Paralelno programiranje

Druga domaća zadaća

Zadatak: uporabom MPI-a ostvariti program za igranje uspravne igre *"4 u nizu"* (*connect 4*) za jednog igrača (čovjek protiv računala).

Opis igre: igra je istovjetna igri križić-kružić u kojoj je cilj napraviti niz od 4 igračeva znaka, s tom razlikom da se odvija na 'uspravnom' 2D polju u kojemu se novi znak može staviti samo na polje ispod kojega već postoji neki znak ili se stavlja na dno polja ('ajmo reć' uz gravitaciju). Standardne dimenzije igračeg polja su 6 polja u visinu i 7 u širinu, mada je veličina proizvoljna. Zbog jednostavnosti, može se pretpostaviti da polje nije ograničeno u visinu, dok ograničenje u širinu mora postojati zbog ograničavanja broja mogućih poteza iz zadanog stanja.

Opis slijednog algoritma: neka metoda rješavanja bude djelomično pretraživanje prostora stanja, u obliku stabla, do neke zadane dubine od trenutnog stanja. Dakle, ne pokušavamo *naučiti* strategiju, već se za svaki potez računala obavlja pretraga podstabla i odabire sljedeće stanje (*brute force* pristup). Za svako se stanje u stablu određuje vrsta:

- stanje je 'pobjeda' ako računalo ima 4 u nizu (vrijednost 1);
- stanje je 'poraz' ako igrač ima 4 u nizu (vrijednost -1);
- inače, stanje je neutralno, a vrijednost će ovisiti o stanjima u podstablu (ako se podstabla pretražuju).

HINT: potragu za 4 u nizu treba obaviti samo sa polja posljednjeg odigranog poteza.

Nakon pretraživanja stabla do zadane dubine, primjenjuju se sljedeća rekurzivna pravila:

1. ako je neko stanje 'pobjeda' i ako se u njega dolazi potezom računala, tada je i nadređeno stanje također 'pobjeda' (jer računalo iz nadređenog stanja uvijek može pobijediti, potezom koji vodi u stanje pobjede);
2. ako je stanje 'poraz' i ako se u njega dolazi potezom igrača (protivnika), tada je i nadređeno stanje 'poraz' (jer iz nadređenog stanja igrač može jednim potezom pobijediti);
3. ako su sva podstanja nekog stanja 'pobjeda' ili 'poraz', tada je i nadređeno stanje iste vrste.

Osim ovim pravilima, svaki se mogući potez računala (odnosno stanje u koje se tim potezom dolazi) ocjenjuje promatranjem broja i dubine *pobjedničkih* stanja u podstablu u koje taj potez vodi. Mjera kvalitete poteza (jednog podstabla) definira se rekurzivno kao razlika broja pobjedničkih stanja i broja poraza na nekoj dubini, podijeljenih sa umnoškom broja mogućih poteza na toj dubini, odnosno:

(broj_pobjeda_u_dubini_n - broj_poraza_u_dubini_n)/(broj_mogućih_poteza)

Broj mogućih poteza se može smatrati konstantnim (npr. 7) uz pretpostavku o neograničenosti polja u visinu, a u stvarnim uvjetima potrebno je uzeti u obzir samo moguće poteze (ako je neki stupac popunjen). Računalo tada odabire onaj potez koji ne vodi u stanje 'poraz' (ako ima izbora) a koji ima najveću vrijednost (vrijednosti su po opisanoj definiciji u intervalu [-1,1]). Eventualna dodatna pojašnjenja dana su na predavanjima.

NAPOMENA: moguće je uporabiti i neku drugu (bolju) funkciju ocjene poteza!.

Ostvarenje paralelnog algoritma: Program treba imati minimalno tekstno sučelje u obliku prikaza stanja polja i upita igrača o potezu. Računanje vrijednosti pojedinog poteza treba načiniti raspodijeljeno, a minimalna dubina pretraživanja stabla *n* je 4 (složenost je 7^*n*). Minimalni broj zadataka paralelnog algoritma je broj mogućih poteza (7), no taj broj je potrebno povećati (npr. dijeljenjem pri većoj dubini) poradi boljeg ujednačavanja opterećenja po procesorima.

Algoritam u kojemu postoji stalan broj od najviše 7 zadataka (radnika) nije prihvatljiv!

Predaja zadaće:

- Paralelni algoritam je potrebno **dokumentirati** tako da se ukratko opišu četiri faze razvoja: podjela, komunikacija, aglomeracija i pridruživanje (samo elektronički dokument).
- Osim same implementacije, potrebno je empirijski utvrditi **ubrzanje i učinkovitost** algoritma (definicija na predavanju) mjerenjem trajanja programa (trajanje jednog poteza računala na samom početku igre) na broju **procesora** P = 1,..,8. Mjerenje je potrebno obavljati na početku igre kada su uvjeti jednaki, a rezultate pripremiti u elektroničkom obliku, grafički i tablično. Mjerenja treba provesti tako da je najmanje mjereno trajanje (za 8 procesora) **reda veličine barem nekoliko sekundi** (definirajte potrebnu dubinu pretraživanja). Za mjerenje je potrebno 8 procesora odnosno 8 jezgara (*hyperthreading* se u pravilu ne broji!) *Dobivene rezultate potrebno je komentirati (obrazložiti)*

Primjer rezultata: Npr. ako su rezultati mjerenja sljedeći:

broj procesora (P)	1	2	3	4	5	6	7	8
trajanje (s)	20	11	8.5	7	5.8	5	4.5	4

prikaz ubrzanja i učinkovitosti izgledao bi ovako: