Igra z umetno inteligenco

Algoritem minimax

Minimax je splošen algoritem za igre dveh igralcev

- Rabimo metodo, ki da numerično oceno poziciji igre.

  Pozitivna ocena pomeni, da je pozicija dobra za umetnega igralca. Negativna ocena pomeni, da je slaba.

  Ocena je približek kakovosti pozicije.

- Algoritem minimax raziskuje drevo možnih potez do globine, ki mu damo na začetku.

  Algoritem predlaga najboljšo potezo glede ocen.

Implementacija v aplikaciji TicTacToe

```
private static final int ZMAGA = 100; // vrednost zmage
private static final int ZGUBA = -ZMAGA;  // vrednost izgube
private static final int NEODLOC = 0;  // vrednost neodločene igre
```

```
public class OcenjenaPoteza {

    Poteza poteza;
    int ocena;

    public OcenjenaPoteza (Poteza poteza, int ocena) {
        this.poteza = poteza;
        this.ocena = ocena;
    }
}
```

Polji `poteza` in `ocena` sta brez določila. To pomeni, da sta dostopni samo od paketa (package private). Vsi razredi, ki se tičejo inteligence, so v paketi `Inteligenca`.

Rekursivni algoritem za minimax v Javi

```
public OcenjenaPoteza minimax(Igra igra, int globina, Igralec jaz) {
    OcenjenaPoteza najboljsaPoteza = null;
    List<Poteza> moznePoteze = igra.poteze();
    for (Poteza p: moznePoteze) {
        Igra kopijaIgre = new Igra(igra);
        kopijaIgre.odigraj (p);
        int ocena;
        switch (kopijaIgra.stanje()) {
        case ZMAGA_O: ocena = (jaz == Igralec.O ? ZMAGA : ZGUBA); break;
        case ZMAGA_X: ocena = (jaz == Igralec.X ? ZMAGA : ZGUBA); break;
        case NEODLOCENO: ocena = NEODLOC; break;
        default: // nekdo je na potezi
            if (globina == 1) ocena = oceniPozicijo(kopijaIgre,jaz);
            else ocena = minimax(kopijaIgre, globina-1, jaz).ocena;
        }
        // ali je p z oceno boljsa kot najboljsaPoteza?
        if (najboljsaPoteza == null
            // odlocimo min ali max, odvisno od igralca, ki je igral p
            || igra.naPotezi()==jaz && ocena > najboljsaPoteza.ocena  //max
            || igra.naPotezi()!=jaz && ocena < najboljsaPoteza.ocena) //min
           najboljsaPoteza = new OcenjenaPoteza (p, ocena);
    }
    return najboljsaPoteza;
}
```

Pomembne točke

- Vrednost parametra `globlina` mora biti ≥ 1, in ne sme igra `igra` biti že končana.
- Vrednosti parametrov `globlina` in `igra` se spremenita, medtem ko se izvaja algoritem. Vrednost parametra `jaz` se ne spremeni.
- Naredimo novo kopijo igre za vsako možno potezo `p`.
- Zelo pomembno je, da so polji `plosca` in `naPotezi` razreda `Igra` sta dinamični. Zato, ko igramo s kopjo igre `kopijaIgre`, ne spremenimo originalno igro `igra`.

Paket `inteligenca` ima razrede:

- `Inteligenca`

  Abstraktni razred, ki ima abstraktno metodo

  ```
  Poteza izberiPotezo (Igra igra)
  ```

- `Minimax`

  Podrazred razreda `Inteligenca`. Metoda `izberiPotezo` je implementacija algoritma minimax.

- `OceniPozicijo`

  Razred ima statično metodo

  ```
  int oceniPozicijo(Igra igra, Igralec jaz)
  ```

  ki oceni trenutno pozicijo igre `igra` z vidike igralca `jaz`.

- `OcenjenaPozicija`

  Videli smo že ta razred.

Razred Minimax

```
public class Minimax extends Inteligenca {

    private static final int ZMAGA = 100;
    private static final int ZGUBA = -ZMAGA;
    private static final int NEODLOC = 0;

    private int globina;

    public Minimax (int globina) {
        super("minimax globina " + globina);
        this.globina = globina;
    }

    @Override
    public Poteza izberiPotezo (Igra igra) {
        OcenjenaPoteza najboljsaPoteza =
                minimax(igra, this.globina, igra.naPotezi());
        return najboljsaPoteza.poteza;
    }

    public OcenjenaPoteza minimax(Igra igra,int globina,Igralec jaz) { ... }
}
```

V razredu Vodja

```
public static Inteligenca racunalnikovaInteligenca = new Minimax(3);

public static void igrajRacunalnikovoPotezo() {
    Igra zacetkaIgra = igra;
    SwingWorker<Poteza, Void> worker =
            new SwingWorker<Poteza, Void> () {
        @Override
        protected Poteza doInBackground() {
            Poteza poteza = racunalnikovaInteligenca.izberiPotezo(igra);
            try {TimeUnit.SECONDS.sleep(1);} catch (Exception e) {};
            return poteza;
        }
        @Override
        protected void done () {
            Poteza poteza = null;
            try {poteza = get();} catch (Exception e) {};
            if (igra == zacetkaIgra) {
                igra.odigraj(poteza);
                igramo ();
            }
        }
    };
    worker.execute();
}
```

Konstructor v razredu Igra, ki naredi kopijo igre

Nepravilno:

```
public Igra(Igra igra) {
    this.plosca = igra.plosca;
    this.naPotezi = igra.naPotezi;
}
```

Pravilno:

```
public Igra(Igra igra) {
    this.plosca = new Polje[N][N];
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            this.plosca[i][j] = igra.plosca[i][j];
        }
    }
    this.naPotezi = igra.naPotezi;
}
```

Pazite na identiteto objektov!

Primerjajte:

```
int[] x = {1,2,3};
int[] y = x;
x[1] = 5;
System.out.println(Arrays.toString(y));
```

```
[1, 5, 3]
```

```
Igralec i = Igralec.O;
Igralec j = i;
i = Igralec.X;
System.out.println(j);
```

```
O
```

Obrezovanje alfa-beta

Obrezovanje alfa-beta (alpha-beta pruning) je izboljšava algoritma minimax, ki raziskuje drevo samo kjer je raziskava potrebna.

- α je spodnja meja vrednosti najboljše poteze, ko maksimiramo vrednost.
- β je zgornja meja, ko minimiziramo vrednost.
- Ko je $\beta \leq \alpha$, ni nam treba raziskovati poddrevo kjer smo, ker najboljša poteza ne pride do tega poddrevesa.

Algoritem za obrezovanje alfa-beta v Javi (1/2)

```
public OcenjenaPoteza alphabetaPoteze(Igra igra, int globina,
        int alpha, int beta, Igralec jaz) {
    int ocena;
    // Če sem računalnik, maksimiramo oceno z začetno oceno ZGUBA
    // Če sem pa človek, minimiziramo oceno z začetno oceno ZMAGA
    if (igra.naPotezi() == jaz) {ocena = ZGUBA;} else {ocena = ZMAGA;}
    List<Poteza> moznePoteze = igra.poteze();
    Poteza kandidat = moznePoteze.get(0); // Ne more biti null.
    for (Poteza p: moznePoteze) {
        Igra kopijaIgre = new Igra(igra);
        kopijaIgre.odigraj (p);
        int ocenap;
        switch (kopijaIgre.stanje()) {
        case ZMAGA_O: ocenap = (jaz == Igralec.O ? ZMAGA : ZGUBA); break;
        case ZMAGA_X: ocenap = (jaz == Igralec.X ? ZMAGA : ZGUBA); break;
        case NEODLOCENO: ocenap = NEODLOC; break;
        default: // Nekdo je na potezi
            if (globina == 1) ocenap = oceniPozicijo(kopijaIgre, jaz);
            else ocenap = alphabetaPoteze
                (kopijaIgre, globina-1, alpha, beta, jaz).ocena;
        }

        ...
```

Algoritem za obrezovanje alfa-beta v Javi (2/2)

```
        ...

        if (igra.naPotezi() == jaz) { // Maksimiramo oceno
            if (ocenap > ocena) { // mora biti > namesto >=
                ocena = ocenap;
                kandidat = p;
                alpha = Math.max(alpha,ocena);
            }
        } else { // igra.naPotezi() != jaz, torej minimiziramo oceno
            if (ocenap < ocena) { // mora biti < namesto <=
                ocena = ocenap;
                kandidat = p;
                beta = Math.min(beta, ocena);
            }
        }
        if (alpha >= beta) // Ostale poteze ne pomagajo
            return new OcenjenaPoteza (kandidat, ocena);
    }
    return new OcenjenaPoteza (kandidat, ocena);
}
```