Playing Cards

- documentatie –

NOTIUNI INTRODUCTIVE DESPRE JOCURILE DE CARTI

Cartile de joc cu punctaj mai mic sunt numerotate de la 2 la 10, la care se adauga si cartile mai puternice precum Juve(valoarea 12), Dama(valoarea 13), Popa(valoarea 14), As(poate reprezenta valoarea 1, 11 sau 15) formand un total de 54 de carti in pachet. Acestea se impart in carti de inima rosie, inima neagra(sau frunza), romb si trefla, plus doi jockeri, unul negru si unul rosu.

DESCRIEREA JOCULUI

Jocul implementat este inspirat din mai multe jocuri de carti renumite, precum Bridge, Spades, Whist si Rentz. Acesta este jucat intr-un numar de exact patru jucatori. Pe parcursul jocului, se vor folosi carti puternice (“atu-uri”).

La jocul Spades, atu-ul este chiar cartea de inima neagra, indiferent de valoarea sa.

Ideea de baza a jocului este de a realiza un punctaj cat mai apropiat de estimarea initiala. La inceput jucatorii primesc fiecare cate treisprezece carti. Acestea sunt amestecate si impartite cate una fiecarui jucator.

Apoi fiecare participant trebuie sa aproximeze, in functie de cartile pe care le-a primit, numarul de runde pe care le va avea castigate la final.

Jucatorul care deschide partida este cel care a pregatit cartile, iar fiecare mana noua incepe cu cartea data de castigatorul mainii anterioare.

In functie de cartea oferita de primul jucator al rundei, ceilalalti participanti sunt obligati sa puna in teanc o carte de acelasi semn ca cea existenta, nu au voie sa ascunda o carte potrivita mainii jucate. Daca nu au tipul necesar, pot folosi atu-ul, cartea suprema, ce va desemna jucatorul ce o ofera drept castigatorul rundei. In lipsa cartii de tip se poate da orice alta carte.

Daca un jucator, realizeaza mai multe maini fata de numarul estimat initial, acestuia I se vor scadea din puncte. De asemenea, daca acesta castiga doar o parte din mainile propuse la inceput, va primi puncte insa nu la fel de multe.

La finalul jocului, concurentul cu cea mai buna aproximare, mai exact, castigarea a cat mai multe runde fata de numarul estimat, este declarat castigatorul partidei de Spades.

CONSTANTE

Agoritmul trebuie sa stie cine cui trimite informatia si ce reprezinta aceasta informatie, iar acest lucru este realizat prin atasarea de id-uri unice fiecarui jucator si fiecarui tip de tranzactie . Constantele ce se definesc in fisierul constants.h si au rolul de usura utilizarea de valori fixe pe parcursul programului.

/\* id-ul jucatorului ce tine pariurile \*/

#define DEALER 0

/\* numarul de procese \*/

#define PLAYER\_NR 4

/\* numarul de carti din setul initial \*/

#define DECK\_SIZE 52

/\* numarul de carti necesar unui jucator \*/

#define PLAYER\_NR\_CARDS 13

/\* id-ul cartilor de inima neagra \*/

#define SPADES 0

/\* id-ul cartilor de inima rosie \*/

#define HEARTS 1

/\* id-ul cartilor de romb \*/

#define DIAMONDS 2

/\* id-ul cartilor de trefla \*/

#define CLUBS 3

/\* id-ul cartii ce joaca rolul atu-ului \*/

#define TRUMP 0

/\* identificatorul tranzactie de impartire a cartilor \*/

#define DISTRIBUTE\_CARD\_TAG 10

/\* anunta procesul ca el incepe runda \*/

#define START\_HAND\_TAG 11

/\* anunta procesul ca el urmeaza in runda curenta \*/

#define PLAY\_HAND\_TAG 1

METODE AJUTATOARE

Intregul joc reprezinta un schimb de informatie intre participanti. Acestia trebuie sa stie mereu, ce carti i-au fost distribuite,cine este jucatorul care incepe runda, ce carti s-au dat intr-o runda.

Pentru ca jucatorii sa primeasca imediat informatiile necesare, acestea sunt afisate in consola fiecaruia folosind functia **print\_card** care afiseaza simbolul si valoarea unei carti.

void print\_card(int card) {

int card\_id = card / 4 + 1;

int card\_type = card % 4;

string card\_symbol = to\_string(card\_id);

if (card\_id == 11) card\_symbol = 'J';

if (card\_type == SPADES) cout << "\u2660" << card\_symbol;}

De asemenea, algoritmul foloseste functia **print\_deck** ce afiseaza toate cartile unui jucator pentru o runda.

Metoda **print\_hand** este cea care afiseaza mereu cartile jucate pe parcursul unei runde.

AMESTECAREA CARTILOR

La inceputul jocului, algoritmul foloseste functia **random\_shuffle** care amesteca intre ele cele 52 de carti.

DISTRIBUIREA CARTILOR LA JUCATORI

Metoda **distribute\_cards** este cea care imparte cartile jucatorilor.

Jucatorii, in numar de patru, au atasat un cod unic de identificare de la 0 la 3. In cazul acestei implementari, cel care se ocupa de preluarea pariurilor si declararea castigatorului este jucatorul cu id-ul 0.

Cei patru jucatori reprezinta cele patru procese pe care algoritmul trebuie sa le incheie, in codul de mai jos, **numtasks**.

Pentru fiecare proces in parte, se distribuie carti si se insereaza in vectorul **own\_cards.**Acesta salveaza pe rand cate o carte pentru fiecare process in parte din vectorul initial si amestecat de carti **deck.**Daca acestea trebuie trimise catre alt jucator, se apeleaza MPI\_Send cu urmatoarele argumente in aceasta ordine: adresa cartii ce se distribuie, numarul de elemente trimise, tipul elementului trimis, id-ul jucatorului catre care se trimite cartea, id-ul tranzactiei ce indica faptul ca se trimite o noua carte din set si grupul de procese din care cel curent face parte.

void distribute\_cards(int numtasks, int \*deck, int \*own\_cards) {

int i=0, k=0;

while (i < DECK\_SIZE) {

for (size\_t j = 0; j < numtasks; j++) {

if (j == 0) {

own\_cards[k++] = deck[i];

} else {

MPI\_Send(&deck[i], 1, MPI\_INT, j, DISTRIBUTE\_CARD\_TAG, MPI\_COMM\_WORLD);

}

i++;

}

}

}

ESTIMAREA INITIALA SI CASTIGATORUL UNEI RUNDE

Dupa ce se primesc toate cartile, algoritmul calculeaza numarul de maini pe care jucatorul poate sa le castige . Acest lucru este realizat de metoda **calculate\_possible\_tricks(int \*deck).**

Functia parcurge tot setul de 13 carti primit de jucator si determina tipul acesteia si valoarea pe care o poarta.

Cum tipurile sunt de patru feluri, este suficient sa afle restul impartirii pozitiei cartii in setul initial de carti la patru, iar pentru a afla valoarea atasata se face catul impartirii pozitiei la patru si retine intr-un vector de pozitie cartea citita si incrementeaza contorul afferent tipul de carte, pentru a stii cate carti din fiecare tip mai are.

int card\_id = deck[i] / 4;

int card\_type = deck[i] % 4;

Pasul urmator realizat de algoritm este de a estima numarul de maini castigatoare in functie de valorile cele mai mari primite si tipul cartilor respective. Deci, posibilele maine castigatoare se calculeaza pentru fiecare tip in parte cu **functia tricks\_per\_suit(int \*suit, int size** ).

Desemnarea castigatorului unei runde este facuta cu ajutorul metodei **calculate\_winner** ce cheama la randul ei functia **card\_is\_better(int card, int card\_compare)** unde cartile sunt comparate ca tip si valoare doua cate doua, prin folosirea acesteia de mai multe ori gasind cartea cea mai buna de pe masa.

In fiecare runda, un jucator trebuie sa ofere cartea sa. Metoda **pick\_card** hotaraste daca jucatorul este human\_player chemand metoda **pick\_best\_card\_human(int \*own\_cards, int \*hand, int rank)**, iar in caz contrar este ghidat de catre algoritm (computer-player) prin metoda

**pick\_best\_card\_ia(int \*own\_cards, int \*hand, int rank),**

avand rolul de a lua aceasta decizie. Human player nu beneficiaza de un algoritm care sa aleaga cel mai bun scenariu, ci i se prezinta toate cartile ramase pentru a selecta una cu care vrea sa participle la runda.

DESEMNAREA CASTIGATORULUI JOCULUI

Castigatorul este desemnat pe baza punctajului acumulat. La sfarsitul fiecarei runde, acesta poate in cazul ideal, sa realizeze numarul estimat de maini castigatoare, primind cate 10 puncte pentru fiecare.

Daca a facut mai multe, primeste maini estimate si castigate \* 10 + maini castigate in plus, puncte. Daca a facut mai putine, I se scad cate 10 puncte pentru fiecare mana castigata. In plus , daca diferenta intre estimare si realitate este mai mare de 5 maini, jucatorul este penalizat cu (100 – maini castigate) puncte.

DERULAREA UNEI RUNDE

Cu ajutorul metodelor explicate mai sus, se poate incepe jocul propriu-zis, unde va exista si un castigator pentru runda curenta. Dupa plasarea pariurilor, jucatorul cu id-ul zero, pentru prima runda din joc, va oferi prima carte de pe masa. Carte jucata de acesta, trebuia transmisa catre toti ceilalti participanti :

if (i == 0 && rank == 0) {

pick\_card(own\_cards, hand, rank);

MPI\_Send(hand, PLAYER\_NR, MPI\_INT, next, PLAY\_HAND\_TAG, MPI\_COMM\_WORLD);

i++;}

Se folosesc **MPI\_Status si MPI\_Probe(MPI\_ANY\_SOURCE, MPI\_ANY\_TAG, MPI\_COMM\_WORLD, &status)** pentru a primi informatii despre tranzactia realizata, cum ar fi expeditorul mesajului, dimensiunea. De asemenea, **MPI\_Probe** receptioneaza mesajul si verifica tagul tranzactiei .Daca acesta este START\_HAND inseamna ca incepe o noua runda, altfel se joaca mai departe.

Jucatorii care receptioneaza informatia folosesc **MPI\_Recv(hand, PLAYER\_NR, MPI\_INT, MPI\_ANY\_SOURCE, START\_HAND\_TAG, MPI\_COMM\_WORLD, &status).**

DERULAREA INTREGULUI JOC

Functia **int main(int argc, char \*argv[])**ruleaza pentru fiecare proces in parte. Iar de aici sunt chemate toate functiile ajutatoare ce vor determina pe rand, cartile impartite si pariurile jucatorilor, castigatorii rundelor si punctajul acumulat de fiecare.

Este necesara initializarea pentru MPI si aplicarea conditiilor pentru numarul de jucatori astfel incat sa se asigure corectitudinea jocului.

/\* initializeaza mediul MPI \*/

MPI\_Init(&argc, &argv);

/\* verifica executarea cu succes \*/

if (rc != MPI\_SUCCESS) {

cout << "Error starting MPI program. Terminating.\n";

/\* inchide mediul MPI \*/

MPI\_Abort(MPI\_COMM\_WORLD, rc);}

/\* determina dimensiunea grupului comunicatorului \*/

MPI\_Comm\_size(MPI\_COMM\_WORLD, &numtasks);

if (numtasks != 4) {

cout << "Need 4 players";

MPI\_Finalize();}

/\* determina rangul procesului apelant \*/

MPI\_Comm\_rank(MPI\_COMM\_WORLD, &rank);

Dupa intoarcerea din toate metodele apelate din corpul programului, este necesar apelul **MPI\_Finalize().**Aceasta inchide toate apelurile specific MPI, nemaiputandu-se realiza nicun alt apel MPI dupa acesta.

PERFORMANTA

Luand in considerare numerele mici in jurul carora se invarte fiecare runda a jocului, de complexitate redusa, avem un timp constant O(1).

Mai exact, sunt contorizate doar apelurile care se fac intre procese.

Timpul necesar executiei in paralel este definit astfel:

tp = t comp+tcomm

Primul argument este timpul computational, in acest caz, constant : O(1).

Cel de-al doilea argument, timpul necesar comunicarii este dat de formula

tcomm = tstartup + n\*tdata .

Aplicand formula pe algoritmul nostru, stiind ca intr-un joc, fiecare proces lanseaza cate 14 semnale, rezulta:

tcomm = 4 \* tstartup + 52\*tdata + 14\*tdata

tcomm = 4 \* tstartup + 66\*tdata

Se obtine :

tp = O(1) + 4 \* tstartup + 66\*tdata.