ALGORITMI PARALELI SI DISTRIBUITI
Tema #2 Traffic simulator

Termen de predare: 16.12.2020

Cerinta

Pornind de la notiunile invatate in cadrul laboratoarelor, in aceasta tema veti avea de implementat modele de sincronizare ale urmatoarelor situatii din trafic:

- **simple_semaphore** - Masini din 2 sensuri diferite asteapta la semafor un numar de T secunde
- Masini care asteapta trecerea altor masini la intrarea de pe un drum fara prioritate; masinile fara prioritate pot intra doar cate una la un moment dat; masinilor cu prioritate le ia 2s sa paraseasca intersectia!
- **simple_roundabout -** Sens giratoriu in care pot intra maximum X masini odata si dureaza T secunde sa-l paraseasca
- **simple_strict_1_car_roundabout** - Sens giratoriu in care se asteapta sa intre exact cate o masina de la fiecare intrare odata si dureaza T secunde sa-l paraseasca
- Sens giratoriu in care se asteapta sa intre exact cate X masini de la fiecare intrare odata si dureaza T secunde sa-l paraseasca
- Sens giratoriu in care se asteapta sa intre cel mult cate X masini de la fiecare intrare odata si dureaza T secunde sa-l paraseasca; obligatoriu pastrati wait-ul inainte de a intra in sensul giratoriu
- Trecere de pietoni cu activare la un numar minim de oameni in asteptare; puteti considera re-entranta aceasta trecere de pietoni (masinile merg in cerc)
- Drum in lucru, trec doua sensuri pe o singura banda; trec exact cate X masini din fiecare sens, alternativ; prima data trec masinile de pe sensul 0
- Drum in lucru, exista un singur sens de mers, N benzi dintre care M sunt libere. Redirectati traficul astfel incat sa se mapeze cateva benzi initiale la cate o banda libera. Dintre cele L benzi initiale asignate, se alege pe rand cate una din care trebuie sa treaca maxim cate X masini odata. Benzile de pe care pornesc masini trebuie sa se roteasca intre ele. Masinile trebuie sa porneasca in ordinea in care au ajuns pe benzile initiale. Exemplu: Initial drumul are A1 … A10 benzi. La un moment dat, unele intra in renovare. Putem presupune ca vor ramane libere doar benzile A1, A2 si A3. Putem realiza cuplajul de benzi A1 <-> {A1, A2, A3}, A2 <-> {A4, A5, A6}, A3 <-> {A7, A8, A9, A10}. Pe A1 vor circula prima data X masini

care veneau de pe A1, apoi X masini de pe A2, urmate de X masini de pe A3. Pe masura ce ramanem fara masini pe unele dintre benzi, le scoatem din coada. Daca A3 ramane fara masini, atunci corelatia se va modifica in A1 <-> {A1, A2}, A2 <-> {A4, A5, A6}, A3 <-> {A7, A8, A9, A10}.
- Trecere la cale ferata, masinile din ambele sensuri trebuie sa astepte si sa porneasca in ordinea in care au ajuns in coloane; asteptarea dureaza T secunde

Bonus:

- Intersectie nemarcata in care intra 4 masini din si cu directii diferite fixate; se doreste trecerea lor in ordinea corecta conform codului rutier, fara a exista accidente, fara deadlock-uri
- Intrecere a N masini fiecare pe banda ei; un obstacol poate aparea random blocand masina de pe linia sa timp de T secunde (hint: use Listeners). Considerand un numar maxim de obstacole ce pot aparea, printati clasamentul la fiecare schimbare a acestuia (inceput + dupa fiecare obstacol). Hint: use Listeners

Observatii

- Masinile vor avea intotdeauna un id si un sens de mers (start - end). La majoritatea task-urilor veti primi doar start-ul. Numai la bonus e nevoie si de end.
- Vi se vor oferi in teste informatii exacte de initializare a sensurilor giratorii (timp T de parcurge, respectiv capacitate N unde este cazul)
- Vi se va specifica in teste numarul variabil / maxim de masini ce pot intra in intersectie unde este cazul
- Citirea se va face din fisiere
- Un test verifica o **singura** situatie enumerata anterior (un singur task).
- Pe langa testul in sine veti primi si
    - un test_sablon care va specifica semnificatia fiecarui element din fisierul de test din tipul respectiv
    - un output_sablon care va specifica ordinea, momentul si mesajul afisarii
- Veti primi un script de testare automata. Testarea manuala se poate realiza prin parcurgerea test cu test si redirectarea output-ului intr-un fisier.
- Afisarea se va face la Standard Output.
- Puteti modifica dupa cum doriti structura codului / puteti sa nu folositi scheletul cat timp respectati, unde este cazul, timpii de asteptare, respectiv modul de generare a pietonilor.
- Se garanteaza ca la testare se va tine cont de variabilitatea id-urilor.

Sugestii

- Incercati sa va scrieti codul cat mai lizibil, modular si usor de extins
- Utilizati variabile si clase cu nume sugestive
- Lasati cat mai multe comentarii explicative in cod
- Respectati un coding style
- Utilizati cat mai multe design pattern-uri care considerati ca v-ar usura viata
- Exemple de design pattern-uri: Singleton, Factory, Prototype, Listener, Strategy etc.
- Puteti utiliza atat structuri de date sincronizate, cat si bariere, semafoare, syncronized, wait - notify - notifyAll, sleep

Notare

Tema se va trimite si testa automat la aceasta adresa: https://apd-checker.dfilip.xyz. Se va trimite sub forma unei arhive Zip care, pe langa fisierele sursa, va trebui sa contina urmatoarele doua fisiere ın radacina arhivei:

- Makefile - cu directiva build care compileaza tema voastra si genereaza un executabil numit tema2 par aflat in radacina arhivei
- README - fisier text in care sa se descrie pe scurt implementarea temei, precum si alternative la implementare dupa cum considerati.

Punctajul este divizat dupa cum urmeaza:

- 50p - trecerea testelor
- 30p - corectitudinea implementarii
- 20p - claritatea codului si a explicatiilor din README
- 20p - Bonus

Prin corectitudinea implementarii se intelege regasirea elementelor de sincronizare potrivite fiecarui caz din cele mentionate anterior. **NU** se vor considera corecte solutiile care **forteaza** sincronizarea prin **sleep-uri** si **yield-uri**. Sleep poate fi folosit numai in situatiile mentionate in enunt, in care este impus un timp de asteptare.

Model de test_sablon

Nume_situatie
Numar_masini
Directie_masina_1 (Prioritate_masina_1)

…
Timp_sens_giratoriu (Capacitate_sens_giratoriu)