資訊傳播學系學士班B科技組

期末專題報告

黑白程式專題報告

Othello Program Report for Practical Project

指導老師：方文聘教授

作 者：鄭兆晉

中華民國一零九年六月

**目錄**

封面…………………………………………………………………… 0

目錄…………………………………………………………………… 1

問題說明……………………………………………………………… 2

演算法………………………………………………………………… 3

程式碼………………………………………………………………… 4

1.基本判斷………………………………………………………4

1.1findway………………………………………………………4

1.2in\_blank…………………………………………………… 5

1.3still\_in……………………………………………………6

2.棋盤改變………………………………………………………6

2.1change………………………………………………………6

3.棋盤計算………………………………………………………7

3.1counting……………………………………………………7

討論……………………………………………………………………8

附錄……………………………………………………………………9

**問題說明**

棋盤程式要如何運用引入值team t來去判斷當前棋盤的可放置之位置。

至於如何找尋到最佳步法，就必須運用到，這學期所學得各式邏輯與語法，好比各個棋子所對應當前局面的參數值，又或者如何用函數去取代冗長的數學算式，這些都是我們該去深究的問題。

|  |
| --- |
|  |
| 圖1.1:檔案轉換流程 |

**演算法**

由比賽判斷程式可得先後手(who)以及當前棋盤版面(A[8][8])，藉此得知當前比賽狀況，以當前棋盤版面(圖2.1)，由function find來找出合法位置(圖2.2) ，並將放置位置以point回傳。

|  |  |
| --- | --- |
| C:\Users\mogac\AppData\Local\Microsoft\Windows\INetCache\Content.Word\投影片3.jpg | 投影片4 |
| 圖2.1:棋盤版面 | 圖2.2:棋盤放置位置(代碼0處) |

**程式碼**

|  |  |
| --- | --- |
| C:\Users\mogac\AppData\Local\Microsoft\Windows\INetCache\Content.Word\簡報1.jpg |  |
| 圖3.1.1: findway流程 | 圖3.1.2:findway程式碼 |

Findway

運用迴圈掃過整面棋譜，每當遇到空格(blank)時，往該空格(blank)的八個方向做掃描，將其參數放入in\_blank(圖.3.2.2)來看是否為合法棋步。

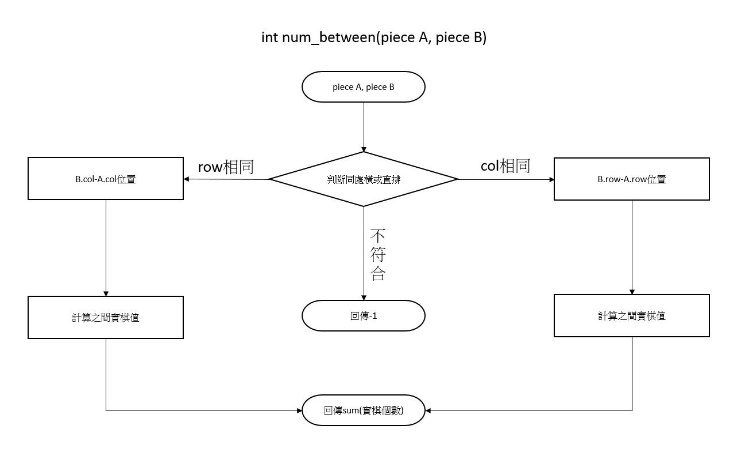
In\_blank

將findway(圖.3.1.2)的參數放入，查看該方向的質是否為敵方旗子，如果是繼續查詢直到只到自己棋子則成立，但找到空白則不成立。

|  |
| --- |
| C:\Users\mogac\AppData\Local\Microsoft\Windows\INetCache\Content.Word\簡報12.jpg |
| 圖3.2.1: in\_blank 流程圖 |
| C:\Users\mogac\AppData\Local\Microsoft\Windows\INetCache\Content.Word\in_blank.png |
| 圖3.2.2: in\_blank程式碼 |

Still\_in

查詢該位置是否超出棋盤。



|  |
| --- |
| C:\Users\mogac\AppData\Local\Microsoft\Windows\INetCache\Content.Word\still_in.png |
| 圖3.3.1:still\_in程式碼 |

Change

運用in\_blank(圖.3.2.2)查看是否為合理放置步法，若為合理放置步法將路徑上的敵方棋變為我方棋，直到遇到我方棋停止。

|  |
| --- |
|  |
| 圖3.4.1:change程式碼 |

Counting

掃過棋盤每當遇到自己sum++，每當遇到敵方sum--，最後將sum回傳。

|  |
| --- |
| C:\Users\mogac\AppData\Local\Microsoft\Windows\INetCache\Content.Word\counting.png |
| 圖3.5.1:counting程式碼 |

|  |
| --- |
| **討論**  **分析與證明程式之功能**  失敗一:盲目地尋找角格而無法將中心佔領  調整一:加入比較當前棋子差異的參數  失敗二:時常將自己逼入困境  調整二:加入棋盤最外圍(+)與內縮一圈(-)的參數  **缺陷還有與現有方法比較**  缺陷一:雖然把最外圈占滿內圈依然是敵方的最終棋子數量仍然輸  調整一與調整二加入後棋盤的比例逐漸接近平衡，時常以一兩棋子之差距獲勝。  **特色**  參數對照可將不同形式的對戰方式加以應對。 |

**附錄**

列表1.1:完整程式碼

|  |
| --- |
| #include "pch.h""  #include <iostream>  #include <windows.h>  #include <iomanip>  #include <time.h>  using namespace std;  struct team {  int who;  int A[8][8];  };  struct point {  int x;  int y;  };  struct way {  int times;  point ways[60];  way() {  times = 0;  }  };  const int blank = 0;  const int white = 1;  const int black = 2;  void setboard(struct team& t, struct team other) {  for (int i = 0; i < 8; i++)  for (int j = 0; j < 8; j++)  t.A[i][j] = other.A[i][j];  t.who = other.who;  }  void findway(team t, way\* w);  void change(point p, team\* t);  bool still\_in(int row, int col);  bool in\_blank(int row, int col, int x, int y, team t);  int counting(team m, team o);  extern"C" {  \_declspec(dllexport) struct point play(struct team my\_t) {  //檔案內  int tmp = 0;  int ti = 0;  int score[60] = { 0 };  team opp\_t;  way my\_w, opp\_w;  struct point p;  findway(my\_t, &my\_w);  for (int i = 0; i < my\_w.times; i++) {  p.x = my\_w.ways[i].x;  p.y = my\_w.ways[i].y;  setboard(opp\_t, my\_t);  change(p, &opp\_t);  if (((p.x == 0) && (p.y == 0)) ||  ((p.x == 0) && (p.y == 7)) ||  ((p.x == 7) && (p.y == 0)) ||  ((p.x == 7) && (p.y == 7))) {  score[i] += 1000;  }  else if (((p.x == 0) && (p.y == 1)) ||  ((p.x == 1) && (p.y == 0)) ||  ((p.x == 6) && (p.y == 0)) ||  ((p.x == 7) && (p.y == 1)) ||  ((p.x == 0) && (p.y == 6)) ||  ((p.x == 1) && (p.y == 7)) ||  ((p.x == 6) && (p.y == 7)) ||  ((p.x == 7) && (p.y == 6))) {  score[i] += 20;  }  else if (((p.x == 0) && (p.y == 1)) ||  ((p.x == 1) && (p.y == 1)) ||  ((p.x == 1) && (p.y == 6)) ||  ((p.x == 6) && (p.y == 1)) ||  ((p.x == 6) && (p.y == 6))) {  score[i] = 0;  }  else {  score[i] += 60;  }  if (p.x == 0 || p.y == 0 || p.x == 7 || p.y == 7) {  score[i] += 40;  }  else if (p.x == 1 || p.y == 1 || p.x == 6 || p.y == 6) {  score[i] += 0;  }  else {  score[i] += 20;  }  opp\_w.times = 0;  opp\_t.who = (3 - my\_t.who);  findway(opp\_t, &opp\_w);  score[i] += counting(my\_t, opp\_t) \* 65;  score[i] -= opp\_w.times \* 5;  for (int k = 0; k < opp\_w.times; k++) {  if (((opp\_w.ways[k].x == 0) && (opp\_w.ways[k].y == 0)) ||  ((opp\_w.ways[k].x == 0) && (opp\_w.ways[k].y == 7)) ||  ((opp\_w.ways[k].x == 7) && (opp\_w.ways[k].y == 0)) ||  ((opp\_w.ways[k].x == 7) && (opp\_w.ways[k].y == 7))) {  score[i] -= 100;  break;  }  }  }  for (int i = 0; i < my\_w.times; i++) {  if (score[i] >= tmp) {  tmp = score[i];  ti = i;  }  }  p.x = my\_w.ways[ti].x;  p.y = my\_w.ways[ti].y;  return p;  }  }  void findway(team t, way\* w) {  for (int i = 0; i < 8; i++) {  for (int j = 0; j < 8; j++) {  if (t.A[i][j] == blank) {  bool could = false;  for (int ii = -1; ii <= 1; ii++) {  for (int jj = -1; jj <= 1; jj++) {  if (in\_blank(i, j, ii, jj, t)) {  could = true;  }  }  }  if (could) {  w->ways[w->times].x = i;  w->ways[w->times].y = j;  w->times++;  }  }  }  }  }  bool still\_in(int row, int col) {  if (row >= 0 && row < 8 && col >= 0 && col < 8) {  return true;  }  return false;  }  bool in\_blank(int row, int col, int x, int y, team t) {  if (t.A[row + x][col + y] == (3 - t.who)) {  for (int max = 2; max < 8; max++) {  if (t.A[row + max \* x][col + max \* y] == blank &&  still\_in(row + max \* x, col + max \* y)) {  return false;  }  else if (t.A[row + max \* x][col + max \* y] == t.who &&  still\_in(row + max \* x, col + max \* y)) {  return true;  }  }  }  return false;  }  void change(point p, team\* t) {  for (int ii = -1; ii <= 1; ii++) {  for (int jj = -1; jj <= 1; jj++) {  if (in\_blank(p.x, p.y, ii, jj, \*t)) {  t->A[p.x][p.y] = t->who;  for (int max = 1; max < 8; max++) {  if (t->A[p.x + max \* ii][p.y + max \* jj] == (3 - t->who) &&  still\_in(p.x + max \* ii, p.y + max \* jj)) {  t->A[p.x + max \* ii][p.y + max \* jj] = t->who;  }  if (t->A[p.x + max \* ii][p.y + max \* jj] == t->who &&  still\_in(p.x + max \* ii, p.y + max \* jj)) {  break;  }  }  }  }  }  }  int counting(team m, team o) {  int sum = 0;  for (int i = 0; i < 8; i++) {  for (int j = 0; j < 8; j++) {  if (o.A[i][j] == m.who) {  sum++;  }  if (m.A[i][j] == m.who) {  sum--;  }  }  }  return sum;  } |