SISTEME TOLERANTE LA DEFECTE
Tema #1 Langton's Ant

Termen de predare: 02-Apr-2023 23:55

Obiective

Scopul acestei teme este de a implementa, în C, folosind biblioteca MPI, un program distribuit, scalabil cu numărul de procese, având ca scop simularea unei variante modificate a jocului Langton's Ant - inventat de Chris Langton.

Date introductive

Jocul presupune existența unei hărți sub forma unei matrici de dimensiune H x W, fiecare element având una din următoarele două culori:

- **Alb**
- **Negru**

Simularea va fi executată pentru un număr dat de etape (pași). Astfel, pe unul dintre elementele matricei se indentifică o furnică ce se poate deplasa în una dintre cele patru direcții cardinale, la fiecare pas al simulării, după următoarele reguli:

- Dacă se află pe un element **Alb**, furnica își modifică direcția cu 90° dreapta (sens orar), schimbă culoarea pătratului și avansează pe următorul pătrat;
- Dacă se află pe un element **Negru**, furnica își modifică direcția cu 90° stânga (sens trigonometric), schimbă culoarea pătratului și avansează pe următorul pătrat.

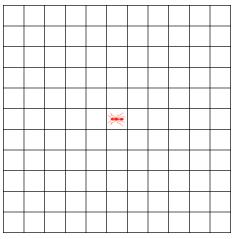
Totuși, pentru ca această temă să îndeplinească unul dintre principalele scopuri ale materiei, adică cel de a implementa programe distribuite și scalabile, vom considera mai multe furnici pe hartă. Astfel, pe fiecare element al hărții va fi instanțiată **cel mult** o furnică. Acest lucru implică faptul că pot exista, la un moment dat, mai multe furnici pe același element.

Cu toate acestea, pentru a nu crește prea mult nivelul de complexitate al implementării, în cazul în care se află mai multe furnici pe același element, culoarea elementului va fi schimbată o singură dată.

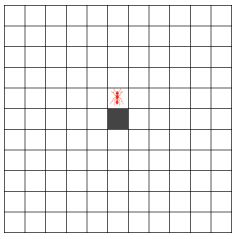
Deci, date fiind regulile de deplasare și faptul că un pătrat are patru laturi, acest lucru implică faptul că după fiecare pas al simulării vor putea exista, concomitent, **cel mult patru furnici** pe același element.

În plus, harta are o dimensiune fixă (limitată), astfel furnica poate să părăsească zona, fapt ce o va scoate din joc.

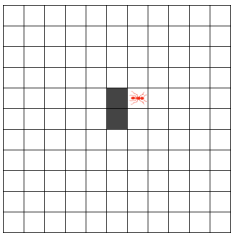
Exemplu simulare pe o hartă de dimensiune 11 x 11



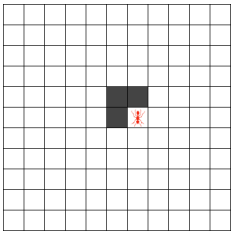
(a) Harta inițială



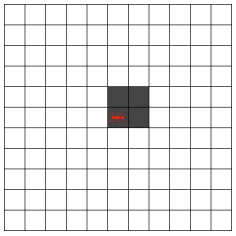
(b) După pasul 1



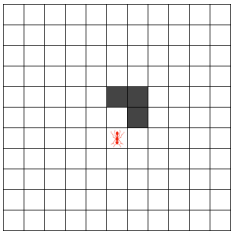
(c) După pasul 2



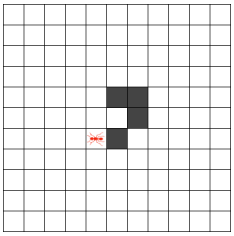
(d) După pasul 3



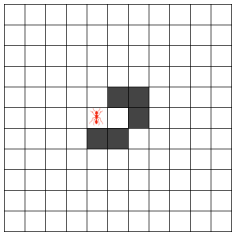
(e) După pasul 4



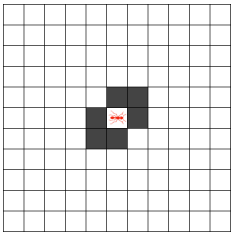
(f) După pasul 5



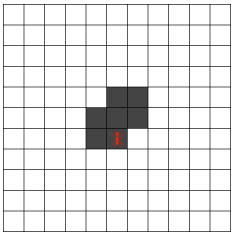
(g) După pasul 6



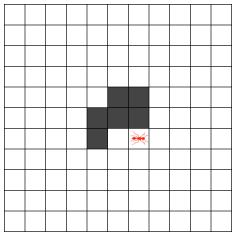
(h) După pasul 7



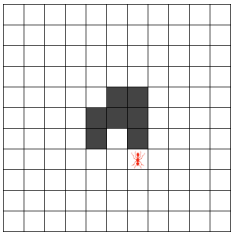
(i) După pasul 8



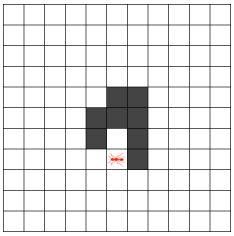
(j) După pasul 9



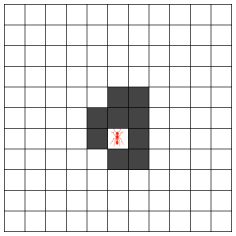
(k) După pasul 10



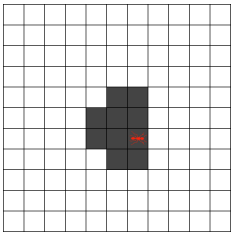
(l) După pasul 11



(m) După pasul 12



(n) După pasul 13



(o) După pasul 14

Procesul de simulare

Simularea va fi constituită dintr-un număr de etape, starea celulelor schimbându-se 'simultan' în funcție de celulele din etapa anterioară.

Programul va primi ca input un fișier și un număr de pași ce va reprezenta numărul de etape ale simulării, iar rezultatul final va fi scris într-un fișier de ieșire.

Pe prima linie vom avea:

- Un număr întreg **H** ce indică înălțimea hărții;
- Un număr întreg **W** ce indică lungimea hărții;
- Un număr întreg **S** ce indică numărul total de pași ce vor fi efectuați în cadrul simulării. Acesta există doar în cazul fișierului de intrare;

Pe urmă vor urma **H** linii cu **W** coloane, fiecare celulă având valoarea reprezentată de concatenarea dintre elementele **CULOARE** și, opțional, **FURNICĂ**, unde:

- **CULOARE** are valoarea **1** (ALB) sau **0** (NEGRU);
- **FURNICĂ** are valoarea **0** (furnică având direcția **STÂNGA**), **1** (furnică având direcția **SUS**), **2** (furnică având direcția **DREAPTA**), **3** (furnică având direcția **JOS**), sau poate să nu existe.

Exemplu fișier de intrare (1)

```
5 5 1
1 1 1 1 1
1 1 12 1 1
1 11 10 13 1
1 1 10 1 1
1 1 1 1 1
```

Exemplu fișier de ieșire (1)

```
5 5
1 1 1 1 1
1 1 01 1 1
1 0 02031 0 1
1 1 0 1 1
1 1 1 1 1
```

Exemplu fișier de intrare (2)

```
5 5 17
1 1 1 1 1
1 1 12 1 1
1 11 10 13 1
1 1 10 1 1
1 1 1 1 1
```

Exemplu fișier de ieșire (2)

```
5 5
1 0 0 0 0
1 0 0 0 0
0 0 0 0 1
1 0 1 0 0
0 1 00 0 0
```

Rularea programului

Rularea programului se va realiza astfel:

```
mpirun --oversubscribe -np NUM_PROCS ./homework IN_FILENAME OUT_FILENAME
```

Unde:

- **NUM_PROCS** - numărul de procese;
- **IN_FILENAME** - numele (path-ul) fișierului de intrare;
- **OUT_FILENAME** - numele (path-ul) fișierului de ieșire;

Exemplu:

```
mpirun --oversubscribe -np 2 ./homework in.txt out.txt
```

Trimitere și punctare

- Este necesară crearea unei arhive **.zip** care trebuie să conțină fișierul "homework.c" și uploadarea acesteia în cadrul checker-ului.
- O temă care nu compilează va primi 0 puncte. Tema va fi rulată și testată pentru corectitudine și scalabilitate pe un checker al ATM.
- Link-ul către acest checker va fi transmis ulterior.

Orice încercare de a abuza checker-ul va duce la un punctaj de 0 pe toate temele.

Distribuția punctajului este următoarea:

- 60 puncte - Output corect program distribuit și scalabil pentru hărți de dimensiuni divizibile cu 2, 4 și 8 procese.
- 40 puncte - Output corect program distribuit și scalabil pentru hărți de dimensiuni nedivizibile cu 2, 4 și 8 procese.

Arhiva .zip trebuie să conțină fișierul "homework.c". O temă care nu compilează va primi 0 puncte. Tema va fi rulată și testată pentru corectitudine și scalabilitate pe un checker al ATM. Link-ul către acest checker va fi transmis ulterior.

Orice încercare de a abuza checker-ul va duce la un punctaj de 0 pe toate temele.