# Appendices

## Appendix A　偶然気づいたこと

**定義 A.1.** $2n \times 2n$ のオセロ盤を考える.$(i,j)_{2n}$ で左下の角マスから, 上に $i-1$, 右に $j-1$ 進んだマスを表すものとする.

具体的には $(1,1)_{2n}$ は左下の角マス,$(2n,2n)_{2n}$ は右上の角マスとなる.

**注意** 以下より, オセロのとある試合を考えることになるのだが, 最初の石の配置は $(n,n+1)_{2n},(n+1,n)_{2n}$ には黒,$(n,n)_{2n},(n+1,n+1)_{2n}$ を採用する. また, 先攻は黒とする.

**定義 A.2.** 「$(i,j)_{2n}$ が辞書順で $(i',j')_{2n}$ より小さい」は「$i<i'$ または『$i=i'$ かつ $j<j'$』」と定義する.

**定義 A.3.** 置けるマスすべてのうち, 定義 $A.1$ のように見たときに辞書順で最小のものを選ぶことを ***nice-select*** と言う.

さて, 次のようなことに気が付いた:

**命題 A.1.** $6 \times 6$ のオセロ盤で, 黒も白も nice-select し続けると, 引き分けになる.

オセロ盤の大きさを変えて実験してみた. その結果をまとめると, 次のようになった.

**命題 A.2.** $2m \times 2m$ $(4 \leqslant m \leqslant 100)$ のオセロ盤で, 黒も白も nice-select し続けると,$m = 46,54,74,78,90,94$ のときは, 先攻が勝ち, それ以外では, 後攻が勝つ.

$200 \times 200$ まではほとんど後攻が勝つことが確認できた. これを踏まえて次の予想を考えた.

**予想 A.1.** 3 以上の整数 $n$ であって,$2n \times 2n$ のオセロ盤で, 黒も白も nice-select し続けたとき, 引き分けになるのは $n=3$ だけである.

**予想 A.2.** 3 以上の整数 $n$ であって,$2n \times 2n$ のオセロ盤で, 黒も白も nice-select し続けたとき, 先攻が勝つのは $n = 46,54,74,78,90,94$ 以外にも存在していて, それは無数にある.

**予想 A.3.** 3 以上の整数 $n$ に対して,$a_n, b_n, c_n$ を次のように定義する.

- $3 \leqslant i \leqslant n$ をみたす整数 $i$ であって,$2i \times 2i$ のオセロ盤で, 黒も白も nice-select し続けるて後攻が勝つようなものの総数を $a_n$, そうでないものを $b_n$ とする.(引き分けも含まれる)
- $c_n = a_n/b_n$.

この時,$c_N$ は $N \to \infty$ とすると $c_N \to \infty$ となる.

## Appendix B 実験結果まとめ

初期の色の配置を入れ替えたとき, 言葉だけでは説明し難い結果となった. この結果も併せて以下の表にした.(左が通常, 右が色反転)

| board size | black | white |
|---|---|---|
| 6×6 | 18 | 18 |
| 8×8 | 19 | 45 |
| 10×10 | 35 | 65 |
| 12×12 | 61 | 83 |
| 14×14 | 60 | 136 |
| 16×16 | 86 | 170 |
| 18×18 | 93 | 231 |
| 20×20 | 161 | 239 |
| 22×22 | 198 | 286 |
| 24×24 | 278 | 298 |
| 26×26 | 251 | 425 |
| 28×28 | 336 | 448 |
| 30×30 | 385 | 515 |
| 32×32 | 440 | 584 |
| 34×34 | 472 | 684 |
| 36×36 | 564 | 732 |
| 38×38 | 599 | 845 |
| 40×40 | 708 | 892 |
| 42×42 | 777 | 987 |
| 44×44 | 912 | 1024 |
| 46×46 | 931 | 1185 |
| 48×48 | 1046 | 1258 |
| 50×50 | 1068 | 1432 |
| 52×52 | 1287 | 1417 |
| 54×54 | 1334 | 1582 |
| 56×56 | 1445 | 1691 |

| board size | black | white |
|---|---|---|
| 6×6 | 24 | 12 |
| 8×8 | 40 | 24 |
| 10×10 | 43 | 57 |
| 12×12 | 104 | 40 |
| 14×14 | 54 | 142 |
| 16×16 | 162 | 94 |
| 18×18 | 85 | 239 |
| 20×20 | 149 | 251 |
| 22×22 | 191 | 293 |
| 24×24 | 218 | 358 |
| 26×26 | 252 | 424 |
| 28×28 | 522 | 262 |
| 30×30 | 322 | 578 |
| 32×32 | 542 | 482 |
| 34×34 | 675 | 481 |
| 36×36 | 651 | 645 |
| 38×38 | 820 | 624 |
| 40×40 | 814 | 786 |
| 42×42 | 960 | 804 |
| 44×44 | 828 | 1108 |
| 46×46 | 1194 | 922 |
| 48×48 | 1184 | 1120 |
| 50×50 | 1416 | 1084 |
| 52×52 | 1178 | 1526 |
| 54×54 | 1546 | 1370 |
| 56×56 | 1579 | 1557 |

| | | |
|---|---|---|
| 58×58 | 1474 | 1890 |
| 60×60 | 1672 | 1928 |
| 62×62 | 1717 | 2127 |
| 64×64 | 1896 | 2200 |
| 66×66 | 1894 | 2462 |
| 68×68 | 2300 | 2324 |
| 70×70 | 2200 | 2700 |
| 72×72 | 2293 | 2891 |
| 74×74 | 2383 | 3093 |
| 76×76 | 2844 | 2932 |
| 78×78 | 2564 | 3520 |
| 80×80 | 2876 | 3524 |
| 82×82 | 3029 | 3695 |
| 84×84 | 3417 | 3639 |
| 86×86 | 3276 | 4120 |
| 88×88 | 3477 | 4267 |
| 90×90 | 3583 | 4517 |
| 92×92 | 4251 | 4213 |
| 94×94 | 3943 | 4893 |
| 96×96 | 4502 | 4714 |
| 98×98 | 4211 | 5393 |
| 100×100 | 4805 | 5195 |
| 102×102 | 4644 | 5760 |
| 104×104 | 5023 | 5793 |
| 106×106 | 5183 | 6053 |
| 108×108 | 5914 | 5750 |
| 110×110 | 5361 | 6739 |
| 112×112 | 5987 | 6557 |
| 114×114 | 5675 | 7321 |
| 116×116 | 6669 | 6787 |
| 118×118 | 6573 | 7351 |
| 120×120 | 6317 | 8083 |
| 122×122 | 6420 | 8464 |
| 124×124 | 7352 | 8024 |
| 126×126 | 7297 | 8579 |
| 128×128 | 7666 | 8718 |
| 130×130 | 7791 | 9109 |
| 132×132 | 8577 | 8847 |
| 134×134 | 8037 | 9919 |

| | | |
|---|---|---|
| 58×58 | 1872 | 1492 |
| 60×60 | 1540 | 2060 |
| 62×62 | 1963 | 1881 |
| 64×64 | 1977 | 2119 |
| 66×66 | 2453 | 1903 |
| 68×68 | 1990 | 2634 |
| 70×70 | 2690 | 2210 |
| 72×72 | 2719 | 2465 |
| 74×74 | 2835 | 2641 |
| 76×76 | 2523 | 3253 |
| 78×78 | 3337 | 2747 |
| 80×80 | 3074 | 3326 |
| 82×82 | 3141 | 3583 |
| 84×84 | 2944 | 4112 |
| 86×86 | 4144 | 3252 |
| 88×88 | 3393 | 4351 |
| 90×90 | 4511 | 3589 |
| 92×92 | 3770 | 4694 |
| 94×94 | 4664 | 4172 |
| 96×96 | 4341 | 4875 |
| 98×98 | 5352 | 4252 |
| 100×100 | 3959 | 6041 |
| 102×102 | 5273 | 5131 |
| 104×104 | 4766 | 6050 |
| 106×106 | 6211 | 5025 |
| 108×108 | 4723 | 6941 |
| 110×110 | 7001 | 5099 |
| 112×112 | 5161 | 7383 |
| 114×114 | 7277 | 5719 |
| 116×116 | 5635 | 7821 |
| 118×118 | 7958 | 5966 |
| 120×120 | 7005 | 7395 |
| 122×122 | 8352 | 6532 |
| 124×124 | 6229 | 9147 |
| 126×126 | 9009 | 6867 |
| 128×128 | 8651 | 7733 |
| 130×130 | 9584 | 7316 |
| 132×132 | 8004 | 9420 |
| 134×134 | 9743 | 8213 |

| | | |
|---|---|---|
| 136×136 | 8787 | 9709 |
| 138×138 | 8189 | 10855 |
| 140×140 | 9656 | 9944 |
| 142×142 | 8942 | 11222 |
| 144×144 | 9892 | 10844 |
| 146×146 | 9856 | 11460 |
| 148×148 | 11143 | 10761 |
| 150×150 | 9968 | 12532 |
| 152×152 | 10564 | 12540 |
| 154×154 | 11029 | 12687 |
| 156×156 | 12492 | 11844 |
| 158×158 | 11683 | 13281 |
| 160×160 | 12279 | 13321 |
| 162×162 | 11582 | 14662 |
| 164×164 | 12604 | 14292 |
| 166×166 | 12031 | 15525 |
| 168×168 | 13122 | 15102 |
| 170×170 | 12921 | 15979 |
| 172×172 | 14786 | 14798 |
| 174×174 | 14053 | 16223 |
| 176×176 | 14415 | 16561 |
| 178×178 | 13614 | 18070 |
| 180×180 | 16456 | 15944 |
| 182×182 | 15211 | 17913 |
| 184×184 | 15676 | 18180 |
| 186×186 | 16255 | 18341 |
| 188×188 | 18049 | 17295 |
| 190×190 | 16800 | 19300 |
| 192×192 | 17207 | 19657 |
| 194×194 | 16448 | 21188 |
| 196×196 | 18950 | 19466 |
| 198×198 | 18393 | 20811 |
| 200×200 | 19681 | 20319 |

| | | |
|---|---|---|
| 136×136 | 9047 | 9449 |
| 138×138 | 10846 | 8198 |
| 140×140 | 7665 | 11935 |
| 142×142 | 11553 | 8611 |
| 144×144 | 9788 | 10948 |
| 146×146 | 12169 | 9147 |
| 148×148 | 10202 | 11702 |
| 150×150 | 12966 | 9534 |
| 152×152 | 11009 | 12095 |
| 154×154 | 13636 | 10080 |
| 156×156 | 11227 | 13109 |
| 158×158 | 14221 | 10743 |
| 160×160 | 12265 | 13335 |
| 162×162 | 15087 | 11157 |
| 164×164 | 11998 | 14898 |
| 166×166 | 14479 | 13077 |
| 168×168 | 11718 | 16506 |
| 170×170 | 16706 | 12194 |
| 172×172 | 11488 | 18096 |
| 174×174 | 17449 | 12827 |
| 176×176 | 14575 | 16401 |
| 178×178 | 18335 | 13349 |
| 180×180 | 14149 | 18251 |
| 182×182 | 17363 | 15761 |
| 184×184 | 16460 | 17396 |
| 186×186 | 20005 | 14591 |
| 188×188 | 16333 | 19011 |
| 190×190 | 20524 | 15576 |
| 192×192 | 17187 | 19677 |
| 194×194 | 21867 | 15769 |
| 196×196 | 17896 | 20520 |
| 198×198 | 20459 | 18745 |
| 200×200 | 18938 | 21062 |

## Appendix C　プログラム

平衡二分探索木を用いて実装しようと考えたが,Python をあまり触ったことがなかったため一から実装をしないといけず, 面倒くさいなと思い C++ の std::set を用いて書くことにした. 実際のプログラムを以下に記す.

```
#include <bits/stdc++.h>
using namespace std;

constexpr int black = 1;
constexpr int white = -1;
constexpr int empty = 0;

vector<vector<int>> direction = {{1, 1}, {1, 0}, {1, -1}, {0, 1}, {0, -1}, {-1, 1},
    {-1, 0}, {-1, -1}};

vector<vector<int>> create_board(int board_size) {
    vector<vector<int>> board(board_size, vector<int>(board_size, 0));
    int center = board_size / 2;
    board[center - 1][center - 1] = white;
    board[center][center] = white;
    board[center - 1][center] = black;
    board[center][center - 1] = black;
    return board;
}

set<pair<int, int>> create_valid(int board_size) {
    int center = board_size / 2;
    set<pair<int, int>> valid;
    for (int i = -2; i < 1 + 1; i++) {
        if (i == -2 or i == 1) {
            for (int j = -2; j < 1 + 1; j++) {
                valid.insert(make_pair(center + i, center + j));
            }
        }
        else {
            for (int j = -2; j < 1 + 1; j += 3) {
                valid.insert(make_pair(center + i, center + j));
            }
        }
    }
    return valid;
}

bool valid_move(vector<vector<int>> &board, int now_turn, int board_size, vector<int> &sum,
```

```
 set<pair<int, int>> &valid) {
bool flag = false;
vector<vector<int>> dirnum;
for (auto a : valid) {
    int i = a.first, j = a.second;
    for (auto k : direction) {
        bool nice_direction = false;
        int now_i = i + k[0], now_j = j + k[1], numnum = 0;
        while (true) {
            if (!(0 <= now_i and now_i < board_size and 0 <= now_j and now_j <
                board_size)) break;
            else if (!(abs(board[now_i][now_j]))) break;

            if (board[now_i][now_j] + now_turn) {
                if (numnum) nice_direction = true;
                break;
            }
            else {
                numnum++;
                now_i += k[0];
                now_j += k[1];
            }
        }
        if (nice_direction) {
            flag = true;
            dirnum.push_back({k[0], k[1], numnum});
        }
    }
    if (flag) {
        sum[(-now_turn + 1) / 2]++;
        valid.erase((make_pair(i, j)));
        for (auto p : dirnum) {
            vector<int>k(2);
            k[0] = p[0];
            k[1] = p[1];
            int numnum = p[2];
            sum[(-now_turn + 1) / 2] += numnum;
            sum[(now_turn + 1) / 2] -= numnum;
            for (int l = 0; l < numnum + 1; l++) board[i + k[0] * l][j + k[1] * l] =
                now_turn;
        }
        for (auto k : direction) {
            if (0 <= i + k[0] and i + k[0] < board_size and 0 <= j + k[1] and j + k
                [1] < board_size) {
                if (!(abs(board[i + k[0]][j + k[1]]))) valid.insert(make_pair(i + k[0],
                     j + k[1]));
```

```
                }
            }
            break;
        }
    }
    return flag;
}

// latexにコピペできるように
void fingame(int board_size, vector<int> sum) {
  vector<string>s(3);
  s[0] = "\\cellcolor{blue!10}";
  s[1] = "\\cellcolor{red!10}";
  s[2] = "\\cellcolor{yellow!10}";
  string g;
  if (sum[0] > sum[1]) g = s[0];
  else if (sum[0] < sum[1]) g = s[1];
  else g = s[2];
  cout << g << board_size << "\\times" << board_size << " & ";
  cout << g << sum[0] << " & " << g << sum[1] << " ";
  cout << endl;
}

int main() {
    clock_t st = clock();
    int start = 6; int board_size = start;
    int limit = 200;
    while (board_size < limit + 2) {
        vector<vector<int>> board = create_board(board_size);
        set<pair<int, int>> valid = create_valid(board_size);
        int now_turn = black;
        vector<int> sum = {2, 2};
        while (true) {
            if (valid_move(board, now_turn, board_size, sum, valid)) now_turn *= -1;
            else if (valid_move(board, -now_turn, board_size, sum, valid));
            else break;
        }
        fingame(board_size, sum);
        board_size = board_size + 2;
    }
    clock_t ed = clock();
    cout << (double) (ed - st) / CLOCKS_PER_SEC << "s" << endl;
    return 0;
}
```