
    "AllValid": "No",
    "teamSize": 2,
    "start": [
        [
            15,
            10,
            "E"
        ],
        [
            30,
            20,
            "W"
        ]
    ],
    "numTaskFinished": 1,
    "sumOfCost": 20,
    "makespan": 10,
    "actualPaths": [
        "F,C,F,F,R,F,F,W,W,F",
        "F,R,R,F,C,C,F,W,W,F"
    ],
    "plannerPaths": [
        "F,C,F,F,R,F,F,T,T,F",
        "F,R,R,F,C,C,F,T,T,F"
    ],
    "plannerTimes": [
        0.85070321,
        0.12345678,
        0.12345678,
        0.12345678,
        0.12345678,
        0.12345678,
        0.12345678,
        2.12345678
    ],
    "errors": [
        [
            -1,
            -1,
            8,
            "timeout"
        ],
        [
            -1,
            -1,
            9,
            "timeout"
        ]
    "events": [
        [
            [
                0,
                0,
                "assigned"
            ],
```

```
            [
                0,
                6,
                "finished"
            ],
            [
                2,
                6,
                "assigned"
            ]
        ],
        [
            [
                1,
                0,
                "assigned"
            ]
        ]
    ],
    "tasks": [
        [
            0,
            2,
            27
        ],
        [
            1,
            4,
            29
        ],
        [
            2,
            27,
            32
        ]
    ]
}
```

Minden számonkérés maximum 30 pontig lesz pontozva (4x30=120 pont). Mindegyik beszámolónál legalább 10 pontot el kell érni. Ezenkívül az alábbiakra további maximum 30 pontot lehet kapni. Alapvetően a végső jegy az összes pontszám huszonötöd részének egészrésze. Az alap verzió kell az elégségeshez.

Részfeladatok

Alap (elégségeshez kell): Meg lehet adni a szimuláció indításához szükséges adatokat, a robotok tudnak feladatokat végrehajtani, tudják kezelni a konfliktusokat legalább olyan szinten, hogy inkább várnak, de nem mennek egymásnak vagy akadálynak, naplófájl készül, és a naplófájl visszajátszható. Egy ütemező meg van valósítva, de kicserélhető másikra is.

Online megrendelések (10 pont): célpontokat a szimuláció futása közben interaktív módon is ki lehet jelölni.

Útvonal keresés (10 pont): Útvonalkereséshez valamilyen szélességi vagy mélységi keresésnél jobb algoritmust használ, pl. A*.

Útvonal keresés (10 pont): Útvonalkeresésnél a jövőbeli konfliktusok elkerülésére is figyel, pl. Cooperative A*.

Deadlock (10 pont): Detektálja és feloldja a deadlock szituációkat.