

mini project3 report

1. Calling tree search

```
517 void write_valid_spot(std::ofstream& fout) {
518
519     srand(time(NULL));
520     Point p = next_valid_spots[0];
521     fout << p.x << " " << p.y << std::endl;
522     fout.flush();
523     // =====
524     // find good moves here
525     // black =1 =maximizer
526     myOthello cur;
527     cur.set(board);
528
529     if(cur.disc_count[0] != 64-4){
530         cur.cur_player = player;
531         cur.next_valid_spots = next_valid_spots;
532
533         for(int i = 1; i <= 9; i += 2){
534             MaxDepth = i;
535             cur.heuristic = abprune(cur,MaxDepth, INT32_MIN, INT32_MAX);
536             for(auto i:h_map){
537                 if(i.first == cur.heuristic){
538                     p.x = i.second.x;
539                     p.y = i.second.y;
540                     break;
541                 }
542             }
543             h_map.clear();
544             fout << p.x << " " << p.y << std::endl;
545             fout.flush();
546         }
547     }
548     // =====
549     // Remember to flush the output to ensure the last action is written to file.
550     fout << p.x << " " << p.y << std::endl;
551     fout.flush();
552 }
```

2. Tree search & abprune

```
462 int abprune(myOthello curnode, int depth, int alpha, int beta){
463     bool maximizer = curnode.cur_player==1;
464
465     if(depth == 0 || curnode.done){
466         return heuristic(curnode);
467     }
468     if(maximizer){
469         int maxeval = INT32_MIN;
470         for(auto i:curnode.next_valid_spots){
471             myOthello next = curnode;
472             if(!next.put_disc(i)){
473                 cout << "hi\n";
474                 continue;
475             }
476             else{
477                 int eval = abprune( next,depth-1,alpha,beta);
478                 maxeval = max(maxeval,eval);
479                 if(depth == MaxDepth)
480                     h_map.insert(pair<int,Point>(eval,i));
481
482                 alpha = max(alpha,eval);
483                 if(beta <= alpha)
484                     break;
485             }
486         }
487         curnode.heuristic = maxeval;
488         return maxeval;
489     }
```

```
490     else{
491         int mineval = INT32_MAX;
492         for(auto i:curnode.next_valid_spots){
493             myOthello next = curnode;
494             if(!next.put_disc(i)){
495                 cout << "hi\n";
496                 continue;
497             }
498             else{
499                 int eval = abprune( next,depth-1,alpha,beta);
500                 mineval = min(mineval,eval);
501                 if(depth == MaxDepth)
502                     h_map.insert(pair<int,Point>(eval,i));
503
504                 beta = min(beta,eval);
505                 if(beta <= alpha)
506                     break;
507             }
508         }
509         curnode.heuristic = mineval;
510         return mineval;
511     }
512
513 } // end function
```

3. State value function design

```
439 int heuristic(myOthello cur){
440     int heuristic = 0;
441     if(cur.disc_count[0] >= 44){
442         // opening game
443         heuristic = 10000*count_corners(cur)
444                     + 10000*count_stability(cur)
445                     + 1000*count_line(cur)
446                     + 20*count_weight(cur)
447                     + 5*count_mobility(cur)
448                     + 1000*count_xc(cur);
449     }
450
451     else if(cur.disc_count[0] >= 6){
452         heuristic = 10000*count_corners(cur)
453                     + 10000*count_stability(cur)
454                     + 1000*count_line(cur)
455                     + 10*count_weight(cur)
456                     + 2*count_mobility(cur)
457                     + 2000*count_xc(cur);
458     }
459     else{
460         // end game
461         heuristic = 10000*count_corners(cur)
462                     + 10000*count_stability(cur)
463                     + 1000*count_line(cur)
464                     + 3000*count_xc(cur);
465         //+ 10*count_weight(cur)
466         //+ 300*count_mobility(cur);
467     }
```

```

32 | int MaxDepth = 7;
33 | std::array<std::array<int, SIZE>, SIZE> board;
34 | std::vector<Point> next_valid_spots;
35 | std::array<Point, 4> corners{{
36 |     Point(0, 0), Point(0,7), Point(7,0),Point(7,7)
37 | }}
38 | };
39 | std::array<Point, 8> c_spots{{
40 |     Point(0, 1), Point(1,0), Point(0,6),Point(1,7),
41 |     Point(6, 0), Point(7,1), Point(7,6),Point(6,7)
42 | }}
43 | };
44 | std::array<Point, 4> x_spots{{
45 |     Point(1, 1), Point(1,6), Point(6,1),Point(6,6)
46 | }}
47 | };
48 | std::array<Point, 8> dir{{
49 |     Point(1, 0), Point(0,1),
50 |     Point(1, 0), Point(0,-1),
51 |     Point(-1, 0), Point(0,1),
52 |     Point(-1,0), Point(0,-1)
53 | }}
54 | };
55 | std::array<Point, 4> dir_stability{{
56 |     Point(-1, 1),
57 |     Point(-1,-1),
58 |     Point(1, 1),
59 |     Point(1,-1)
60 | }}
61 | };
62 | std::array<Point, 4> center{{
63 |     Point(3,3),
64 |     Point(3,4),
65 |     Point(4,3),
66 |     Point(4,4)
67 | }}
68 | };

```

```

299 | //heuristic little functions
300 | > int count_corners(myOthello cur){ ...
311 |
312 | > int count_line(myOthello cur){ ...
335 |
336 | > int count_mobility(myOthello cur){ ...
345 |
346 | > int count_weight(myOthello cur){ ...
362 |
363 | > int count_xc(myOthello cur){ ...
395 |
396 | > int count_stability(myOthello cur){ ...
436 |
437 | // end functions

```

```

300 int count_corners(myOthello cur){
301     int bk = 0, wh = 0;
302
303     for(auto i:corners){
304         if( cur.board[i.x][i.y] == 1)
305             bk ++;
306         else if(cur.board[i.x][i.y] == 2)
307             wh ++;
308     }
309     return (bk-wh);
310 }
311
312 int count_line(myOthello cur){
313
314     // 角 連邊
315     int bk_linecount = 0;
316     int wh_linecount = 0;
317     for(int i = 0; i < 4; i ++){
318
319         int color;
320         Point c = corners[i];
321         if(!cur.board[c.x][c.y]) continue;
322         else color = cur.board[c.x][c.y];
323         for(int j = 0; j < 2; j ++){
324             Point next = c + dir[2*i+j];
325             while(cur.board[next.x][next.y] == color && on_board(next)){
326                 if(color == 1) bk_linecount++;
327                 else wh_linecount++;
328                 next = next + dir[2*i+j];
329             }
330         }
331     }
332
333     return (bk_linecount-wh_linecount);
334 }

```

```

336 int count_mobility(myOthello cur){
337     bool maximizer = cur.cur_player==1;
338
339     // Mobility
340     //const int weight_mobility = 30;
341     if(maximizer)
342         return cur.next_valid_spots.size();
343     else return -1* cur.next_valid_spots.size();
344 }

```

```

346 int count_weight(myOthello cur){
347     int heuristic = 0;
348
349     int bk = 0;
350     int wh = 0;
351     for(int i = 0; i < 8; i++){
352         for(int j = 0; j < 8; j++){
353             if(!cur.board[i][j] || !weightmap[i][j]) continue;
354             if(cur.board[i][j] == 1)
355                 bk += weightmap[i][j];
356             else wh -= weightmap[i][j];
357         }
358     }
359     heuristic += (bk-wh)*10;
360     return heuristic;
361 }

```

```

363 int count_xc(myOthello cur){
364     int heuristic = 0;
365     int bk = 0;
366     int wh = 0;
367     // x-squares
368     const int weight_x = 1000;
369     for(int i = 0; i < 4; i++){
370         Point C0 = corners[i];
371         Point X = x_spots[i];
372         if( !cur.board[X.x][X.y] && cur.board[X.x][X.y] != cur.board[C0.x][C0.y] ){
373             if(cur.board[X.x][X.y] == 1)
374                 bk -= weight_x;
375             else wh += weight_x;
376         }
377     }
378
379     // c-squares
380     const int weight_c = 1000;
381     for(int i = 0; i < 4; i++){
382         Point C0 = corners[i];
383         for(int j = 0; j < 2; j++){
384             Point C = c_spots[2*i+j];
385             if( !cur.board[C.x][C.y] && cur.board[C.x][C.y] != cur.board[C0.x][C0.y] ){
386                 if(cur.board[C.x][C.y] == 1)
387                     bk -= weight_c;
388                 else wh += weight_c;
389             }
390         }
391     }
392     heuristic = bk-wh;
393     return heuristic;
394 }


```

```

388 int count_stability(myOthello cur){
389     int heuristic = 0;
390     bool wingame = false;
391
392     for(int i = 0; i < 4; i ++){
393         // 對這四個角探討穩固性
394         Point co = corners[i];
395         if(!cur.board[co.x][co.y]) continue;
396         int color = cur.board[co.x][co.y];
397         int lv = 1;
398         Point next = co + dir[i*2];
399         bool good = true;
400         while(good){
401             if(lv == 8){
402                 wingame = true;
403                 break;
404             }
405             if(!on_board(next)){
406                 lv ++;
407                 Point nt;
408                 nt.x = co.x + lv*dir[i*2].x;
409                 nt.y = co.y + lv*dir[i*2].y;
410                 next = nt;
411             }
412             else if(cur.board[next.x][next.y] == color){
413                 next = next + dir_stability[i*2];
414             }
415             else{
416                 good = false;
417                 break;
418             }
419         }
420         if(color == 1)
421             heuristic += lv;
422         else heuristic -= lv;
423         if (wingame)
424             heuristic += 10000;
425     }
426     return heuristic;
427 }

```

4. Version control

 ccl1616 / pj3

Unwatch

1

Star

0

Fork

0

<> Code

Issues

Pull requests

Actions

Projects

Wiki

Security

Insights

Settings

Branch: master

Commits on Jun 28, 2020

ai6 added
ccl1616 committed 2 hours ago

5:4, 39:25
ccl1616 committed 3 hours ago

ai5 implemented
ccl1616 committed 4 hours ago

Update README.md
ccl1616 committed 6 hours ago

ai4 currently best, ignore ai5
ccl1616 committed 6 hours ago

some heuristic update
ccl1616 committed 11 hours ago

Commits on Jun 27, 2020


accidentally ai3>ai4, ai4 added stability
ccl1616 committed 23 hours ago

ai3 is now best
ccl1616 committed yesterday

update
ccl1616 committed yesterday

ai4 not working
ccl1616 committed yesterday

ai3 done, wightmap implemented
ccl1616 committed yesterday

 Search or jump to...

Pull requests

Issues

Marketplace

Explore

Unwatch

1

Star

0

Fork

0

<> Code

Issues

Pull requests

Actions

Projects

Wiki

Security

Insights

Settings

Branch: master

Go to file

Add file

Clone

ccl1616 committed 369ff04 2 hours ago

41 commits

1 branch

0 tags

.vscode	minimax sometimes failed	2 days ago
.DS_Store	ai4 currently best, ignore ai5	7 hours ago
README.md	Update README.md	7 hours ago
ai_0	ai3 is now best	yesterday
ai_0.cpp	ai3 is now best	yesterday
ai_1	update	yesterday
ai_1.cpp	update	yesterday
ai_2	accidentally ai3>ai4, ai4 added stability	23 hours ago
ai_2.cpp	accidentally ai3>ai4, ai4 added stability	23 hours ago
ai_3	ai4 currently best, ignore ai5	7 hours ago
ai_3.cpp	ai4 currently best, ignore ai5	7 hours ago
ai_4	5:4, 39:25	4 hours ago
ai_4.cpp	5:4, 39:25	4 hours ago
ai_5	5:4, 39:25	4 hours ago
ai_5.cpp	5:4, 39:25	4 hours ago
ai_6	ai6 added	2 hours ago

About

No description, website, or topics provided.

Readme

Releases

No releases published
[Create a new release](#)

Packages

No packages published
[Publish your first package](#)

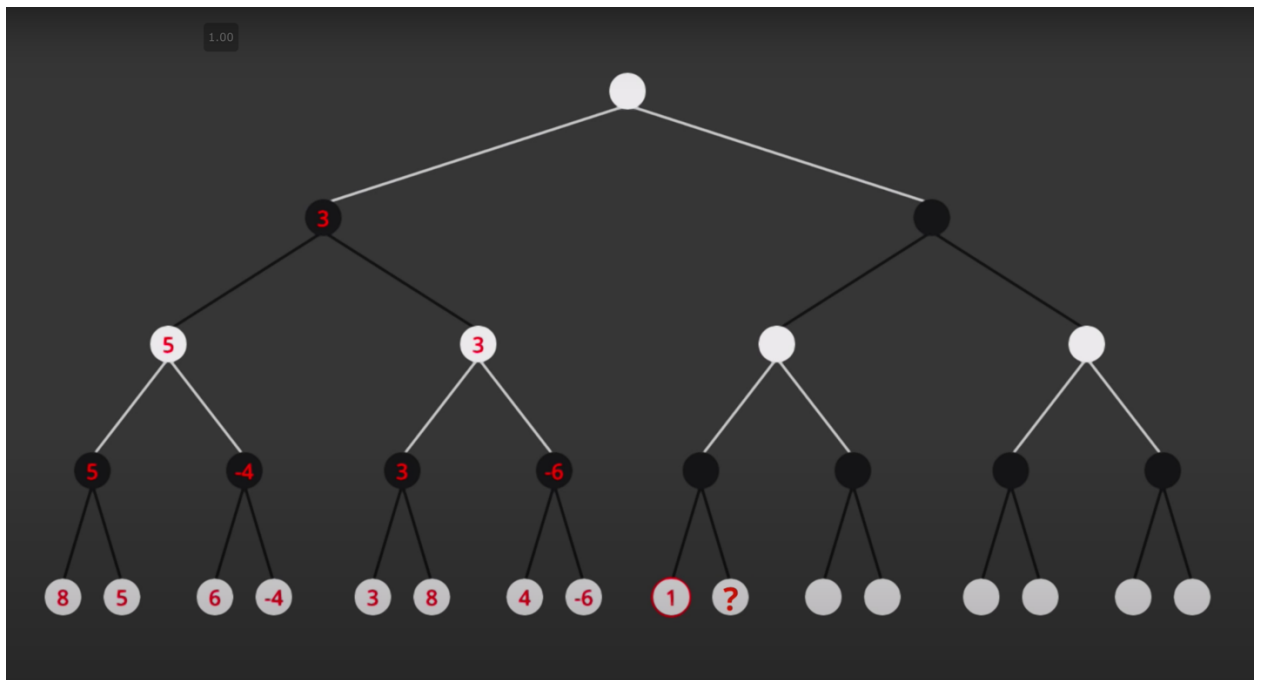
Languages

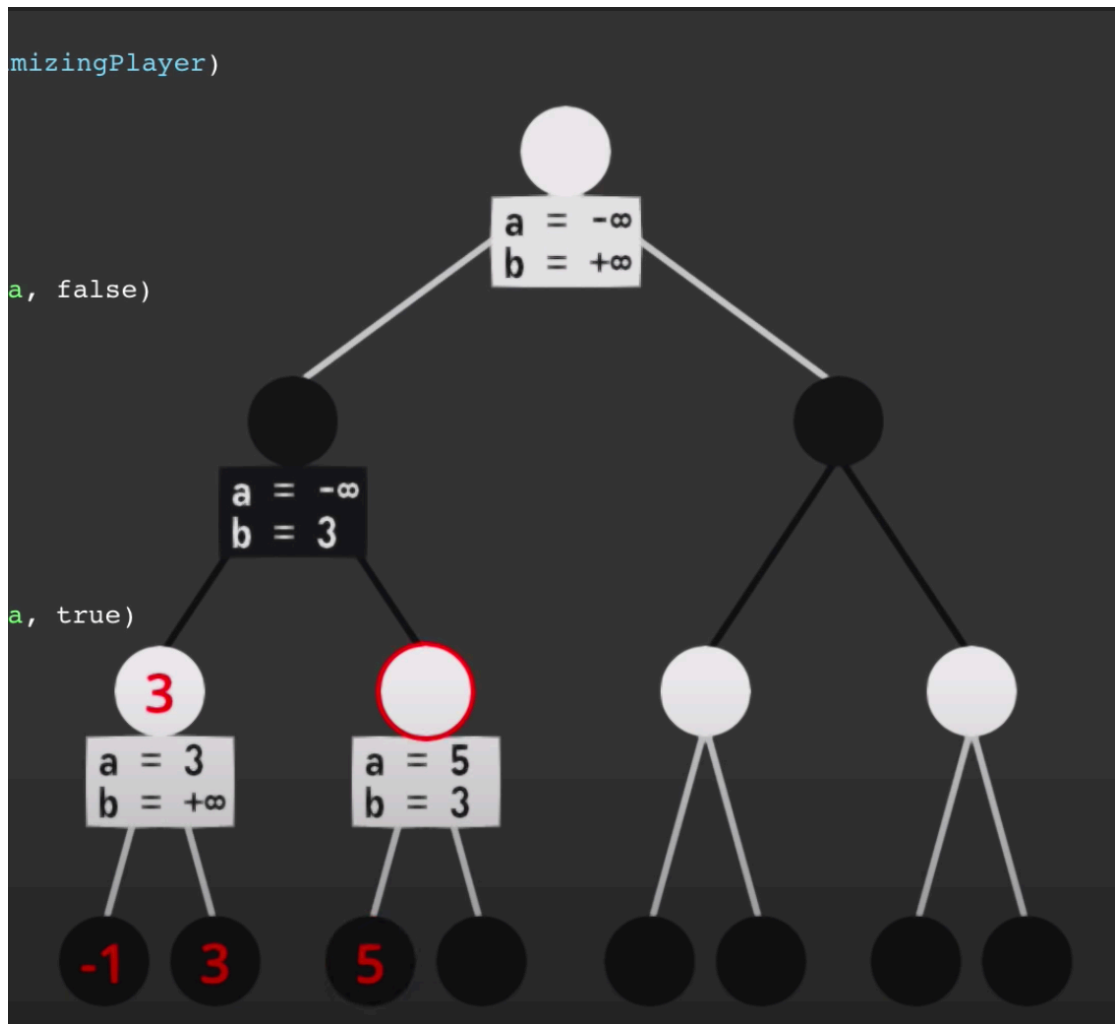
C++ 99.5%

Makefile 0.5%

5. Abprune

<https://www.youtube.com/watch?v=l-hh51ncgDI>





6. Reference

<https://github.com/eigenfoo/otto-othello>