ElevatorWorld

1 Problema

O clădire are F etaje. Facem abstracție de noțiunea de parter: cel mai de jos etaj este etajul 1, iar cel mai de sus etaj este etajul F. În clădire există un grup de E lifturi. Toate lifturile au aceeași capacitate, de C persoane.

La fiecare etaj f există $F-1$ butoane de apel, numite Go_1, ..., Go_{e-1}, Go_{e+1}, ..., Go_F, care semnalează intenția de a merge la celalalte etaje. Lifturile nu au butoane.

Obiectivul proiectului este de a implementa o soluție pentru gestionarea lifturilor, care este distribuită într-un un sistem multi-agent.

Important: nu ne interesează să avem un algoritm cât mai bun pentru gestionarea lifturilor, cât o soluție în care eficiența să se bazeze pe comunicare și **negociere între agenți**.

2 Specificații

Un lift are următorii parametri dinamici (numere întregi):

- deschiderea, respectiv închiderea ușilor are o durată de T_{doors} secunde. Dacă închiderea ușilor este întreruptă, deschiderea lor durează tot T_{doors} secunde.
- în mod normal ușile încep să se închidă la T_{close} secunde de la deschiderea lor completă.
- tranzitul unei persoane care coboară din lift sau urcă în lift durează $T_{transit}$; pentru simplitate, acest timp înglobează ideea că mai multe persoane pot urca / coborî simultan. Astfel, dacă la un etaj trebuie să coboare 3 persoane și să urce 4, ușile vor trebui să rămână deschise $7 * T_{transit}$ secunde.
- liftul va începe să închidă ușile fie la T_{close} după ce s-au deschis complet, fie, dacă mai sunt persoane de urcat sau coborât, după ce toate persoanele care aveau de coborât sau urcat au făcut asta, conform cu $T_{transit}$.
- dacă un lift are de parcurs între două opriri 1 singur etaj, o va face în T_{slow}.
- dacă un lift are de parcurs între două opriri succesive traseul $f_i \to f_j$, cu $|i-j| \geq 2$ și $d = sig(j-i)$, vom avea următoarele etape:
  - între f_i și f_{i+d} liftul va accelera în timpul T_{accel}.
  - între f_{i+d} și f_{j-d} liftul va merge în timpul $T_{fast} * (|i-j|-2)$.

– între f_{j-d} și f_j liftul va decelera în timpul T_{accel}.

Astfel, un lift poate să aibă următoarele stări:

- *IN_TRANSIT* dacă este în mișcare, iar starea este caracterizată de direcția de deplasare (1 sau -1 pentru urcare respectiv coborâre) și și de etajul de care a trecut ultima oară.
- *DOORS_OPENING*, și starea este caracterizată de etajul la care se află.
- *DOORS_CLOSING*, și starea este caracterizată de etajul la care se află.
- *DOORS_OPEN*, și starea este caracterizată de etajul la care se află.
- *DOORS_CLOSED*, și starea este caracterizată de etajul la care se află.

Toate lifturile dintr-un scenariu au aceeași parametri dinamici.

Considerăm că persoanele care folosesc lifturile din clădire urmează același comportament:

- orice persoană care dorește să ia liftul are intenția de a se deplasa către un etaj diferit de cel care se află și va apăsa pe butonul corespunzător în momentul în care ajunge în zona lifturilor.
- orice persoană care ajunge în zona lifturilor va apăsa pe unul dintre butoane, indiferent că există un lift la etaj sau nu.
- persoanei i se va afișa numărul liftului pe care trebuie să îl ia.
  - dacă liftul are ușile complet deschise, liftul va mai sta la etaj cel puțin încă $T_{transit}$ (sau mai mult, dacă mai erau persoane care aveau de coborât/urcat din/în lift).
  - dacă liftul are ușile în curs de închidere, va deschide ușile, apoi persoanele care au de urcat vor urca.

Dacă unei persoanei-a fost atribuită un lift, aceasta trebuie să aibă loc în el, și liftul trebuie să o ducă până la etajul dorit.

3 Agentificare

Vom modela problema ca un sistem multi agent format din $E + F$ agenți, având E agenți care controlează lifturile și F agenți care gestionează etajele. Pentru simplitate, fiecare agent lift își va gestiona singur starea (poziția liftului, trecerea timpilor, persoanele din lift etc); iar fiecare agent etaj își va gestiona persoanele din zona lifturilor de pe etajul respectiv. Fiecare persoană care dorește să ia liftul va fi identificată printr-o *comandă*, caracterizată prin etajul de origine și etajul destinație. O comandă poate avea una dintre următoarele stări:

- *REGISTERING* – persoana a apăsat pe buton dar nu i s-a asociat un lift.
- *WAITING* – persoana așteaptă pe etaj, pentru că liftul nu a ajuns, liftul nu are încă ușile deschise, sau încă nu a apucat să se urce în lift.
- *IN_ELEVATOR* – persoana este într-un lift în mișcare, sau nu a apucat încă să coboare din lift.
- *COMPLETED* – persoana a coborât din lift. Comenzile în această stare rămân în gestiunea etajului destinație.

Pentru simplitate, atunci când la un etaj dintr-un lift trebuie să coboare c persoane și în lift trebuie să se urce u persoane, avem următoarea secvență:

- se deschid ușile, într-un timp T_{doors};
- din momentul în care s-au deschis ușile, măsurăm un timp $c * T_{transit}$ și apoi toate cele c comenzi care aveau de coborât devin *COMPLETED*.
- apoi, măsurăm un timp $u * T_{transit}$ și apoi toate cele u comenzi care aveau de urcat ajung în starea *IN_ELEVATOR*.
- ușile încep să se închidă.

Pe lângă agenți, vom avea o structură externă agenților, numită *Simulator*, care va avea acces la toți agenții și care:

- înregistrează la agenții etaje comenzile în momentul în care persoanele respective apasă pe buton (intră în zona lifturilor de pe un etaj).
- interoghează agenții etaj despre comenzile care se află pe etaj, în stările *REGISTERING* și *WAITING*, despre comenzile terminate, sau despre statutul cererilor emise spre lifturi.
- interoghează agenții lift despre parametrii stării lor și despre comenzile aflate în starea *IN_ELEVATOR*.
- atunci când lifturile se află la etaje, coordonează coborârea și urcarea persoanelor în lift și schimbă starea comenzilor.

Avem următorul comportament *de bază*:

- lifturile sunt inițial goale, cu ușile închise, la etajul 1.
- când Simulatorul înregistrează o comandă la un agent etaj, acesta trimite un mesaj (preferabil un CFP) tuturor lifturilor pentru a servi comanda respectivă.
- când un agent lift care este gol și stă pe loc (nu are comenzi atribuite) primește un CFP, îl acceptă și începe tratarea comenzii.
- când un agent lift care are comenzi atribuite primește un CFP, verifică dacă direcția de deplasare comenzii (ascendentă sau descendentă) coincide cu direcția curentă de deplasare a liftului, și dacă este prevăzut să aibă loc în spațiul respectiv (conform cu celelalte comenzi acceptate), și răspunde afirmativ dacă poate lua comanda.
- agentul etaj acceptă prima propunere care vine de la un agent lift care este disponibil și anunță persoana ce lift trebuie să ia.
- dacă ușile au început să se închidă, nu se redeschid pentru nimeni.
- un lift în mișcare va prelua accepta comenzi doar dacă se află la cel puțin 5 etaje de etajul de origine al comenzii.
- un lift care nu mai are comenzi închide ușile și rămâne la etajul unde este.

Agenții nu au acces direct unii la starea altora, dar pot afla starea altor agenți folosind mesaje de interogare (sau pot infera care ar trebui să fie starea altui agent).

Simulatorul este singurul care acces direct (prin referință la obiecte) la ceilalți agenți, pentru a putea citi starea și a înregistra comenzi /

Simulatorul va afișa starea întregului sistem la fiecare secundă de timp simulat și va afișa statistici ale timpului total pentru comenzi (timpul minim, timpul pentru percentila 10, timpul median, timpul pentru percentila 90, timpul maxim). Timpul total pentru o

comandă este timpul în secunde de când comanda intră în starea *REGISTERING* până când intră în starea *COMPLETED*.

4 Cerințe și lucru pe echipe

O echipă are 2 membri, membrul **A** și membrul **B**. Trebuie implementate următoarele elemente de către următorii membri:

- **A**: Citirea din fișier a scenariului pentru simulator (2p).
- **B**: Afișarea stării sistemului, folosind o interfață grafică sau text-graphics, care conține o reprezentare a etajelor, a lifturilor, a comenzilor din lifturi și a comenzilor care așteaptă pe etaj (ca *REGISTERING* sau *WAITING*) (2p).
- **A**: Simulatorul de evenimente (2p).
- **A**: Comportamentul de bază pentru agenții etaje (3p).
- **B**: Comportamentul de bază și o dinamică corectă pentru agenții lifturi (5p).
- **A**: Comportamentul inteligent 2 pentru agenții etaje (3p sau mai mult)
- **B**: Comportamentul inteligent 1 pentru agenții lifturi (3p sau mai mult)

Un comportament inteligent pentru un agent înseamnă un comportament care îmbunătățește timpul total median pentru comenzile din toate scenariile.

Pentru cerința de comportament inteligent, fiecare membru al echipei va redacta un document de 1-2 pagini care descrie comportamentul implementat.

Bonusul pentru un comportament inteligent bine proiectat poate merge până la 2 puncte.

4.1 Etape

Vom avea următoarele etape:

1. cu elementele:
   - citirea din fișier
   - afișarea stării sistemului
   - simulatorul de evenimente și înregistrarea comenzilor la agenții etaje
   - un comportament foarte simplu pentru lifturi în care lifturile aleg o destinație aleatoare, merg acolo, deschid ușile, după T_{close} închid ușile și pornesc spre o nouă destinație aleatoare.
2. conținând comportamentul de bază pentru agenții etaje și pentru agenții lifturi.
3. conținând comportamentul inteligent pentru agenții etaje și pentru agenții lifturi.

5 Detalii de implementare

Puteți implementa proiectul fie folosind Jade, fie folosind implementarea cu `step` și `response` din primele laboratoare (și cu `messageBox` pentru mesage). În Jade, Simulatorul poate fi un agent. În abordarea cu `step` și `response`, un `step` poate dura mult mai puțin decât 1 secundă de timp simulat (e.g. 1/10).

Puteți face implementarea în Java sau în Python.

Pentru citirea fișierului de intrare puteți utiliza o implementare externă.

Primul argument al programului în linia de comandă va fi fișierul care conține scenariile, iar al doilea argument va fi numele scenariului de rulat.

Implementați un factor de scalare pentru timp, astfel încât timpul de 1 secundă al simulării să poată dura mai mult sau mai puțin. Acest factor va fi al treilea argument al programului în linia de comandă.

6 Intrare

Fișierul de intrare va fi un fișier JSON, în formatul vizibil în directorul `tests`. Cheia `scenarios` conține un dicționar, în care fiecărui nume de scenariu îi corespunde o listă de *generatoare*.

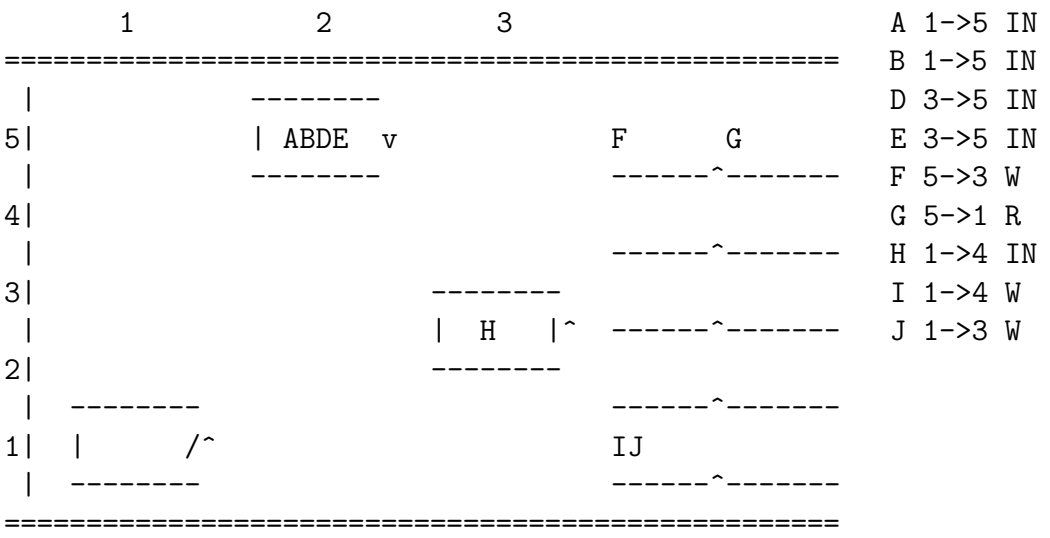
Pentru fiecare *generator* se va genera, între momentele `Tfrom` și `Tto` (inclusiv), la fiecare `Period` secunde, o comandă pentru o persoană care vrea să meargă de la un etaj ales aleator între `Ofrom` și `Oto` (inclusiv), la un etaj ales aleator între `Dfrom` și `Dto` (inclusiv), diferit de etajul de la care a plecat.

Notă importantă: numai Simulatorul cunoaște aceste informații de distribuție a comenzilor; agenții nu au acces la ele. Agenții pot, însă, implementa un comportament adaptiv (e.g. dacă au fost de 10 ori mai multe cereri de la un etaj, este mai probabil ca următoarea cerere să fie tot de la acel etaj).

7 Exemplu de format de ieșire

Pentru 3 lifturi, 5 etaje, capacitate maximă de 4 persoane și cereri notate cu literele A-Z:

Aceasta este doar o sugestie.



În această reprezentare am inclus următoarele informații:

- Liftul 1 este la etajul 1, urmează să meargă în sus, ușa este în curs de deschidere, și comenzile `I` și `J` așteaptă să se urce în[tr-un] lift.
- Liftul 2 este la etajul 5, urmează să coboare, ușa este deschisă, în interior sunt comenzile `ABDE`, iar la etaj așteaptă comanda `F` să urce, iar `G` să i se atribuie un lift.
- liftul 3 este în urcare între etajele 2 și 3, iar în interior se află comanda `H`.
- În dreapta sunt prezentate originea și destinația fiecărei comenzi, și o indicație a stării comenzii.

8 Changelog

- 28.04:
  - mențiune că etajul destinație al unei comenzi trebuie să fie diferit de etajul de la care pleacă (sec. 6).
  - precizare că pot exista mai mulți pași `step` într-o secundă de timp simulat (sec. 5) .
  - precizare a rolului Simulatorului în gestionarea comenzilor (sec. 3).