Algoritma Greedy dalam Artificial Intelligence Permainan Tic Tac Toe

Alif Bhaskoro 13514016
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13514016@std.stei.itb.ac.id

Abstract—Seiring dengan kemajuan teknologi yang semakin pesat, banyak permainan yang seharusnya dimainkan oleh 2 orang dibuat menjadi permainan 1 orang. Tentu perubahan ini tidak merubah cara memainkan permainan tersebut, melainkan merubah pemain lawan yang dulunya merupakan manusia menjadi artificial intelligence (inteligensi buatan). Dalam makalah ini akan dijelaskan mengenai implementasi algoritma greedy dalam artificial intelligence permainan tic tac toe.

Keywords—artificial intelligence, board game, greedy, tic tac toe

I. PENDAHULUAN

Pada zaman sekarang, handphone sudah menjadi hal yang lazim ditemui pada kehidupan sehari-hari. Handphone yang pada zaman dahulu hanya digunakan untuk menelpon seseorang atau mengirim sms, sekarang dapat digunakan sebagai sarana hiburan. Salah satu bentuk hiburan dari handphone adalah games / permainan. Banyak permainan-permainan yang bersifat board game di adopsi menjadi permainan handphone, salah satunya adalah tic tac toe.

Permainan tic tac toe haruslah dimainkan oleh 2 orang. Tidak mungkin kita bermain tic tac toe sendiri tanpa lawan. Namun jika telah diadopsi menjadi permainan HP, bagaimana mungkin permainan ini dapat dimainkan oleh 2 orang yang berbeda? Salah satu cara penyelesaian masalah ini adalah dengan menghubungkan 2 orang pemain dengan handphone berbeda melalui internet atau online. Tentu saja dengan cara ini, permainan board game yang tadinya gratis jadi memakan biaya yaitu biaya internet untuk menghubungkan kedua pemain. Belum lagi waktu tunggu yang dibutuhkan salah satu pemain untuk mencari lawan. Jika salah satu pemain ingin bermain tic tac toe pada pukul 3 dini hari, tentu saja waktu tunggu yang dibutuhkan untuk mencari lawan pada jam tersebut akan lama. Alangkah baiknya jika permainan tic tac toe ini dapat dimainkan secara gratis dan tidak memiliki waktu tunggu setelah diadopsi menjadi game handphone.

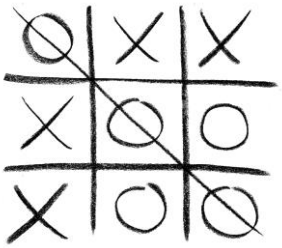
Apakah itu mungkin ? Ya, dengan menggunakan artificial intelligence atau intelligensi buatan sebagai lawan main manusia. Artificial intelligence atau AI adalah sebuah kecerdasan buatan yang dapat melakukan pekerjaan seperti yang dilakukan oleh manusia. Untuk kasus kali ini adalah bermain tic tac toe melawan user. Tapi tentu saja user tidak ingin AI lawannya hanya sekedar bermain saja, Tentu user ingin mendapatkan perlawanan sengit oleh AI layaknya saat user melawan manusia lain.

Algoritma greedy dapat digunakan untuk membuat AI masalah ini. Dengan mengimplementasikan algoritma greedy dalam pengambilan keputusan / langkah oleh AI, maka AI ini dapat memberikan perlawanan sengit kepada user selayaknya manusia. Dalam makalah ini akan dijelaskan secara lengkap bagaimana algoritma greedy dapat diimplementasikan dalam membuat AI untuk memecahkan masalah ini.

II. DASAR TEORI

2.1 TIC TAC TOE

Tic tac toe adalah permainan papan yang dimainkan oleh 2 orang pemain, X dan O. Kedua pemain ini bergantian menandai papan 3 x 3 sesuai dengan tanda mereka (X / O). Permainan akan berakhir jika salah satu dari pemain telah berhasil membuat garis baik itu horizontal, vertikal, maupun diagonal.



Gambar 1. Contoh permainan tic tac toe

Tic tac toe pertama kali dimainkan di Kerajaan Roma. Dulu permainan ini disebut Terni Lapilli. Jika pada tic tac toe sekarang setiap pemain memiliki jumlah piece yang tidak terbatas, pada Terni Lapilli setiap pemain hanya memiliki 3 buah piece yang nantinya akan dipindah-pindah ke kotak kosong agar permainan tetap berlanjut. Papan permainan Terni Lapilli yang dibuat dengan kapur ini ditemukan di seluruh Roma.

Ada 8 jenis langkah pada permainan tic tac toe ini yaitu :

1. Win, jika pemain memiliki 2 piece pada satu baris maka player dapat menaruh piece ke 3 untuk menang

2. Block, jika pemain lawan memiliki 2 piece pada satu baris maka pemain harus menaruh piece nya untuk menghalangi lawannya.
3. Fork, langkah dimana player memiliki 2 buah baris yang sudah memiliki 2 buah piece
4. Block fork lawan, langkah menghalangi jalan lawan agar lawan tidak membentuk fork yang akan menyebabkan kekalahan
5. Center, menaruh piece pada posisi tengah. Biasanya dilakukan pada saat awal permainan
6. Opposite Corner, jika piece lawan berada di suatu pojok papan maka pemain memainkan piecenya di pojok yang berlawanan
7. Empty Corner, memainkan piece pada pojok papan yang kosong
8. Empty Side, memainkan piece pada kotak tengah salah satu sisi papan permainan

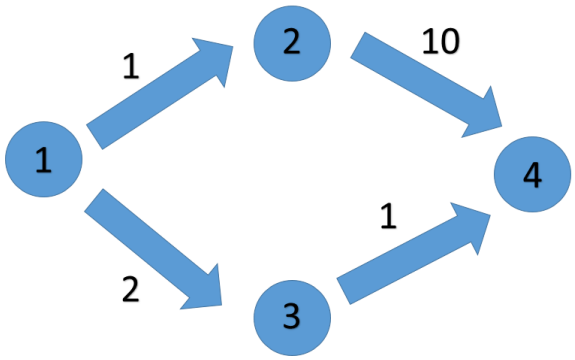
Setiap pemain yang memiliki langkah pertama memiliki 3 opsi langkah, yaitu center, empty corner atau empty side. Walau nampaknya pemain memiliki 9 buah opsi yang berbeda karena papan terdiri dari 9 buah kotak kosong, namun jika papan diputar maka dapat dilihat bahwa seluruh corner memiliki nilai yang sama secara strategis untuk langkah pertama. Begitu juga dengan setiap sisi kosong. Langkah paling baik untuk pemain pertama adalah corner karena akan membatasi pilihan pemain lawan.

Pemain yang memiliki langkah ke-2 harus merespon secara tepat untuk mencegah terjadinya forced win oleh pemain pertama. Jika pemain pertama menaruh di posisi corner, maka pemain kedua harus merespon dengan menaruh di center. Jika pemain pertama menaruh di posisi edge, maka pemain kedua dapat merespon dengan menaruh piecenya di center, corner yang berada disebelah edge yang telah diisi atau opposite edge.

Walau permainan tic tac toe ini terlihat simple namun jika dihitung dengan teori matematika, terdapat 19,683 layout yang mungkin dibuat dari pemainan ini (termasuk kotak kosong) dan 362,880 kemungkinan games (jika urutan pemasangan piece diperhatikan). Namun tidak semuanya dapat dicapai karena biasanya sudah permainan sudah dimenangkan salah satu pemain dan constraint yaitu jumlah X harus sama dengan atau lebih banyak 1 dari jumlah O.

2.2 Algoritma Greedy

Algoritma greedy membentuk solusi secara step by step dengan prinsip take what you can get now. Pada setiap step terdapat banyak pilihan yang tersedia. Algoritma greedy akan mengambil langkah optimum pada tiap step dengan harapan dengan mengambil setiap langkah optimum pada setiap step akan menghasilkan langkah optimum global nantinya. Algoritma greedy terkadang tidak berhasil menghasilkan solusi optimum. Karena banyak kasus dimana solusi optimum suatu langkah tidak menghasilkan solusi optimum global.



Gambar 2. Contoh kasus algoritma greedy

Pada kasus yang digambarkan pada gambar 2 tersebut, dapat dilihat bahwa algoritma greedy akan gagal jika diterapkan. Misalkan permasalahan yang ada adalah mencari jalan terpedek dari 1 ke 4. Pada langkah pertama, kita akan dihadapkan dengan 2 pilihan yaitu jalan ke 2 yang berbobot 1 atau jalan ke 3 yang berbobot 2. Jika menggunakan algoritma greedy maka kita akan memilih jalan ke 2 yang memiliki bobot lebih kecil. Setelah sampai di titik 2, satu-satunya jalan adalah jalan ke 4 yang memiliki nilai bobot 10. Sehingga total bobot jalan dari 1 ke 4 dengan jalur 1-2-4 adalah 11. Padahal jika kita memilih jalan ke 3 pada langkah pertama maka jalur yang akan terpilih adalah 1-3-4 dengan bobot 3.

Algoritma greedy memiliki 5 buah elemen umum yaitu :

1. Himpunan Kandidat, C

   Himpunan kandidat berisi elemen-elemen yang dapat membentuk solusi.

2. Himpunan Solusi. S

   Himpunan yang berisi elemen dari himpunan kandidat yang telah dipilih sebagai solusi dari persoalan.

3. Fungsi seleksi

   Fungsi yang pada setiap langkah akan memilih elemen dari himpunan kandidat yang paling memungkinkan mencapai solusi optimal.

4. Fungsi kelayakan (feasible)

   Fungsi yang memeriksa apakah elemen kandidat yang telah dipilih jika dimasukkan ke dalam himpunan solusi tidak melanggar constrant yang ada.

5. Fungsi objektif

   Fungsi untuk meminimumkan atau memaksimalkan solusi yang telah dipilih.

Walau algoritma greedy terkadang tidak memberikan solusi optimal, namum untuk banyak kasus algoritma greedy berhasil memberikan solusi optimal dari suatu persoalan. Algoritma greedy banyak dipakai untuk menyelesaikan masalah penukaran uang, minimasi waktu dalam sistem, knapsack, TSP, dll.

III. IMPLEMENTASI

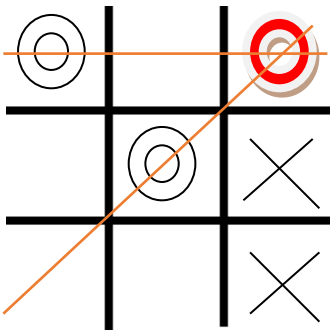
3.1 Algoritma Greedy dalam Turn-Based Game

Tactical turn based game AI dapat dibuat dengan pendekatan algoritma greedy. Hal ini disebabkan karena untuk setiap turn, AI harus memilih langkah paling optimal yang dapat dia lakukan. Karena tactical turn based game ini bersifat dinamis (tidak ada yang bisa memprediksi gerakan pemain lawan) maka pemilihan langkah optimal saat itu akan menghasilkan solusi optimal juga.

Tic tac toe adalah salah satu tactical turn based game, sehingga pembuatan AI tic tac toe dapat menggunakan implementasi algoritma greedy. Misalnya pada suatu turn, AI telah memiliki 2 piece yang berada pada 1 garis dan sekarang ada giliran AI. Jika dihitung nilainya maka dapat dilihat jika menaruh piece berikutnya di garis yang telah memiliki 2 piece adalah langkah tercepat untuk menang. Hal ini mirip sekali dengan prinsip algoritma greedy. Contoh lainnya adalah ketika AI hanya memiliki 1 piece pada suatu garis dan tidak memiliki piece lain di garis lainnya. Alih-alih menaruh piece pada garis yang belum memiliki piece, akan lebih cepat memenangkan permainan jika menaruh piece pada garis yang sudah memiliki piece sebelumnya. Sehingga garis itu akan memiliki 2 piece setelah giliran AI dan tinggal membutuhkan 1 buah piece lagi agar memenangkan permainan.

3.2 Sistem Pemilihan Langkah AI

Pemilihan langkah AI dapat diimplementasikan dengan algoritma greedy. Setiap langkah yang dapat dilakukan oleh AI akan diberi nilai. Semakin tinggi nilainya berarti langkah tersebut akan lebih dekat ke kemenangan.



Gambar 3. Contoh tic tac toe

Pada kasus seperti pada gambar 3, langkah yang paling baik bagi AI (O) adalah menaruh piece nya pada baris ke-1 kolom ke-3. Hal ini karena selain posisi itu mengagalkan X untuk mencapai kemenangan (X sudah memiliki 2 buah piece pada kolom ke 3 dan hanya membutuhkan 1 buah piece lagi). Tapi posisi ini juga akan menciptakan fork yang nantinya akan menghasilkan kemenangan. Posisi itu akan menghasilkan 2 garis yang pada masing-masing garis ada 2 buah piece. Garis pertama adalah baris pertama dan garis kedua adalah garis diagonal naik (digambarkan dengan garis berwarna merah). Karena itu posisi 1-3 (baris ke-1 kolom ke-3) adalah langkah terbaik atau optimal pada saat ini dan akan memiliki nilai paling besar.

Algoritma greedy memiliki 5 buah elemen, berikut juga AI tic tac toe ini. 5 buah elemen tersebut adalah

1. Himpunan Kandidat

   Himpunan kandidat berisi kotak-kotak yang dapat dipilih oleh AI. Pada game tic tac toe ini, berarti adalah kotak-kotak yang belum diisi oleh pemain maupun AI itu sendiri (kotak kosong).

2. Himpunan Solusi

   Himpunan solusi berisi kotak-kotak yang sudah diisi baik oleh pemain maupun oleh AI.

3. Fungsi Seleksi

   Memilih langkah yang memiliki nilai paling tinggi / paling menguntungkan.

4. Fungsi Kelayakan

   Jumlah langkah yang dipilih oleh AI hanya boleh 1 langkah per giliran, tidak boleh lebih dan tidak boleh kurang.

5. Fungsi Objektif

   Langkah yang diambil AI jumlahnya minimum

Pada program yang saya kembangkan, fungsi seleksi akan memilih posisi yang memiliki nilai paling besar dengan algoritma :

```
procedure max(input hasil : int[3][3],output x : int,output  y : int)
    //Deklarasi
    m,i,j : int
    //Algoritma
    x,y,i,j <- 0
    m = hasil[x][y];
    while(i<3) do
        while (j<3) do
            if(hasil[i][j]>m)
                x <- i
                y <- j
                m <- hasil[i][j]
            j <- j+1
        endwhile
        i <- i+1
    endwhile
```

Gambar 4. Prosedur max

Prosedur max akan mencari nilai terbesar dari matriks of integer bernama hasil. Matriks hasil berisi skor dari masing-masing posisi pada board. Semakin tinggi skornya maka posisi itu semakin menguntungkan untuk diisi oleh AI. Misalnya hasil[0][1] bernilai 10 dan hasil[1][1] bernilai 9. Maka AI akan menaruh piecenya di baris pertama kolom kedua (board[0][1]) karena dilihat dari skor yang dimilikinya, disimpulkan posisi itu lebih menguntungkan. Jika ada 2 buah posisi yang memiliki skor yang sama, maka program akan memilih posisi yang pertama kali ditemukan. Sehingga program tidak melanggar constraint (1 langkah per giliran) dan memenuhi fungsi

kelayakan. Untuk perhitungan skor setiap posisi digunakan algoritma sebagai berikut :

```
function isBaris(input board : int[3][3],input i : int)
    //Deklarasi
    hasil,j : int
    //Algoritma
    hasil <- 0
    j <- 0
    while(j<3)
        if(board[i][j]==2)
            hasil <- hasil + 1
        else if(board[i][j]==1)
            hasil <- hasil -1
        j <- j + 1
    endwhile
    if(hasil==3)
        hasil <- 3
    else if(hasil<-1)
        hasil <- 10;
    else if(hasil>=2)
        hasil <- 999;
    return hasil;
```

Gambar 5. Fungsi isBaris

Fungsi isBaris adalah fungsi yang menerima input berupa matriks of integer berukuran 3x3 yang bernama board dan sebuah integer i. Board merupakan papan permainan tic tac toe dalam program. Jika board[i][j] bernilai 0 berarti posisi tersebut (baris ke i-1 dan kolom j-1) masih kosong. Jika bernilai 1 berarti berisi piece milik pemain dan jika bernilai 2 berarti berisi piece milik AI. Integer i merupakan nomor baris yang hendak dihitung skornya.

Fungsi isBaris akan mereturn sebuah integer yaitu hasil. Hasil merupakan skor yang menandakan seberapa menguntungkan posisi tersebut jika di tempati oleh AI. Pertama fungsi akan mengecek berapa banyak piece yang sudah ada di baris i. Untuk setiap piece AI maka hasil akan di increment dengan 1 dan untuk setiap piece pemain maka hasil akan di decrement dengan 1. Jika setelah dari perhitungan ini, hasil bernilai -2 (sudah ada 2 piece pemain pada baris tersebut) maka hasil akan dirubah menjadi 10. Hal ini disebabkan posisi itu memiliki prioritas cukup tinggi karena jika tidak ditempati oleh AI maka pemain dapat memenangkan permainan. Jika hasil bernilai 2 (sudah ada 2 piece AI pada baris tersebut), maka hasil akan dirubah menjadi 999. Hal ini disebabkan karena posisi ini memiliki prioritas tertinggi untuk ditempati (AI akan memenangkan permainan jika menempati posisi ini). Jika hasil bernilai 3, maka biarkan. Hal ini akan berguna untuk menghentikan loop pada program utama karena menandakan permainan sudah usai.

Fungsi isBaris adalah fungsi untuk menghitung skor dari suatu baris. Selain fungsi isBaris, ada beberapa fungsi lain untuk menghitung skor yaitu isKolom (menghitung skor suatu kolom tertentu), isDiagonalKiri (menghitung skor diagonal yang dimulai dari pojok kiri atas sampai pojok kanan bawah) dan isDiagonalKanan (menghitung skor diagonal yang dimulai dari pojok kiri bawah sampai pojok kanan atas).

Pada program yang saya kembangkan, fungsi objektifnya adalah memilih langkah paling minimum untuk memenangkan permainan. Karena itu pada program utamanya, AI akan memilih posisi yang memiliki skor paling besar dengan algoritma :

```
for(int i=0;i<3;i++){
    int result = isBaris(board,i);
    for(int j=0;j<3;j++){
        if(board[i][j]==0){
            hasil[i][j]+=result;
        }
    }
}
for(int j=0;j<3;j++){
    int result = isKolom(board,j);
    for(int i=0;i<3;i++){
        if(board[i][j]==0){
            hasil[i][j]+=result;
        }
    }
}
int n=0;
int reskiri = isDiagonalKiri(board);
while(n<3){
    if(board[n][n]==0){
        hasil[n][n]+=reskiri;
    }
    n++;
}
int reskanan = isDiagonalKanan(board);
int k = 2;
n=2;
while(n>-1){
    if(board[n][k-n]==0){
        hasil[n][k-n]+=reskanan;
    }
    n--;
}
int x,y;
max(hasil,x,y);
board[x][y]=2;
```

Gambar 6. Program utama

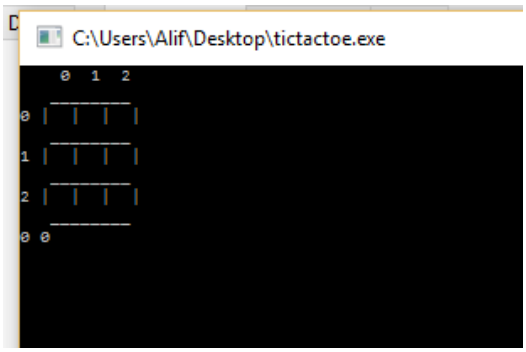
Gambar diatas merupakan gambar salah satu potongan program utama dari AI tic tac toe ini. Terdapat 4 loop pada program ini. Keempat loop itu memanggil fungsi isBaris, isKolom, isDiagonalKiri, dan isDiagonalKanan. Kemudian menambahkan hasil dari keempat fungsi tersebut ke matriks hasil sesuai posisinya masing-masing. Kemudian program akan memanggil prosedur max. Prosedur tersebut akan menghasilkan 2 buah output x & y. Kemudian board pada posisi tersebut akan diisi dengan piece AI.

Program utama ini mengimplementasikan fungsi objektif dari algoritma greedy. Karena fungsi objektif untuk AI tic tac toe ini adalah jumlah langkah seminimal mungkin, maka pengisian nilai matriks hasil menggunakan operator +=. Sehingga posisi yang nantinya akan dipilih AI adalah posisi

yang paling menguntungkan dilihat dari segi baris, kolom, maupun diagonal.

3.3 Implementasi pada program

Untuk mengetes AI yang saya kembangkan ini, saya membuat program permainan tic tac toe itu sendiri. Program ini mengimplementasikan permainan tic tac toe 3x3 dengan 1 orang pemain dan 1 AI. AI akan menggunakan simbol O dan pemain akan menggunakan simbol X. Giliran selalu dimulai dari pemain. Program akan menerima input berupa 2 buah integer yang merupakan posisi simbol X (piece pemain). Program akan mengupdate board bedasarkan masukan dan kemudian AI akan bergerak.

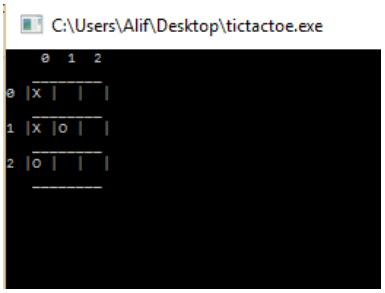


Gambar 7. Contoh eksekusi program

Gambar 8. Contoh eksekusi program

Pada gambar 7, program menerima input berupa 0 dan 0. Sehingga program akan menaruh piece pemain (X) pada posisi 0,0. Kemudian AI akan merespon dengan menaruh piecenya pada posisi 1,1 karena dinilai paling menguntungkan (gambar 8).

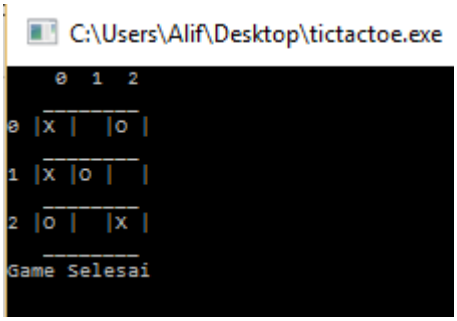
Gambar 9. Contoh eksekusi program



Gambar 10. Contoh eksekusi program

Kemudian jika pemain memilih posisi 1,0 untuk langkah berikutnya. AI akan merespon dengan menaruh piece nya pada posisi 2,0. Karena selain untuk menghalangi pemain dari kemenangan, posisi ini juga menguntungkan untuk AI karena sekarang AI memiliki 2 piece yang berada pada 1 garis (garis diagonal kanan).

Gambar 9. Contoh eksekusi program



Gambar 10. Contoh eksekusi program

Selanjutnya jika kita memilih 2,2 maka AI akan merespon dengan menaruh piecenya pada posisi 0,2. AI akan memilih langkah ini karena langkah ini akan memberikan kemenangan pada AI. Kemudian program akan berhenti dan mengeluarkan pesan “Game Selesai”.

IV. ANALISIS

Pada program yang saya buat ini, AI telah merespon dengan baik langkah-langkah dari pemain. Namun karena program ini hanya mengimplementasikan tic tac toe 3x3, maka gerakan yang dapat dilakukan baik oleh AI maupun pemain sangatlah terbatas. Pola serangan yang mungkin dilakukan pemain sangatlah terbatas juga. Sehingga AI masih dapat bekerja dengan baik.

Tapi jika permainan ini diperluas seperti menjadi tic tac toe 5x5 maka AI belum dapat bekerja dengan maksimal. Hal ini disebabkan karena pada tic tac toe yang lebih besar, pola serangan lebih bervariasi dan lebih memungkinkan fork dibanding 3x3. Sehingga AI harus dikembangkan lebih lanjut agar dapat memahami pattern-pattern serangan yang umum dilakukan oleh pemain. Dengan begitu AI akan bekerja lebih maksimal.

V. KESIMPULAN

Pengimplementasian AI dalam turn based board game seperti tic tac toe dapat dilakukan dengan pendekatan algoritma greedy. Hal ini disebabkan karena algoritma greedy memilih langkah paling optimal. Namum AI yang dihasilkan oleh pendekatan ini sangatlah sederhana. Untuk digunakan pada game yang lebih besar, AI ini haruslah dikembangkan terlebih dahulu dengan mengenalkannya terhadap pattern-pattern serangan umum.

VI. ACKNOWLEDGEMENT

Terima kasih kepada Tuhan YME karena tanpa berkat dari-Nya penulis tidak dapat menyelesaikan makalah ini. Terima kasih juga kepada Ir Rinaldi Munir M.T., dan Dr. Nur Ulfa Maulidevi, S.T., M.Sc karena tanpa ilmu dari beliau, makalah ini tidak dapat dirampungkan. Terima kasih juga kepada semua pihak yang terkait dalam pembuatan makalah ini.

REFERENCES

[1] http://www.mathrec.org/old/2002jan/solutions.html
[2] Kevin Crowley, Robert S. Siegler (1993). "Flexible Strategy Use in Young Children's Tic-Tac-Toe". Cognitive Science 17 (4): 531–561. doi:10.1016/0364-0213(93)90003-Q.
[3] Slide kuliah Strategi Algoritma 2015 - 2016 dari http://informatika.stei.itb.ac.id/~rinaldi.munir/

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 8 Mei 2016
Alif Bhaskoro
13514016

LAMPIRAN

```cpp
#include <iostream>
#include <stdlib.h>

using namespace std;

void printBoard(int board[3][3]){
    cout<<"    0  1  2" <<endl;
    for(int i=0;i<3;i++){
        cout << "   _______" << endl;
        cout << i<< " |";
        for(int j=0;j<3;j++){
            if(board[i][j]==0){
                cout << " " << " |";
            }
            else if(board[i][j]==1){
                cout << "X" << " |";
            }
            else{
                cout << "O" << " |";
            }

        }
        cout << endl;
    }
    cout << "   _______" << endl;
}

int isBaris(int board[3][3],int i){
    int hasil=0;
    for(int j=0;j<3;j++){
        //1 = X  2 = O bot = 0
        if(board[i][j]==2){
            hasil++;
        }
        else if(board[i][j]==1){
            hasil--;
        }
    }
    if(hasil==3){
        hasil=3;
    }
    else if(hasil<-1){
        hasil=10;
    }
    else if(hasil>=2){
        hasil=999;
    }
    return hasil;
}

int isKolom(int board[3][3],int j){
    int hasil=0;
    for(int i=0;i<3;i++){
        //1 = X  2 = O bot = 0
        if(board[i][j]==2){
            hasil++;
        }
        else if(board[i][j]==1){
            hasil--;
        }
    }
    if(hasil==3){
        hasil=3;
    }
    else if(hasil<-1){
        hasil=10;
    }
    else if(hasil>=2){
        hasil=999;
    }
    return hasil;
}

int isDiagonalKiri(int board[3][3]){
    int hasil=0;
    int n=0;
    while(n<3){
        if(board[n][n]==2){
            hasil++;
        }
        else if(board[n][n]==1){
            hasil--;
        }
        n++;
    }
    if(hasil==3){
        hasil=3;
    }
    else if(hasil<-1){
        hasil=10;
    }
    else if(hasil>=2){
        hasil=999;
    }
    else if(hasil==-1){
        hasil = 0;
    }
    return hasil;
}

```

```
int isDiagonalKanan(int board[3][3]){
    int hasil=0;
    int m=2;
    int n=2;
    while(n>-1){
        if(board[n][m-n]==2){
            hasil++;
        }
        else if(board[n][m-n]==1){
            hasil--;
        }
        n--;
    }
    if(hasil==3){
        hasil=3;
    }
    else if(hasil<-1){
        hasil=10;
    }
    else if(hasil>=2){
        hasil=999;
    }
    else if(hasil==-1){
        hasil = 0;
    }
    return hasil;
}
void max(int hasil[3][3],int & x,int & y){
    x=0;
    y=0;
    int m;
    m = hasil[x][y];
    for(int i=0;i<3;i++){
        for(int j=0;j<3;j++){
            if(hasil[i][j]>m){
                x=i;
                y=j;
                m=hasil[i][j];
            }
        }
    }
}

```

```
int main(){
    int board[3][3];
    for(int i=0;i<3;i++){
        for(int j=0;j<3;j++){
            board[i][j] = 0;
        }
    }
    printBoard(board);
    bool value = false;
    while(!value){
        int x,y;
        cin >> x >> y;
        if(board[x][y]==0){
            board[x][y]=1;
            int hasil[3][3];
            for(int i=0;i<3;i++){
                for(int j=0;j<3;j++){
                    if(board[i][j]==0){
                        hasil[i][j]=0;
                    }
                    else{
                        hasil[i][j] = -999;
                    }
                }
            }
            for(int i=0;i<3;i++){
                int result = isBaris(board,i);
                for(int j=0;j<3;j++){
                    if(board[i][j]==0){
                        hasil[i][j]+=result;
                    }
                }
            }
            for(int j=0;j<3;j++){
                int result = isKolom(board,j);
                for(int i=0;i<3;i++){
                    if(board[i][j]==0){
                        hasil[i][j]+=result;
                    }
                }
            }
            int n=0;
            int reskiri = isDiagonalKiri(board);
            while(n<3){
                if(board[n][n]==0){
                    hasil[n][n]+=reskiri;
                }
                n++;
            }
            int reskanan = isDiagonalKanan(board);
            int k = 2;
            n=2;
            while(n>-1){
                if(board[n][k-n]==0){
                    hasil[n][k-n]+=reskanan;
                }
                n--;
            }
            int x,y;
            max(hasil,x,y);
            board[x][y]=2;
```