Universidad Autónoma Metropolitana – Unidad Azcapotzalco

División de Ciencias Básicas e Ingeniería

Ingeniería en Computación

Código fuente de la aplicación

ESTRATEGIA DE TEORÍA DE JUEGOS PARA EL JUEGO DEL GATO

Jonathan Jair Lira Rodríguez

207203766

Juan Carlos Chávez Flores

206304501

Trimestre Lectivo: 12-P

Julio 2012

Dr. Pedro Lara Velázquez

Número económico: 31213

```
#include<iostream>
#include<fstream>

void Permutaciones(int *, int l=0);
void Permutaciones_Main();
void Rangos_Gato(int tiro[], int cont, int &acum, int juga, float
casi_juga_prob[]);
float Lectura_Jug_Gato(int inic, int fin, int juga);

using namespace std;
int tiradas[362880][2];
int posi=0, cont_auxi;
float casi_juga_prob_1[9], casi_juga_prob_2[9];
ofstream Salida;

int main()
{
 int x, x_band=0, y, sele, jueg_fin=0, juga=-1, cont=0, acum=0, acum_auxi=0,
tipo_jueg, prim_pc=0, tiro_pc;
 int casi[9]={0,0,0,0,0,0,0,0,0}, tiro[9];
 float casi_juga_prob[9]={0,0,0,0,0,0,0,0,0};
 char juga_auxi;
 char casi_juga[10]={"---------"};
 int tiro_pc_auxi_1, tiro_pc_auxi_2;
 float prob_1, prob_2;

 cout<<"Tipo de Juego:\n1)Usuario vs Máquina\n2)Usuario vs Usuario"<<endl;
 cin>>tipo_jueg;

 if(tipo_jueg==1)
   {
      cout<<"\nSe seleccionó Usuario vs Máquina"<<endl;
      cout<<"¿Quién tira primero?\n1)Máquina\n2)Usuario"<<endl;
    cin>>prim_pc;

    if(prim_pc==1)
      cout<<"\nTira primero la Máquina"<<endl;
    else
      {
       cout<<"\nTira primero el Usuario"<<endl;
       prim_pc=2;
      }
   }
 else
   {
    cout<<"\nSe seleccionó Usuario vs Usuario"<<endl;
    tipo_jueg=2;
   }

 Permutaciones_Main();

 cout<<" 1 | 2 | 3\n 4 | 5 | 6\n 7 | 8 | 9"<<endl<<endl;

 for(x=0;x<9 && jueg_fin==0;x++)
   {
```

```
juga=juga*(-1);

if(x==x_band)
  {
   for(y=0;y<9;y++)
     {
      if(casi[y]==0)
         {
          tiro[cont]=y+1;
          Rangos_Gato(tiro, cont, acum, juga, casi_juga_prob);
         }

        acum=acum_auxi;
     }

   if(tipo_jueg==1 && ((prim_pc==1 && juga==1) || (prim_pc==2 && juga==-1)))
     {
      tiro_pc_auxi_1=-1;
      tiro_pc_auxi_2=-1;

      if(prim_pc==1)
        {
         for(y=0;y<9;y++)
          if(casi_juga_prob_1[y]!=666)
          {
             if(tiro_pc_auxi_1==-1)
               {
                tiro_pc_auxi_1=y;
                prob_1=casi_juga_prob_1[y];
               }
             else
               if(casi_juga_prob_1[tiro_pc_auxi_1]<=casi_juga_prob_1[y])
                  {
                   tiro_pc_auxi_1=y;
                   prob_1=casi_juga_prob_1[y];
                  }
             }
         for(y=0;y<9;y++)
           if(casi_juga_prob_2[y]!=666)
           {
          if(tiro_pc_auxi_2==-1)
               {
                tiro_pc_auxi_2=y;
                 prob_2=casi_juga_prob_2[y];
               }
             else
               if(casi_juga_prob_2[tiro_pc_auxi_2]>=casi_juga_prob_2[y])
                 {
                  tiro_pc_auxi_2=y;
                  prob_2=casi_juga_prob_2[y];
                 }
           }
        }
      else
        {
```

```
            for(y=0;y<9;y++)
              if(casi_juga_prob_2[y]!=666)
              {
               if(tiro_pc_auxi_1==-1)
                  {
                   tiro_pc_auxi_1=y;
                   prob_1=casi_juga_prob_2[y];
                  }
                else
                  if(casi_juga_prob_2[tiro_pc_auxi_1]<=casi_juga_prob_2[y])
                    {
                     tiro_pc_auxi_1=y;
                     prob_1=casi_juga_prob_2[y];
                    }
              }
            for(y=0;y<9;y++)
             if(casi_juga_prob_1[y]!=666)
             {
              if(tiro_pc_auxi_2==-1)
                  {
                   tiro_pc_auxi_2=y;
                   prob_2=casi_juga_prob_1[y];
                  }
                else
                  if(casi_juga_prob_1[tiro_pc_auxi_2]>=casi_juga_prob_1[y])
                    {
                     tiro_pc_auxi_2=y;
                     prob_2=casi_juga_prob_1[y];
                    }
              }
            }

            if(prob_1<prob_2)
              tiro_pc=tiro_pc_auxi_2;
            else
              tiro_pc=tiro_pc_auxi_1;
          }
/*
      ///Quitar comentarios para imprimir en pantalla tablero de probabilidades
      if(casi_juga_prob[0]==666)
        cout<<"  -   | ";
      else
        if(casi_juga_prob[0]==0)
          cout<<"  0   | ";
        else
          {
           cout.precision(4);
           cout<<showpoint<<casi_juga_prob[0]<<" | ";
          }

      if(casi_juga_prob[1]==666)
        cout<<"  -   | ";
      else
        if(casi_juga_prob[1]==0)
          cout<<"  0   | ";
```

```
  else
    {
     cout.precision(4);
     cout<<showpoint<<casi_juga_prob[1]<<" | ";
    }

if(casi_juga_prob[2]==666)
   cout<<"  -"<<endl;
else
  if(casi_juga_prob[2]==0)
    cout<<"  0"<<endl;
  else
    {
     cout.precision(4);
     cout<<showpoint<<casi_juga_prob[2]<<endl;
    }

if(casi_juga_prob[3]==666)
  cout<<"  -   | ";
else
  if(casi_juga_prob[3]==0)
    cout<<"  0   | ";
  else
    {
     cout.precision(4);
     cout<<showpoint<<casi_juga_prob[3]<<" | ";
    }

if(casi_juga_prob[4]==666)
  cout<<"  -   | ";
else
  if(casi_juga_prob[4]==0)
    cout<<"  0   | ";
  else
    {
     cout.precision(4);
     cout<<showpoint<<casi_juga_prob[4]<<" | ";
    }

if(casi_juga_prob[5]==666)
   cout<<"  -"<<endl;
else
  if(casi_juga_prob[5]==0)
    cout<<"  0"<<endl;
  else
    {
     cout.precision(4);
     cout<<showpoint<<casi_juga_prob[5]<<endl;
    }

if(casi_juga_prob[6]==666)
  cout<<"  -   | ";
else
  if(casi_juga_prob[6]==0)
    cout<<"  0   | ";
```

```
          else
            {
             cout.precision(4);
             cout<<showpoint<<casi_juga_prob[6]<<" | ";
             }

        if(casi_juga_prob[7]==666)
          cout<<"  -   | ";
        else
          if(casi_juga_prob[7]==0)
            cout<<"  0   | ";
          else
            {
             cout.precision(4);
             cout<<showpoint<<casi_juga_prob[7]<<" | ";
            }

        if(casi_juga_prob[8]==666)
           cout<<"  -"<<endl<<endl;
        else
          if(casi_juga_prob[8]==0)
            cout<<"  0"<<endl<<endl;
          else
            {
             cout.precision(4);
             cout<<showpoint<<casi_juga_prob[8]<<endl<<endl;
            }
*/
       x_band++;
      }

    if(tipo_jueg==2)
      {
       cout<<"Selecciona la casilla para tirar (1-9): ";
       cin>>sele;
       cout<<endl;
      }
    else
      if((prim_pc==1 && juga==1) || (prim_pc==2 && juga==-1))
        {
         cout<<"Selecciona la casilla para tirar (1-9):
"<<tiro_pc+1<<endl<<endl;
         sele=tiro_pc+1;
        }
      else
        {
         cout<<"Selecciona la casilla para tirar (1-9): ";
         cin>>sele;
         cout<<endl;
        }

    if(casi[sele-1]==0 && sele<10)
      {
       if(juga==1)
         juga_auxi='X';
```

```
        else
          juga_auxi='O';

        tiro[cont]=sele;
        Rangos_Gato(tiro, cont, acum, 0, casi_juga_prob);//El cero es para que no
calcule las probabilidades
        cont++;//solo afecta a la variable acum
        acum_auxi=acum;

        casi[sele-1]=juga;
        casi_juga[sele-1]=juga_auxi;
        casi_juga_prob[sele-1]=666;
        casi_juga_prob_1[sele-1]=666;
        casi_juga_prob_2[sele-1]=666;
        cout<<" "<<casi_juga[0]<<" | "<<casi_juga[1]<<" | "<<casi_juga[2]<<"\n "
                <<casi_juga[3]<<" | "<<casi_juga[4]<<" | "<<casi_juga[5]<<"\n "
                <<casi_juga[6]<<" | "<<casi_juga[7]<<" |
"<<casi_juga[8]<<endl<<endl;

        if(x>=4)
          if((casi[0]+casi[1]+casi[2])==3 || (casi[3]+casi[4]+casi[5])==3
          || (casi[6]+casi[7]+casi[8])==3 || (casi[0]+casi[4]+casi[8])==3
          || (casi[2]+casi[4]+casi[6])==3 || (casi[0]+casi[3]+casi[6])==3
          || (casi[1]+casi[4]+casi[7])==3 || (casi[2]+casi[5]+casi[8])==3
          || (casi[0]+casi[1]+casi[2])==-3 || (casi[3]+casi[4]+casi[5])==-3
          || (casi[6]+casi[7]+casi[8])==-3 || (casi[0]+casi[4]+casi[8])==-3
          || (casi[2]+casi[4]+casi[6])==-3 || (casi[0]+casi[3]+casi[6])==-3
          || (casi[1]+casi[4]+casi[7])==-3 || (casi[2]+casi[5]+casi[8])==-3)
            jueg_fin=1;

        if(jueg_fin==1)
         {
          if(tipo_jueg==1 && ((prim_pc==1 && juga==1) || (prim_pc==2 && juga==-
1)))
            cout<<"Gané!!!!"<<endl;
          else
            cout<<"Ganaste!!!!"<<endl;
         }
        else
          if(x==8)
            cout<<"Juego empatado"<<endl;
      }
    else
      {
       cout<<"\nCasilla incorrecta, seleccione una válida\n"<<endl;
       cout<<" "<<casi_juga[0]<<" | "<<casi_juga[1]<<" | "<<casi_juga[2]<<"\n "
               <<casi_juga[3]<<" | "<<casi_juga[4]<<" | "<<casi_juga[5]<<"\n "
               <<casi_juga[6]<<" | "<<casi_juga[7]<<" |
"<<casi_juga[8]<<endl<<endl;
       juga=juga*(-1);
       x--;
      }
   }

 return 0;
```

```
}

void Rangos_Gato(int tiro[], int cont, int &acum, int juga, float
casi_juga_prob[])
{
 int x, band=0, inic, fin;
 int perm[8]={40320,5040,720,120,24,6,2,1};
 float prob;

 if(cont==0)
   {
    inic=(tiro[cont]-1)*perm[cont];
    fin=inic+perm[cont]-1;
    acum=inic;
    band=0;
   }
 else
   {
    for(x=0;x<cont;x++)
      if(tiro[x]<tiro[cont])
        band++;

    band++;
    inic=acum+(tiro[cont]-band)*perm[cont];
    fin=inic+perm[cont]-1;
    acum=inic;
    band=0;
   }

 if(cont==8)
   fin=inic;

 if(juga!=0)
   {
    cont_auxi=tiro[cont]-1;
    prob=Lectura_Jug_Gato(inic, fin, juga);
    casi_juga_prob[tiro[cont]-1]=prob;
   }
}

float Lectura_Jug_Gato(int inic, int fin, int juga)
{
 int x;
 float cont_juga_1=0.0, cont_juga_2=0.0, cont_juga_0=0.0;
 int jueg, gana;
 streampos punt;
 ifstream Entrada;

 Entrada.open("jugadas.txt");

 punt=14*(inic);
 Entrada.seekg(punt);

 for(x=0;x<fin-inic+1;x++)
   {
```

```
    Entrada>>jueg;
    Entrada>>gana;

    if(gana==1)
      cont_juga_1++;
    else
      if(gana==-1)
        cont_juga_2++;
      else
        cont_juga_0++;
   }

 Entrada.close();

casi_juga_prob_1[cont_auxi]=cont_juga_1/(cont_juga_1+cont_juga_2+cont_juga_0)*10
0;

casi_juga_prob_2[cont_auxi]=cont_juga_2/(cont_juga_1+cont_juga_2+cont_juga_0)*10
0;

 if(juga==1)
   return cont_juga_1/(cont_juga_1+cont_juga_2+cont_juga_0)*100;
 else
   return cont_juga_2/(cont_juga_1+cont_juga_2+cont_juga_0)*100;
}

void Permutaciones_Main()
{
 int palabra[]={1,2,3,4,5,6,7,8,9};
 int i;

 Permutaciones(palabra);
 Salida.open("jugadas.txt");
 for(i=0;i<362880;i++)
   if(tiradas[i][1]==-1)
     Salida<<tiradas[i][0]<<" "<<tiradas[i][1]<<endl;
   else
    Salida<<tiradas[i][0]<<"  "<<tiradas[i][1]<<endl;

 Salida.close();
}

void Permutaciones(int *cad, int l)
{
 char c;
 int i, j, aux;
 int n=9;
 int
tiro_gana[48]={123,132,147,159,174,195,213,231,258,285,312,321,357,369,375,396,4
17,456,465,

471,519,528,537,546,564,573,582,591,639,645,654,693,714,735,741,753,789,798,
                    825,852,879,897,915,936,951,963,978,987};

 for(i=0;i<n-l;i++)
```

```
   {
    if(n-l>2)
      Permutaciones(cad, l+1);
    else
      {

tiradas[posi][0]=cad[0]*100000000+cad[1]*10000000+cad[2]*1000000+cad[3]*100000+c
ad[4]*10000+cad[5]*1000+cad[6]*100+cad[7]*10+cad[8];
       tiradas[posi++][1]=0;
       aux=0;

       for(j=0;j<48 && aux==0;j++)
         if(cad[0]*100+cad[2]*10+cad[4]==tiro_gana[j])
           {
            tiradas[posi-1][1]=1;
            aux=1;
           }

       for(j=0;j<48 && aux==0;j++)
         if(cad[1]*100+cad[3]*10+cad[5]==tiro_gana[j])
           {
            tiradas[posi-1][1]=-1;
            aux=1;
           }

       for(j=0;j<48 && aux==0;j++)
         if(cad[2]*100+cad[4]*10+cad[6]==tiro_gana[j] ||
cad[0]*100+cad[4]*10+cad[6]==tiro_gana[j] ||
cad[0]*100+cad[2]*10+cad[6]==tiro_gana[j])
           {
            tiradas[posi-1][1]=1;
            aux=1;
           }

       for(j=0;j<48 && aux==0;j++)
         if(cad[3]*100+cad[5]*10+cad[7]==tiro_gana[j] ||
cad[1]*100+cad[5]*10+cad[7]==tiro_gana[j] ||
cad[1]*100+cad[3]*10+cad[7]==tiro_gana[j])
           {
            tiradas[posi-1][1]=-1;
            aux=1;
           }

       for(j=0;j<48 && aux==0;j++)
         if(cad[4]*100+cad[6]*10+cad[8]==tiro_gana[j] ||
cad[2]*100+cad[6]*10+cad[8]==tiro_gana[j] ||
cad[2]*100+cad[4]*10+cad[8]==tiro_gana[j] ||
cad[0]*100+cad[6]*10+cad[8]==tiro_gana[j] ||
cad[0]*100+cad[4]*10+cad[8]==tiro_gana[j] ||
cad[0]*100+cad[2]*10+cad[8]==tiro_gana[j])
           {
            tiradas[posi-1][1]=1;
            aux=1;
           }
      }
```

```
    c=cad[l];
    cad[l]=cad[l+i+1];
    cad[l+i+1]=c;

    if(l+i==n-1)
      {
       for(j=l;j<n;j++)
         cad[j]=cad[j+1];

       cad[n]=0;
      }
   }
}
```

Universidad Autónoma Metropolitana
Unidad Azcapotzalco
Ciencias de Básicas e Ingeniería
Ingeniería en Computación

"Estrategia de teoría de juegos para el *Juego del gato*"

Juan Carlos Chávez Flores
Matrícula: 206304501

Jonathan Jair Lira Rodríguez
Matrícula: 207203766

Trimestre lectivo: 11-O
Fecha: (12 / Diciembre / 2011)

Tercera Versión

Objetivo General

Desarrollar una aplicación de escritorio que implemente el *Juego del gato* haciendo uso de teoría de juegos para estudiar los movimientos de cada jugador y calcular sus probabilidades de ganar con base a la información recopilada en cada turno.

Objetivos Particulares

- Desarrollar un árbol de probabilidades para la aplicación.
- Programar el algoritmo que desarrolle la matriz de probabilidades para cada jugada.
- Desarrollo de pruebas.
- Elaborar la documentación de la aplicación.
- Redactar reporte final sobre la investigación, desarrollo y resultados del Proyecto Terminal.

Introducción

El presente proyecto tiene como propósito mostrar el aspecto probabilístico implícito en el juego del gato. Prácticamente se conoce la complejidad del juego, así como al menos una estrategia sencilla que asegura que en cada partida no exista un ganador/perdedor. En cambio, nuestra aplicación permitirá conocer el espacio probabilístico sobre el que inciden las decisiones de cada jugador.

Justificación

Los algoritmos empleados con anterioridad que resuelven el juego del gato, son creados partiendo de la simetría presente en su tablero de juego, y su lógica gira en torno a un número reducido de variantes predefinidas en el programa, haciendo distinción únicamente de quién haga el primer movimiento.

La evolución de las técnicas de análisis y diseño de algoritmos permiten hoy en día realizar un estudio detallado del estado en el que se encuentre cualquier programa en ejecución en un momento dado. Luego de estudiar el juego y construir su árbol de probabilidades, haremos uso de las técnicas mencionadas para calcular el margen de probabilidad de ganar que tiene cada jugador en relación a sus movimientos en el tablero, limitando también ciertos movimientos que violen los principios básicos del juego.

Antecedentes

Fue en la época del renacimiento cuando Rubino Alberdi escribió el primer tratado matemático en el que combinaba los principios esenciales de las probabilidades con la didáctica del juego, develando con éste sus secretos. **Alberdi demostró que el juego en sí tenía 765 jugadas distintas**.

Este proyecto aborda cada una de las 765 jugadas posibles de Alberdi desde una perspectiva analítica, con la misión de aplicar técnicas computacionales que permitan, mediante el uso de la *teoría de juegos,* recabar la información necesaria para construir el *árbol de juego* cuya complejidad para el juego del gato asciende a 26,830 partidas posibles.

A continuación se hace referencia a propuestas de proyectos terminales de la Universidad Autónoma Metropolitana unidad Azcapotzalco que tienen ciertas relaciones con el proyecto propuesto o utilizan algoritmos similares, pero no llegan al mismo resultado.

Referencias internas

"Algoritmo evolutivo para el problema de coloración difusa" [1]. El proyecto se enfoca en la coloración de grafos. Al igual que el algoritmo propuesto, la solución se obtendrá con la implementación de algoritmos que se desarrollan a lo largo del proyecto, pero con la implementación del algoritmo de coloración difusa. El enfoque del objetivo (Coloración de grafos) difiere del propuesto ya que el resultado de ambos es totalmente diferente.

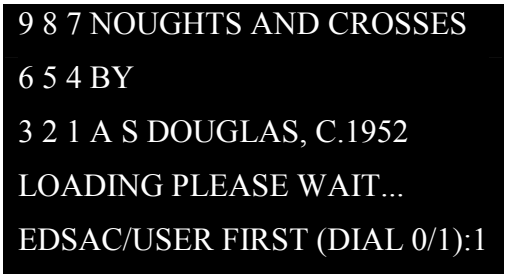
"Implementación del algoritmo k-vecinos más cercanos en MapReduce utilizando Hadoop" [2]. El proyecto tiene como objetivo implementar el algoritmo de k-vecinos más cercanos utilizando el modo de programación MapReduce. Los algoritmos de este proyecto son bastante similares a los que serán implementados en la presente propuesta para el desarrollo del árbol de probabilidades, pero no tiene la misma finalidad ni está relacionado con el tema de juegos.

"Implementación de un algoritmo de aproximación para el problema del cartero con restricciones en los arcos" [3]. En el proyecto se describe el desarrollo y utilización de aproximaciones polinomiales para el problema del cartero en gráficas mixtas. No usa algoritmos heurísticos o genéticos. Su finalidad es el observar el comportamiento del algoritmo para las gráficas creadas. No tiene el mismo objetivo y resultado que el proyecto propuesto.

"Implementación de un árbol de probabilidades con extracción automática del modelo a partir de datos estadísticos de cáncer de mama" [4]. Al igual que el proyecto propuesto, utiliza arboles de decisión para su desarrollo, pero con la finalidad de determinar la probabilidad de ganar que cada jugador tienen dependiendo del nodo en el que se encuentre. El proyecto propuesto es para el tema de juegos, en cambio, la propuesta referida trata sobre el tema de cáncer de mama.

"Un algoritmo de ramificación y acotamiento para el problema de coloración robusta" [5] La propuesta describe la implementación de un algoritmo de ramificación y acotamiento para el problema de coloración, dentro de la herramientas y resultados contiene la utilización de coloración de grafos mínima. Los algoritmos empleados en esta referencia son similares, pero bajo un objetivo distinto.

Referencias externas



Juego del gato sobre EDSAC

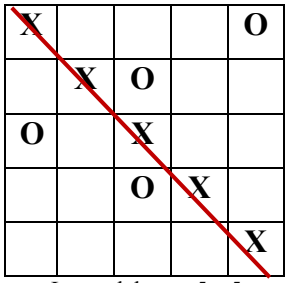
Figura 1.1

En 1952 Alexander Sandy Douglas presentó como tesis doctoral en matemáticas en la Universidad de Cambridge (Inglaterra) el primer juego gráfico de la historia denominado "OXO", orientado al estudio de la interactividad entre seres humanos y computadoras. OXO es una versión del juego del gato para la computadora EDSAC[1], diseñada y construida en la citada universidad. El jugador realizaba sus movimientos mediante un dial telefónico de rueda incorporado en la computadora EDSAC. El juego en ejecución sobre la computadora EDSAC es mostrado en la figura 1.1.

[1] EDSAC, Electronic Delay Storage Automatic Calculator, fue la primera computadora operacional con capacidad para almacenar programas electrónicos.

Posteriormente fueron creadas diversas variantes:

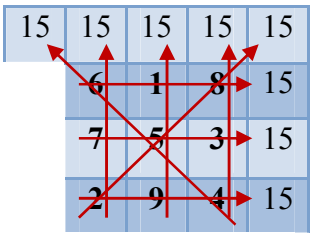
- El juego del gato sobre un tablero de *n x n* casillas, con *n <= 6* (donde teóricamente el valor de *n* puede ser tan grande como se desee, pero el juego pierde su objetivo principal, el divertir, con valores superiores a 6). La figura 1.2 muestra un burdo ejemplo de juego sobre un tablero de *5 x 5* casillas. En esta variante persisten las reglas del juego original, un movimiento por jugador de manera alternada tratando de cruzar el tablero de extremo a extremo con la ficha elegida.



Juego del gato *5 x 5*

Figura 1.2

- El cuadrado mágico es un arreglo de números sobre un tablero donde cada fila, columna y las diagonales principales suman la misma cantidad, a la que se hace referencia como *constante mágica*. En este juego se disponen de tantos números como casillas tenga el tablero utilizado, siguiendo la sucesión *1, 2, 3,..., n*, donde n es el número de casillas. Cada número puede ser usado solo una vez.
En su turno, cada jugador puede rellenar tantas casillas vacías como desee, siempre y cuando se encuentren en la misma fila o columna (no en diagonal). De cualquier forma, el jugador que rellene la última casilla será el ganador, independientemente del número de casillas que haya rellenado. La figura 1.3 muestra un ejemplo clásico de cuadro mágico 3 x 3.



Cuadrado mágico 3 x 3

Figura 1.3

- Qubic es el nombre comercial de un juego *4 en línea* creado en 1953 por la compañía Parker Brothers (originalmente fue fabricado con cuatro tableros sobrepuestos de plástico transparente con 16 casillas marcadas en cada uno), se juega sobre una matriz de 4 x 4 x 4 casillas por dos o tres jugadores a la vez. Debido a sus dimensiones, existen 76 posibles formas de conformar líneas con cuatro fichas del mismo color (rojas, amarillas o azules), así que el primer jugador en lograr 2 líneas será el ganador de la partida.
En 1975 Arthur Hu y Carl Hu escribieron un videojuego con gráficos 3D en la Universidad Lindberg[2], el cual dio vida a Qubic sobre computadora HP 9830. Posteriormente fue adaptada una versión de este juego a la computadora demo HP 2647, misma que utilizaba una transformada matemática simple para resolver la posición de entrada tridimensional.

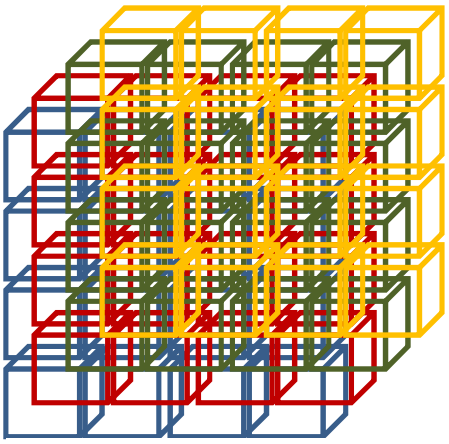

Qubic 4 x 4 x 4

Figura 1.4a

La figura 1.4a presenta la matriz tridimensional vacía del Qubic 4 x 4 x 4.

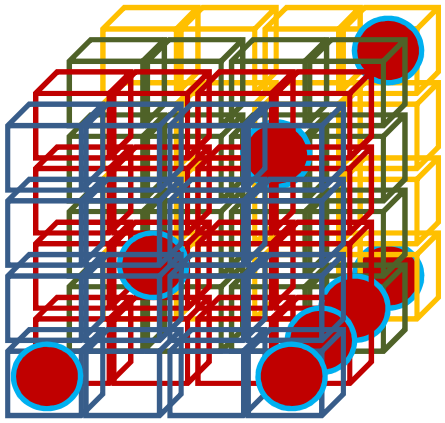

Qubic 4 x 4 x 4 (líneas ganadoras)

Figura 1.4b

La figura 1.4b ilustra como el jugador con fichas rojas gana la partida al completar dos líneas ganadoras, una horizontal y una diagonal. Este ejemplo muestra únicamente 2 de las 76 posibles formaciones válidas, despreciando las fichas del (de los) oponente(s).

A diferencia de los ejemplos mostrados, la solución que presentaremos se apegará a las componentes del modelo de teoría

ndada en 1972, se encuentra en la región sudeste de Renton, Washington.

de juegos y basará su lógica en el árbol probabilístico construido a partir de la información derivada del análisis de estrategias y tácticas del juego del gato original.

La originalidad de la solución residirá en la forma de abordar cada movimiento de la partida en juego ya que, dependiendo del nivel del árbol en el que se encuentre y la rama que se esté recorriendo, la aplicación deberá proporcionar el porcentaje de probabilidad de ganar con que cada jugador cuenta en cada turno.

Descripción Técnica

- De acuerdo al modelo de teoría de juegos se tienen tres componentes principales: jugadores, estrategias y beneficio para cada jugador. Este proyecto hará uso de estrategias heurísticas[3] con la intensión de acercar a los jugadores a su objetivo, el beneficio que obtendrán al ganar una partida.
  Serán dos las estrategias heurísticas a emplear en el desarrollo de este proyecto:
  - El trabajo hacia adelante o método sintético.- Para dar sentido y relación a la información recabada durante en análisis y elaboración del árbol de probabilidades del juego del gato.
  - El método sistémico.- Para modelar el árbol de probabilidades y determinar sus componentes, así como las relaciones simétricas que existen entre ellos.
- Se construirá manualmente el árbol de probabilidades del juego, mismo que servirá como guía para desarrollar el algoritmo generador de la matriz de probabilidades y para validar el correcto funcionamiento del mismo.
- Dado que existe una técnica que garantiza que nunca se pierda un juego (*Alexander Sandy Douglas, 1952*) se puede calcular en cualquier momento del juego la probabilidad de ganar la partida para cada jugador.

Especificación Técnica

El presente proyecto abordará el juego del gato

/*1*/ Desarrollo del árbol de probabilidades que servirá como base para la creación del algoritmo.
/*2*/ Desarrollo del algoritmo que ayude a llenar la matriz de probabilidades.
/*3*/ Aplicación de pruebas al algoritmo teniendo como ayuda el árbol de probabilidades.
/*4*/ Desarrollo de la función que administre el *juego del gato*.
/*5*/ Aplicación de pruebas al algoritmo administrador del juego.
/*6*/ Implementación del entorno gráfico de la aplicación.
/*7*/ Consolidación de resultados y de algoritmos para obtener una sola aplicación.
/*8*/ Aplicación de pruebas, personal ajeno al proyecto competirá contra la computadora.
/*9*/ Elaboración de la documentación de la aplicación.
/*10*/ Redacción de reporte del Proyecto Terminal.

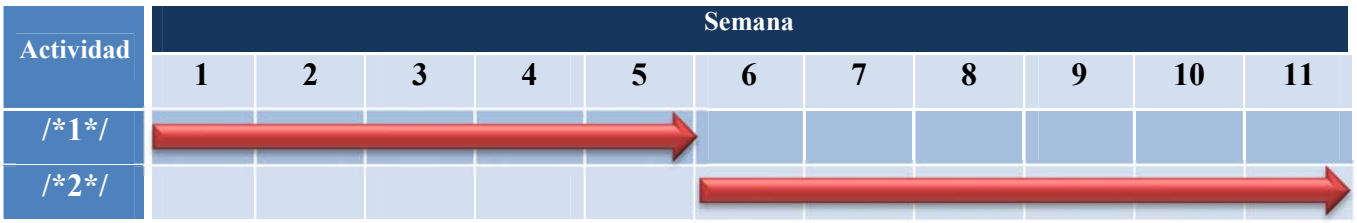
El presente proyecto se tomará por concluido una vez que:
- la aplicación responda satisfactoriamente a la ronda de pruebas especificada en el numeral(#) anterior
- se cumpla cada punto marcado en la lista de entregables

[3] Las **estrategias heurísticas** son los sentidos de orientación que pueden seguirse en el razonamiento para conectar los datos con la solución, forman el paso de descubrimiento e invención durante el proceso de resolución de un problema.

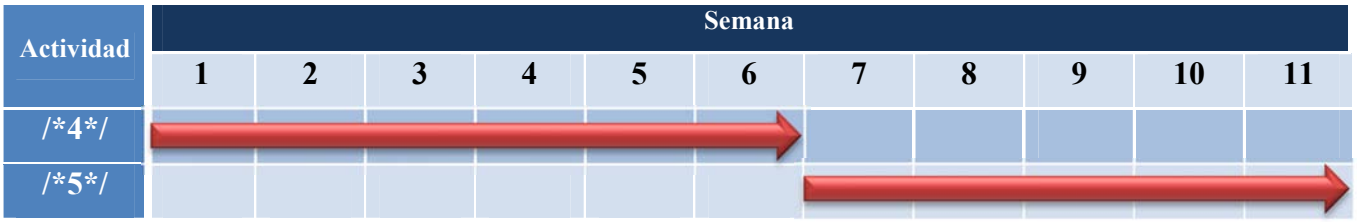
Calendario de Trabajo

Trimestre 12-I, Proyecto Terminal de Ingeniería en Computación I (9 horas / semana).

Juan Carlos Chávez Flores

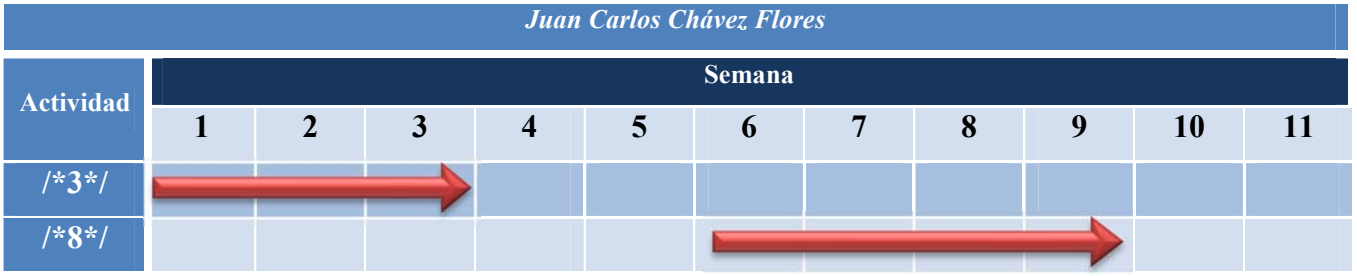


Jonathan Jair Lira Rodríguez

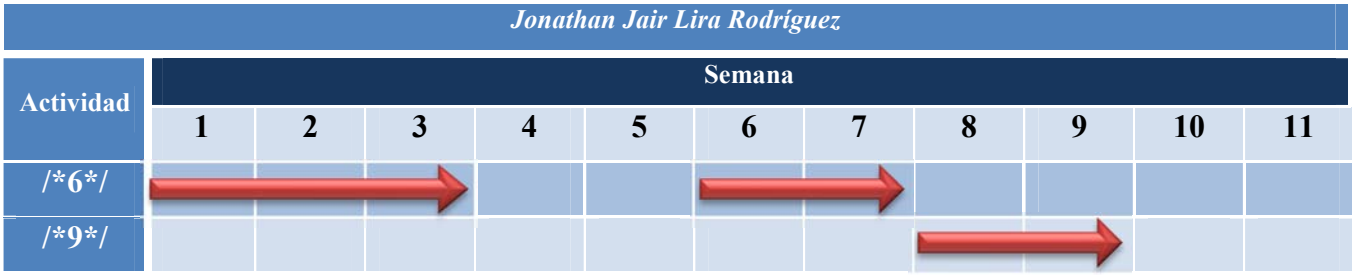


Trimestre 12-P, Proyecto Terminal de Ingeniería en Computación II (18 horas / semana).

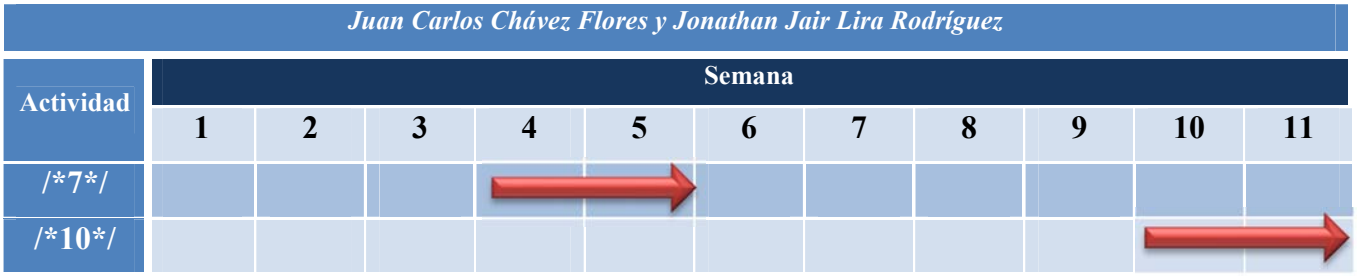
Juan Carlos Chávez Flores



Jonathan Jair Lira Rodríguez



Juan Carlos Chávez Flores y Jonathan Jair Lira Rodríguez



Recursos

- Dos computadoras con sistema operativo Windows 7 (personales).
- Software Dev C++.
- Software Eclipse.
- Suite de Microsoft Office.

Los alumnos ya cuentan con estos recursos en sus respectivas computadoras personales.

Entregables

- CD que contiene:
  - Aplicación
  - Propuesta de Proyecto Terminal
  - Reporte del plan de pruebas
  - Código fuente comentado
  - Reporte del Proyecto Terminal

Bibliografía

[1] A. A. Vázquez Cortés "Algoritmo evolutivo para el problema de coloración difusa," Propuesta de proyecto terminal Ing. En computación, Dept. Sist., UAM-Azc., Ciudad de Méx., D.F., Méx., 2009.

[2] E. Martínez Cruz "Implementación del algoritmo k-vecinos más cercanos en MapReduce utilizando Hadoop," Propuesta de proyecto terminal Ing. En computación, Dept. Sist., UAM-Azc., Ciudad de Méx., D.F., Méx., 2011.

[3] R. Zambrano Gervacio "Implementación de un algoritmo de aproximación para el problema del cartero con restricciones en los arcos," Propuesta de proyecto terminal Ing. En computación, Dept. Sist., UAM-Azc., Ciudad de Méx., D.F., Méx., 2011.

[4] J. A. Aguirre López "Implementación de un árbol de decisiones con extracción automática del modelo a partir de datos estadísticos de cáncer de mama," Propuesta de proyecto terminal Ing. En computación, Dept. Sist., UAm-Azc., Ciudad de Méx., D.F., Méx., 2010.

[5] E. F. Rosas Coronado "Un algoritmo de ramificación y acotamiento para el problema de coloración robusta," Propuesta de proyecto terminal Ing. En computación, Dept. Sist., UAM-Azc., Ciudad de Méx., D.F., Méx., 2010.

[6] J. Beck, Combinatorial Games: Tic-Tac-Toe Theory, Cambridge University Press, Cambridge, 2008.

[7] Patashnik, O, Qubic: 4×4×4 Tic-Tac-Toe, Mathematics Magazine, Vol. 53, 1980, Septiembre, 202–216.

[8] Székely, L. A., On two concepts of discrepancy in a class of Combinatorial Games, Colloq. Math. Soc. János Bolyai Vol. 37 "Finite and Infinite Sets," North-Holland, 1981, 679–683.

[9] Buzzle [Online]. Disponible: http://www.buzzle.com/articles/tic-tac-toe-strategy-guide.html. Consultada en noviembre 2011

[10] Wikipedia [Online]. Disponible en: http://es.wikipedia.org/wiki/OXO.

[11] Sitio web Math Lair [En línea]. Disponible : http://mathlair.allfunandgames.ca/tictactoe.php. Consultada en noviembre 2011

[12] Math Lair [Online]. Disponible : http://mathlair.allfunandgames.ca/magicsquares.php. Consultada en noviembre 2011
[13] Wikipedia [Online]. Disponible : http://en.wikipedia.org/wiki/Qubic. Consultada en noviembre 2011
[14] IQFlash [Online]. Disponible : http://www.iqflash.com/tic-tac-toe-silverman-article.shtml. Consultada en noviembre 2011
[15] Wikipedia [Online]. Disponible : http://en.wikipedia.org/wiki/Lindbergh_Senior_High_School_(Renton,_Washington). Consultada en noviembre 2011
[16] Wikipedia [Online]. Disponible : http://es.wikipedia.org/wiki/Heur%C3%ADstica. Consultada en noviembre 2011
[17] Alammi [Online]. Disponible: http://www.alammi.info/revista/numero2/pon_0010.pdf. Consultada en noviembre 2011

Universidad Autónoma Metropolitana – Unidad Azcapotzalco

División de Ciencias Básicas e Ingeniería

Ingeniería en Computación

Reporte de Proyecto Terminal II

ESTRATEGIA DE TEORÍA DE JUEGOS PARA EL JUEGO DEL GATO

Jonathan Jair Lira Rodríguez

207203766

Juan Carlos Chávez Flores

206304501

Trimestre Lectivo: 12-P

Julio 2012

Dr. Pedro Lara Velázquez

Número económico: 31213

Tabla de contenido

ESTRUCTURA DEL PRESENTE REPORTE:

Por ser dos las personas quienes integramos el equipo de desarrollo del presente proyecto y al ser un requisito estipulado en el "*Instructivo para la elaboración de la Propuesta de Proyecto Terminal de Ingeniería en Computación*" fueron divididas algunas de las actividades, y sólo para dos de éstas se comparte la responsabilidad ya que son necesarias para acoplar los módulos realizados de forma independiente. A continuación se enlistan las 10 actividades involucradas en la implementación de una estrategia (probabilística) de teoría de juegos para el juego del gato:

- **/*1*/** Desarrollo del árbol de probabilidades que servirá como base para la creación del algoritmo.
- **/*2*/** Desarrollo del algoritmo que ayude a llenar la matriz de probabilidades.
- **/*3*/** Aplicación de pruebas al algoritmo teniendo como ayuda el árbol de probabilidades.
- **/*4*/** Desarrollo de la función que administre el juego del gato.
- **/*5*/** Aplicación de pruebas al algoritmo administrador del juego.
- **/*6*/** Implementación del entorno gráfico de la aplicación.
- **/*7*/** Consolidación de resultados y de algoritmos para obtener una sola aplicación.
- **/*8*/** Aplicación de pruebas, personal ajeno al proyecto competirá contra la computadora.
- **/*9*/** Elaboración de la documentación de la aplicación (código fuente comentado).
- **/*10*/** Redacción de reporte del Proyecto Terminal.

La tabla 1 muestra la distribución de actividades a realizar por cada alumno y de manera conjunta en la segunda etapa de implementación de la aplicación, comprendidas en un intervalo de tiempo de 11 semanas.

Juan Carlos Chávez Flores										
Actividad	**Semana**									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
/*3*/	→	→	→								
/*8*/						→	→	→	→		

Jonathan Jair Lira Rodríguez											
Actividad	**Semana**										
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
/*6*/											
/*9*/											

Juan Carlos Chávez Flores y Jonathan Jair Lira Rodríguez											
Actividad	**Semana**										
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
/*7*/											
/*10*/											

Tabla 1. Distribución de actividades

I SEMANAS 1, 2 Y 3 – APLICACIÓN DE PRUEBAS AL ALGORITMO TENIENDO COMO AYUDA EL ÁRBOL DE PROBABILIDADES E IMPLEMENTACIÓN DEL ENTORNO GRÁFICO DE LA APLICACIÓN.

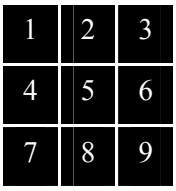
Esta aplicación tiene por objetivo el ofrecer orientación al jugador en turno en cuanto a la toma de decisiones de acuerdo a una matriz de probabilidad creada en tiempo de ejecución a partir de un estudio minucioso practicado al árbol de probabilidad que contiene la cantidad de permutaciones resultantes de alternar entre las diversas opciones de tiro de los jugadores.

Árbol de probabilidad

La cantidad de permutaciones obtenida asciende a 362,880, número que resulta de aplicar la función factorial al número total de opciones de tiro del juego (9!)[1].

[1] El valor de un factorial es el producto de todos los números desde 1 hasta el número del factorial a calcular. Los factoriales se utilizan para determinar las cantidades de combinaciones y permutaciones y para averiguar probabilidades.

A cada casilla del tablero le fue asignado un identificador numérico siguiendo un orden ascendente de izquierda a derecha y de arriba hacia abajo, el número inicial será el dígito 1, este identificador facilitará su visualización en las etapas de desarrollo, pruebas e interacción con el usuario. La figura 1 ejemplifica el tablero de juego acorde al orden descrito anteriormente.



Identificadores

Figura 1

El árbol de probabilidad resulta de un algoritmo iterativo que arroja como salida un archivo de texto plano con dos columnas, la primera describe el trayecto seguido durante cada partida, y la segunda columna denota al primer jugador cuya secuencia de opciones de tiro produce una combinación ganadora.

Matriz de probabilidad

La base sobre la cual se calculan las diferentes probabilidades proviene de un extracto del árbol de probabilidad. El algoritmo que calcula la matriz de probabilidad (en adelante la **Matriz**) acota la base conforme se desarrolla la partida de juego de tal forma que el rendimiento y la eficiencia sobre el uso de recursos se ve favorecida. Su lógica se rige por los siguientes pasos:

1) Recorre el árbol de probabilidad para ofrecer al jugador en el primer turno las probabilidades de ganar que cada casilla posee.
2) Recibe el número de casilla del usuario a través de la consola y lo almacena en una variable.
3) Genera una bandera con valor inicial 0, ésta guardará posteriormente el conteo de casillas cuyo identificador sea menor al identificador de la casilla en turno y sumará 1 al conteo final.
4) Genera un contador para hacer referencia al arreglo cuyos elementos ordenados en forma decreciente corresponden al valor calculado manualmente de los factoriales 8!, 7!, 6!, 5!, 4!, 3!, 2!, y 1! respectivamente.
5) Resta al identificador en turno el valor calculado de la bandera y multiplica el resultado de la operación anterior por el elemento del arreglo referenciado por el contador y lo designa como cota inferior del arreglo a analizar.
6) Suma a la cota inferior el mismo elemento referido por el contador, le resta 1 y lo asigna como cota superior del arreglo a analizar.

7) Incrementa el contador en uno.
8) Busca las secuencias de tiro que producen una combinación ganadora dentro del arreglo previamente acotado y conserva una cuenta de éstas, así como también de las secuencias que producirían pérdida del juego y empate.
9) Lleva a cabo el cálculo de probabilidades y las ofrece al jugador en turno a través del entorno gráfico (en adelante la **Interfaz**).
10) Repite los pasos 2 al 9 un máximo de ocho veces, punto en el que el juego está totalmente definido.

Función de Administración del juego del gato

Antes de iniciar el juego, la función de administración (en adelante el **Administrador**) consulta sobre qué tipo de juego desea invocar, ofreciendo un par de opciones:

1) Usuario Vs Máquina: habilita una contienda entre el usuario y el algoritmo predefinido.
2) Usuario Vs Usuario: habilita una contienda entre un par de usuarios.

Nota: Por default, si el usuario NO elige un dígito válido (1 ó 2), la contienda habilitada será Usuario Vs Usuario.

Para el caso en que el usuario elija la opción 1, el Administrador nuevamente ofrece un par de opciones para elegir entre ser el primero en tirar o para ceder el turno a la máquina:

1) Máquina.
2) Usuario.

Nota: Por default, si el usuario NO elige un dígito válido (1 ó 2), será el Usuario quien tire primero.

Para que el primer jugador en tirar visualice en la Interfaz la probabilidad de ganar para cada casilla, el Administrador invoca al generador del árbol de probabilidad el cual al término de su ejecución genera un archivo de salida de nombre "Jugadas.txt". Posteriormente, el Administrador ordena a la Matriz barrer el contenido del archivo y realizar los cálculos pertinentes para cada casilla, los resultados son enviados al Administrador para posteriormente ser proyectados al usuario mediante la Interfaz.

Luego del primer turno e independientemente de quien lo haya realizado, en los subsecuentes cálculos de probabilidad sólo tendrán participación la Interfaz, el Administrador y la Matriz ya que, como se hizo mención anteriormente, generar las 362,880 jugadas posibles para su posterior análisis consumiría demasiados recursos y afectaría directamente a la velocidad y al rendimiento de la aplicación.

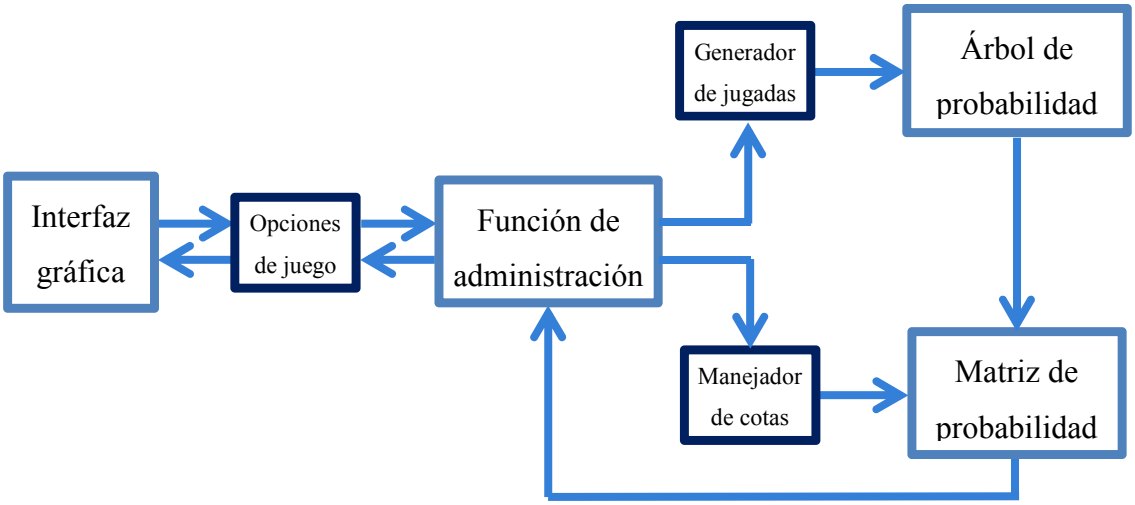


Figura 2. Diagrama de operación de la Función de Administración

Entorno gráfico

A través de la ventana de comandos es presentado inicialmente un menú de selección seguido de un tablero en cuyas casillas está contenido el cálculo de la probabilidad de ganar correspondiente a su posición. Luego de realizarse la primera jugada serán presentados dos tableros de juego por turno, el primero mostrará el estado actual del juego y el segundo indicará la probabilidad de ganar recalculada en consideración a las casillas ocupadas por cada uno de los jugadores partícipes en la partida.

Según sea el caso, la Interfaz proveerá orientación al usuario en situaciones anómalas. Por ejemplo, si un jugador pretende seleccionar una casilla ocupada recibirá el mensaje "*Casilla incorrecta, seleccione una válida*". A su vez, si un jugador logra completar una línea en el tablero, el mensaje proporcionado será "*Ganaste!!!!*". Por el contrario, si la máquina completa primero la línea, el mensaje proporcionado será *"Gané!!!!"*.

II SEMANAS 4 Y 5 - CONSOLIDACIÓN DE RESULTADOS Y DE ALGORITMOS PARA OBTENER UNA SOLA APLICACIÓN.

Durante la redacción de la propuesta de proyecto terminal fue elaborada una estructura de la aplicación tomando en consideración todos los elementos con que hasta ese momento contábamos, pero una vez adentrados en el desarrollo nos percatamos de pequeños detalles que no fueron tomados en cuenta y que cambiaron radicalmente la estructura original. Los sub-módulos ***Opciones de juego, Generador de jugadas y Manejador de cotas*** (figura 3) se derivan de una estrategia de programación extrema para no afectar el esquema planteado originalmente.

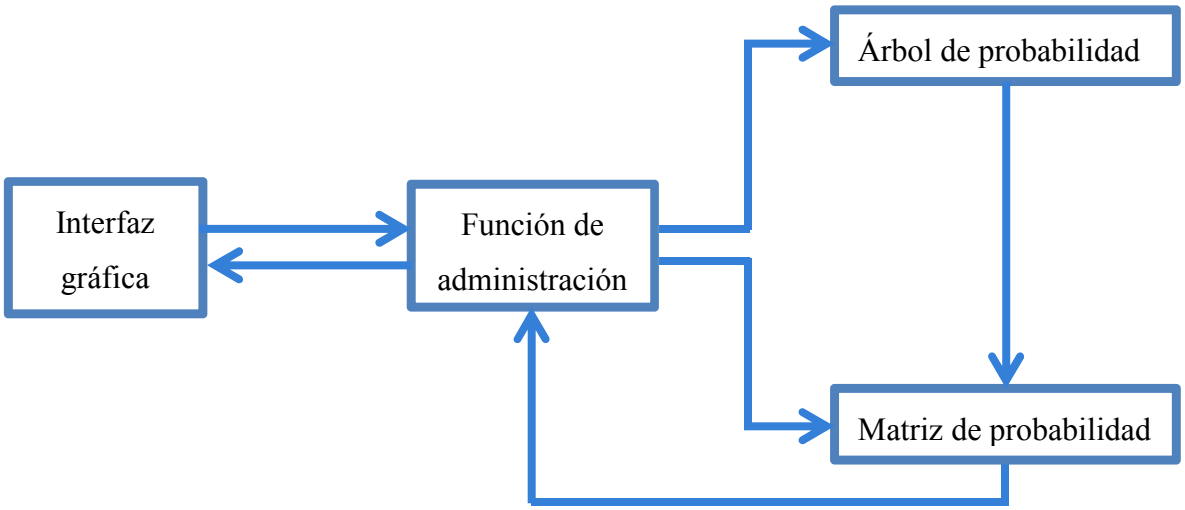


Figura 3. Diagrama general de funcionamiento de la aplicación

¿Cómo se comunican los 4 módulos?

La Interfaz es el medio de comunicación entre el usuario y el Administrador, todas las funciones que ofrece la aplicación pueden ser controladas con los dígitos 1 al 9.

El sub-módulo *Generador de jugadas* no recibe parámetros y sólo es invocado una vez al inicio de cada partida. Éste permuta el orden en que son elegidas las 9 casillas del tablero para generar el espacio muestral[1] del juego, cada permutación representa a una partida jugada. Antes de insertarlas en el árbol de probabilidad, analiza cada combinación en busca de secuencias que produzcan una línea en el tablero conformada por tres elementos del mismo tipo y produce una nueva columna que indica al ganador de la partida o empate,

según sea el caso. Ya con este par de datos, el generador de jugadas inicia la inserción de información en el árbol de probabilidad.

El *Manejador de cotas* es otro sub-módulo de la aplicación diseñado, por una parte, para limitar el rango de operación de la Matriz, y por otra, para incrementar la eficiencia de la aplicación. Como parámetros de entrada requiere una bandera, un contador, la cota actual inferior y el identificador de casilla elegido por el jugador en turno. Como salida ofrece tres contadores, uno por cada jugador y el tercero que lleva el registro de las partidas empatadas.

Su función primordial consiste en fraccionar más y más la cantidad de operaciones ejecutadas sobre el árbol de probabilidad por cada turno.

Por último, el sub-módulo *Opciones de juego* es un *switch* utilizado para alternar entre dos modalidades de partida, además que permite elegir quién tirará primero para una de ellas.

En conjunto, módulos y sub-módulos son provistos de comunicación entre sí por la intervención del Administrador, pero el 80% de la carga de procesamiento es realizada por la Matriz.

III SEMANAS 6 Y 7 – SEGUIMIENTO AL TRABAJO DE IMPLEMENTACIÓN DEL ENTORNO GRÁFICO DE LA APLICACIÓN Y REALIZACIÓN DE PRUEBAS.

Luego de culminar la parte gráfica de la aplicación procedemos a ejecutar una serie de pruebas:

a) Jugada válida: detectará si es posible o no digitar el identificador de una casilla ocupada.
b) Juego ganado: analizará el comportamiento de la aplicación al momento de completarse una línea en el tablero por tres símbolos iguales.
c) Juego empatado: analizará el comportamiento de la aplicación llegada la novena jugada.
d) Juego perdido: analizará el comportamiento de la aplicación luego de que la máquina haya ganado la partida.

e) Máxima probabilidad de ganar: corroborará el cálculo de la máxima probabilidad de ganar con base al cálculo manual de probabilidades auxiliado por una hoja de cálculo.
f) Baja probabilidad de ganar: corroborará el cálculo de la probabilidad más baja de ganar con base al cálculo manual de probabilidades auxiliado por la hoja de cálculo descrita en el numeral anterior.
g) Probabilidad de ganar nula: corroborará el cálculo de la probabilidad nula de ganar con base al cálculo manual de probabilidades auxiliado por la hoja de cálculo descrita en el numeral anterior.

IV SEMANAS 8 Y 9 – SEGUIMIENTO A LA FASE DE APLICACIÓN DE PRUEBAS Y COMENTARIOS AL CÓDIGO DE LA APLICACIÓN

El plan de pruebas está pensado para detectar puntos débiles que pudiere presentar la aplicación, contiene una serie de ejecuciones del programa bajo condiciones estudiadas y controladas que demuestran o no su adecuado funcionamiento.

El Reporte del Plan de Pruebas conforma el "Anexo A" y se adjunta al presente documento.

El código fuente posee una estructura ordenada, lo que facilita su legibilidad. Además, ha sido debidamente comentado e indentado. El código fuente conforma el "Anexo B" y se adjunta al presente documento.

V SEMANAS 10 Y 11 – REDACCIÓN DEL REPORTE DEL PROYECTO TERMINAL.

Durante la redacción de este documento a la aplicación se le estaban practicando más pruebas y una serie de revisiones para intentar detectar cualquier fallo que pudiese ser generado a causa de un defecto en alguno de los módulos del programa o, quizá, por defectos en la planeación del mismo. Hecho que favoreció en cuanto a la generación en tiempo real de información y por lo tanto, no caímos en la necesidad de elaborar una gran cantidad de versiones del reporte del proyecto terminal.

En la primera semana fueron incluidos las descripciones de los módulos, su lógica, y pseudocódigo de ciertas partes importantes de la aplicación. La segunda semana fue

dedicada a darle estructura al reporte, incluir referencias, hacer revisiones ortográficas y sintácticas y porque no, hacer nuevamente inspecciones al desempeño de la aplicación, a su precisión y a que funcionara correctamente respetando las directivas del juego de gato.

VI ANEXOS

Anexo A – Reporte del Plan de Pruebas.pdf

Anexo B – Código_Fuente_Aplicación.pdf

VII REFERENCIAS

[1] R. E. Walpole, R. H. Mayers, S. L. Mayers, "Probabilidad y estadística para ingenieros", Sexta Edición, Pearson Educación, México, 1999, 10-50.

[2] B. Kernighan, D. Ritche, "The C Programming Language", NJ: Prentice Hall, Englewood Cliffs, 1988.

[3] A. A. Vázquez Cortés "Algoritmo evolutivo para el problema de coloración difusa", Propuesta de proyecto terminal Ing. En computación, Dept. Sist., UAM-Azc., Ciudad de Méx., D.F., Méx., 2009.

[4] R. Zambrano Gervacio "Implementación de un algoritmo de aproximación para el problema del cartero con restricciones en los arcos," Propuesta de proyecto terminal Ing. En computación, Dept. Sist., UAM-Azc., Ciudad de Méx., D.F., Méx., 2011.

[5] E. F. Rosas Coronado "Un algoritmo de ramificación y acotamiento para el problema de coloración robusta," Propuesta de proyecto terminal Ing. En computación, Dept. Sist., UAM-Azc., Ciudad de Méx., D.F., Méx., 2010.

[6] J. Beck, Combinatorial Games: Tic-Tac-Toe Theory, Cambridge University Press, Cambridge, 2008.

[7] Székely, L. A., On two concepts of discrepancy in a class of Combinatorial Games, Colloq. Math. Soc. János Bolyai Vol. 37 "Finite and Infinite Sets," North-Holland, 1981, 679–683.

[8] Sitio web Math Lair [En línea]. Disponible : http://mathlair.allfunandgames.ca/tictactoe.php. Consultada en mayo 2012

[9] Math Lair [Online]. Disponible: http://mathlair.allfunandgames.ca/magicsquares.php. Consultada en mayo 2012

[10] IQFlash [Online]. Disponible: http://www.iqflash.com/tic-tac-toe-silverman-article.shtml. Consultada en junio 2012

[11] Sitio web Eclipse Documentation [En línea]. Disponible: http://help.eclipse.org/juno/index.jsp. Consultada en junio 2012

[12] Alammi [Online]. Disponible: http://www.alammi.info/revista/numero2/pon_0010.pdf. Consultada en junio 2012

Universidad Autónoma Metropolitana – Unidad Azcapotzalco

División de Ciencias Básicas e Ingeniería

Ingeniería en Computación

Reporte del Plan de Pruebas

ESTRATEGIA DE TEORÍA DE JUEGOS PARA EL JUEGO DEL GATO

Jonathan Jair Lira Rodríguez

207203766

Juan Carlos Chávez Flores

206304501

Trimestre Lectivo: 12-P

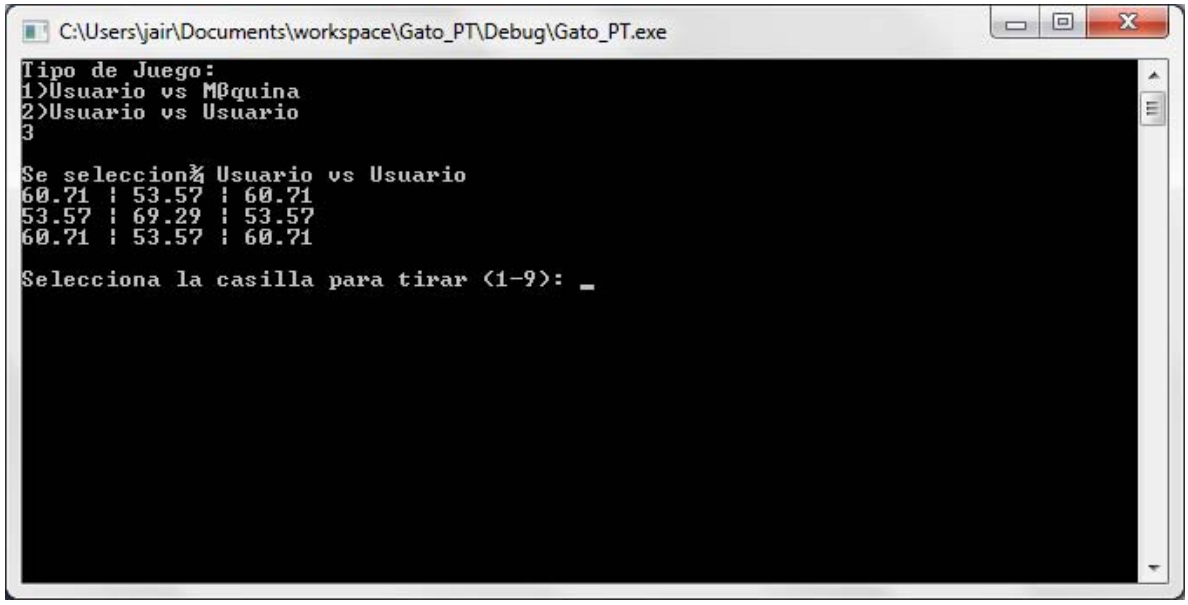
Julio 2012

Dr. Pedro Lara Velázquez

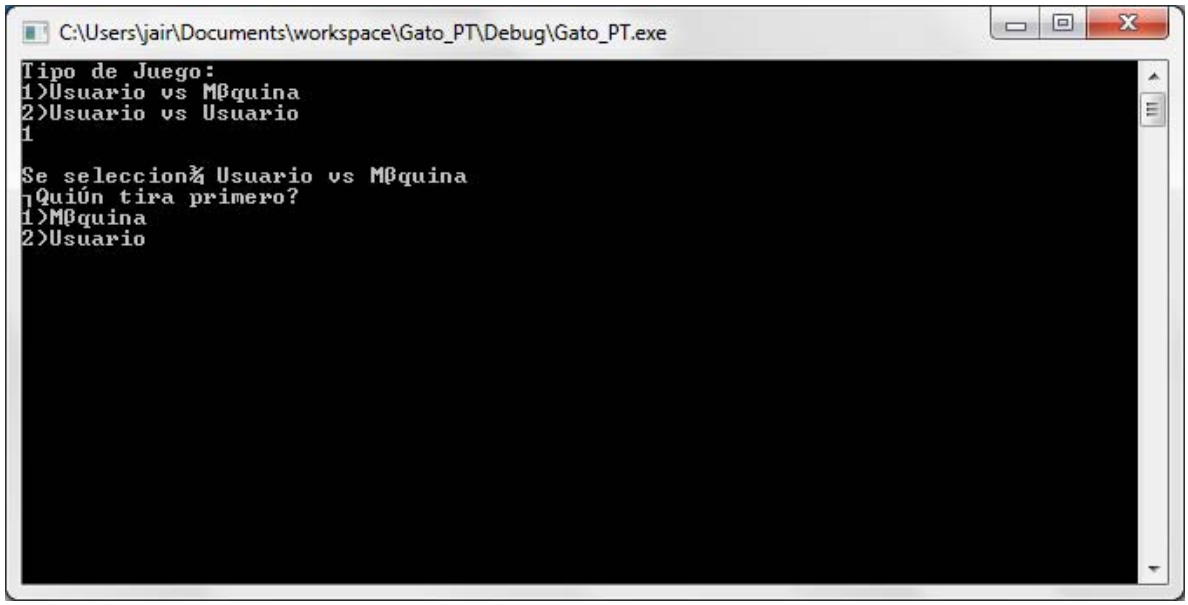
Número económico: 31213

Pruebas

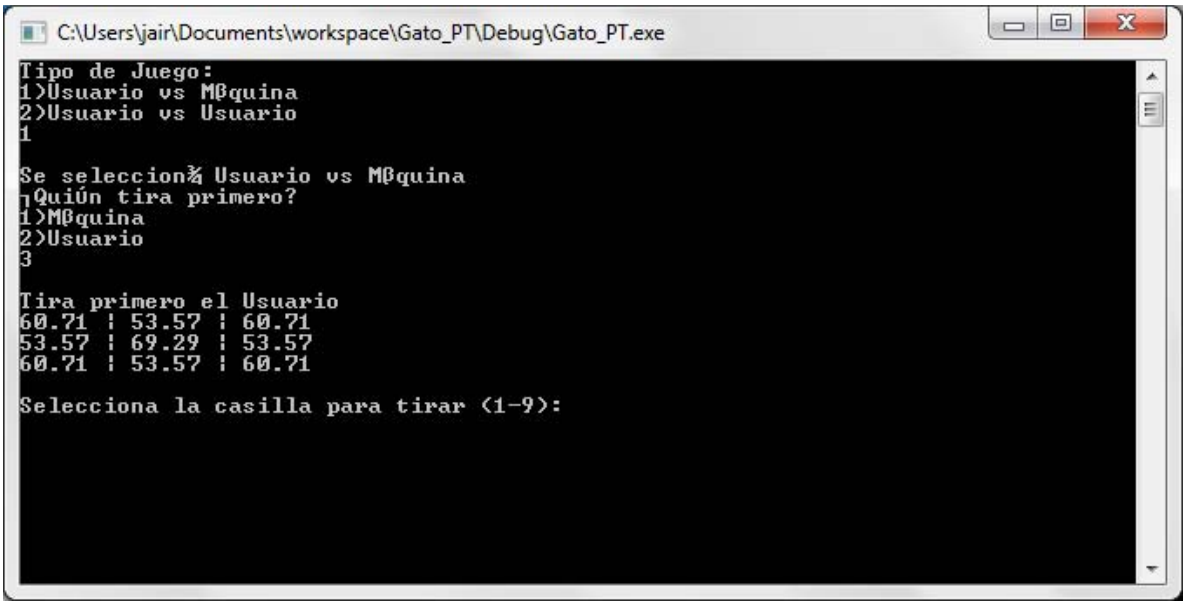
Al inicio, la aplicación muestra un menú para seleccionar el tipo de juego o modalidad del mismo. Al no seleccionar una opción correcta (Usuario vs Máquina o Usuario vs Usuario, 1 ó 2) el Administrador asigna por defecto el modo Usuario vs Usuario.



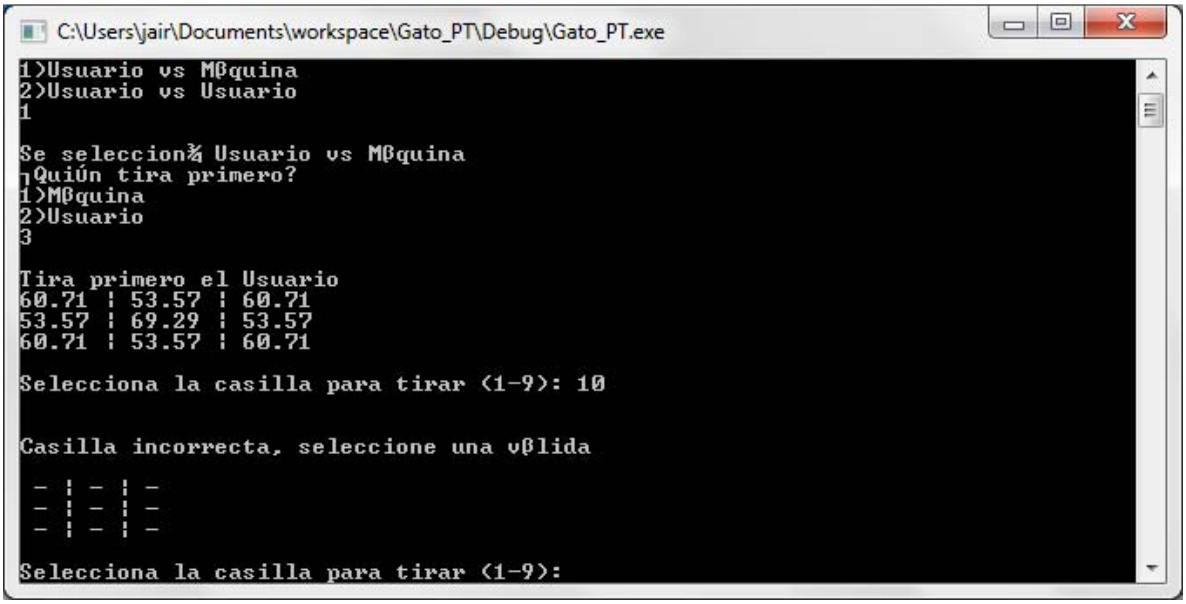
Al seleccionar Usuario vs Máquina el Administrador pide se seleccione quién iniciará la partida, en caso de que no se seleccionar una opción válida el Administrador le da preferencia al Usuario y éste iniciará la partida.



Iniciado el encuentro, en cualquiera modalidad, se muestra una tabla de probabilidades de ganar por casilla y se pide se seleccione alguna de ellas.

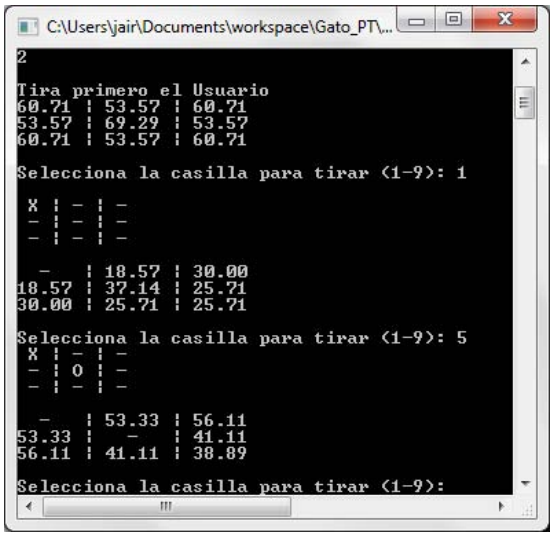


Al no seleccionar una casilla válida, ya sea porque no existe o porque esté ocupada, el Administrador solicitará se seleccione una correcta y se mostrará el tablero del Gato con las jugadas hechas hasta el momento.



Se continúa con la partida hasta que se hagan los tres en línea o se llenen las nueve casillas y se declare un empate:

La primera tabla que muestra la aplicación se obtiene al tomar todas las combinaciones posibles que empiezan con el número de casilla correspondiente, por ejemplo:

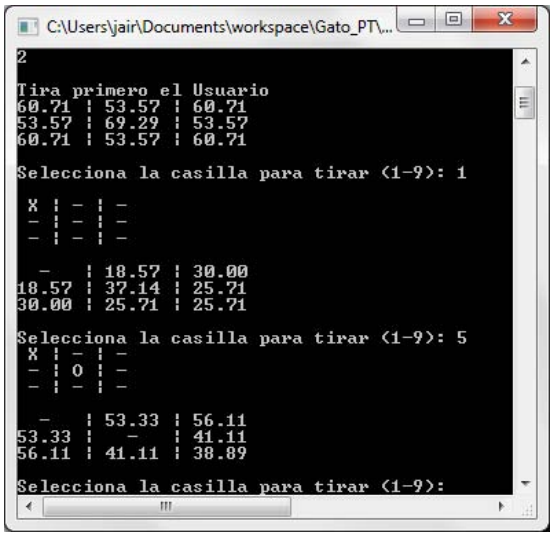


Tomando la casilla 5 (superior izquerda), se cuentan la cantidad de combinaciones del tipo 5abcdefgh, siendo a, b, c, d, e, f, g, h números diferentes entre 1 y 9 (sin el 5). Estas combinaciones dan un total de 40320 números diferentes, mismos que se toman del archivo jugadas.txt, al tomar este conjunto de números de la base se van contabilizando cuantos 1's, 0's y -1's entán relacionados a las combinaciones para que al final de la extracción se pueda calcucular la probabildad de cada caso.

Concluyendo el ejemplo:

1's=27936 apariciones, lo que nos da una probabilidad de 0.692857 (27936/40320)

Este proceso se hace al inicio 9 veces (para los números que inician con 1, 2, 3, etc) para obtener la tabla inicial de probabilidades.



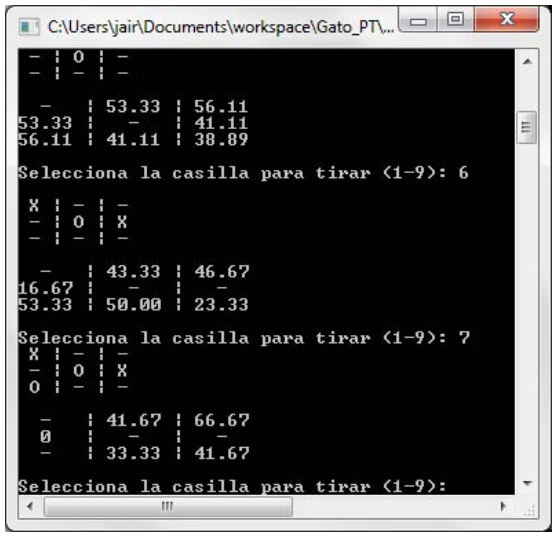
Continuando con el flujo de la aplicación ésta nos pide que ingresemos un número para que sea ahí donde se registre nuestro tiro, si tecleamos 1 la siguiente tabla de problabilidades se va a calcular para sólo 8 casillas ya que la número 1 ya está ocupada. El cálculo es muy similar al trabajado con la tabla inicial, sólo que en este punto se contabilizarán las combinaciones del tipo:

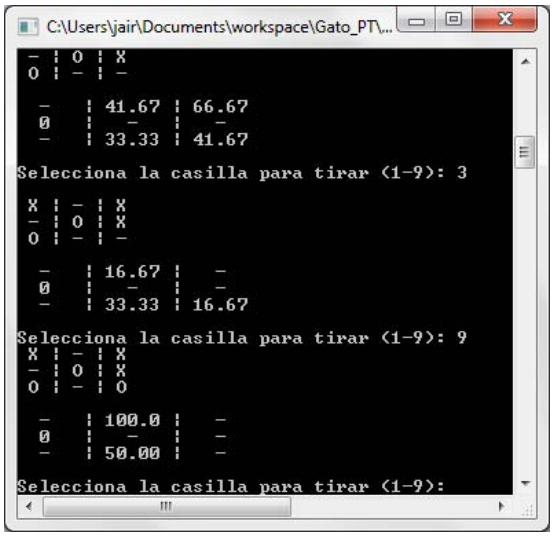
12abcdefg o 13abcdefg o 14abcdefg o 15abcdefg o 16abcdefg, etc.

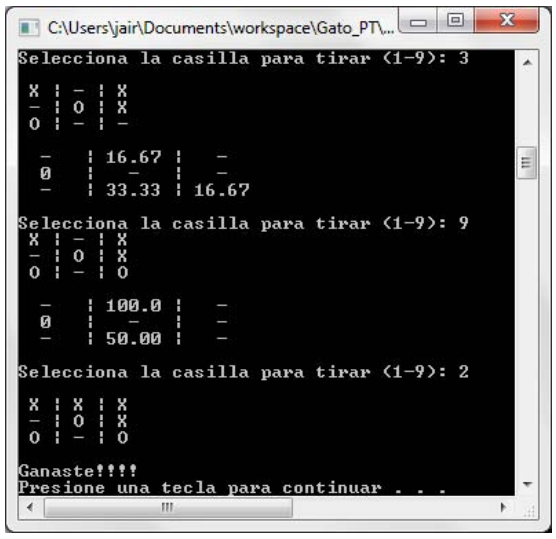
Tomando el ejemplo 15abcdefg hay 5040 combinaciones de la cuales 1872 corresponden a registros relacionados con el -1, obteniendo la probabilidad de aparación con esta información nos da 0.371429.

La selección entre 1 y -1 para obtener los resultados depende del turno que se esté manejando, para los turnos nones (1, 3, 5, 7 y 9) se ocupa el conteo de los 1's, en caso contrario (números pares) se utiliza la información obtenida de los -1's.

Esta secuencia de pasos se continua manejado según avance el juego del Gato hasta obtener una victoria o un empate.



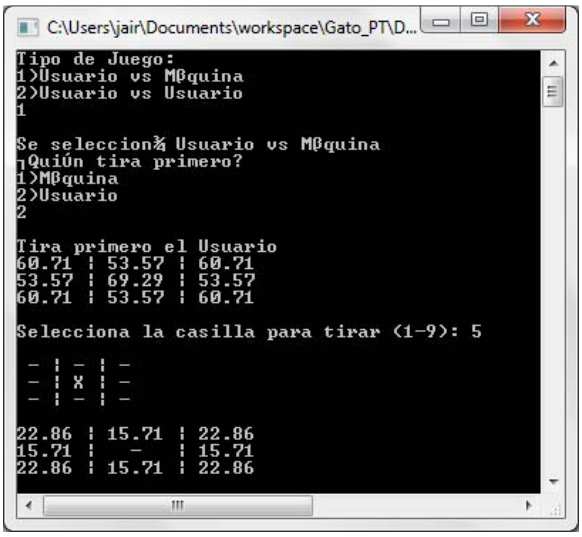


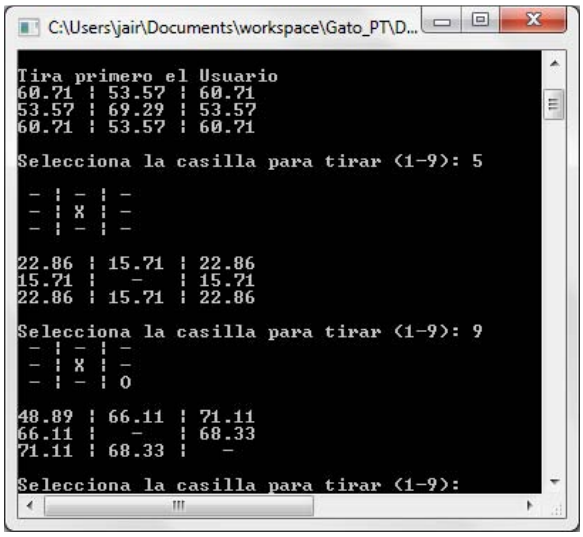


Cuando se esté jugando contra la máquina ésta tirará en las casillas que tengan un mayor porcentaje de ganar o en su defecto en donde el contrincante tenga menos posibilidades de ganar:

El cálculo de las probabilidades en cualquier instante de la ejecución de la aplicación y es el que se explicó con anterioridad en este documento.

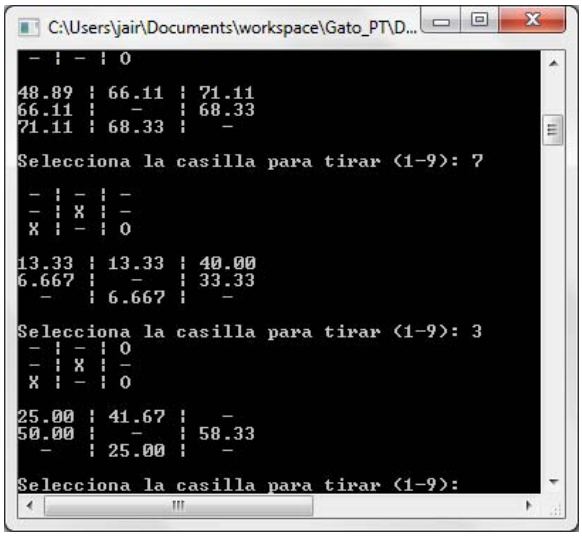
Siguiendo el flujo normal del juego, tirando/seleccionando casillas y esperando a que la máquina haga lo propio, cómo se nuestra a continuación:



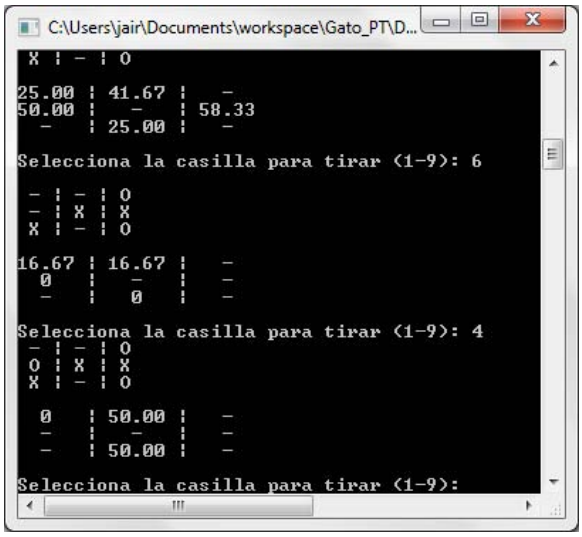


Jugamos tomando las mejores decisiones y observamos el comportamiento de la máquina al seleccionar su siguiente tirada.

Hay cierta clase de jugadas o secuencia de tiros que no llevan a tener situaciones en las que las probabilidades de ganar son mínimas, si no es que nulas, en estos casos la máquina toma la decisión de jugar a no dejarte ganar, esto es, en la imagen se muestran 4 probabilidades dos son cero y las otras son del 16.67%, es aquí cuando acudimos de nuevo al universo de combinaciones correspondientes al turno en



curso, en el ejemplo trabajado en esta secciones tenemos la siguiente secuencia de tiros: 59736 lo que nos deja con 4 número más a este listado y que conforman 24 combinaciones, para las relacionadas con el -1 las probabilidades de ganar son las exhibidas en la imagen mostrada, para las que tienen que ver con el 1 son:



- 83.3333%, para la casilla 1
- 50.0000%, para la casilla 2
- 33.3333%, para la casilla 4 y
- 66.6667%, para la casilla 8

Esta parte de algoritmo lo que hace es comparar nuestra probabilidad de ganar contra la probabilidad de perder del contrincante y si esta última es mayor la elegimos y tiramos en esa casilla.

Para describirlo de manera coloquial se podría decir que: ya que no vamos a ganar, vamos a hacer que tú tampoco ganes y por consiguiente llevar el juego a un resultado sin ganador, un empate