GUÍA DE APRENDIZAJE PARA CONCURSANTES ICPC Y IOI: TEORÍA DE JUEGOS, FUNDAMENTOS BÁSICOS

1. Introducción

La teoría de juego es rama de la matemática y la lógica que se ocupa del análisis de juegos (ejemplo: Las situaciones que implica partes con conflictos de intereses). Además de la elegancia matemática y soluciones completas para juegos simples, los principios de la teoría de juego encuentran también aplicación en juegos mas complicados como son los naipes (cartas), damas y ajedrez, así como también los problemas realmente mundiales tan diverso como la economía, la división de la propiedad, la política, y la guerra. En esta guía abordaremos los elementos esenciales de esta teoría a través de determinados juegos.

2. Conocimientos previos

2.1. Operador xor

En lógica proposicional, la disyunción exclusiva (también llamado bidisyuntor lógico, disyuntor excluyente, or fuerte, or exclusivo, o desigualdad material) es un operador lógico simbolizado como XOR. Una disyunción exclusiva solamente es verdadera cuando ambas frases tienen valores diferentes y es falsa si las dos frases son ambas verdaderas o ambas falsas.

2.2. Juego de información perfecta

Un juego de información perfecta se refiere al hecho de que cada jugador tiene la misma información que estaría disponible al final del juego. Es decir, cada jugador sabe o puede ver los movimientos de otros jugadores. Un buen ejemplo sería el ajedrez, donde cada jugador ve las piezas del otro jugador en el tablero.

2.3. Juego de información imperfecta

La información imperfecta aparece cuando las decisiones tienen que hacerse simultáneamente, y los jugadores necesitan analizar todos los posibles resultados a la hora de tomar una decisión. Un buen ejemplo de juegos de información imperfecta es un juego de cartas donde las cartas de cada jugador están escondidas del resto de los jugadores.

3. Desarrollo

Los juegos de los que hablaremos son juegos para dos personas con información perfecta, sin movimientos al azar, y un resultado de ganar o perder. En estos juegos, los jugadores suelen alternar movimientos hasta alcanzar una posición terminal. Después de eso, un jugador es declarado ganador y el otro perdedor. La mayoría de los juegos de cartas no se ajustan a esta categoría, por ejemplo, porque no tenemos información sobre qué cartas tiene nuestro oponente.

3.1. Ganar o perder

Veamos es el siguiente juego, jugado por dos jugadores que se turnan para moverse. Al principio hay n monedas. Cuando es el turno de un jugador, él puede quitar m_1, m_2, m_3, ... , m_p monedas siendo $m_1 < m_2 < m_3 < \ldots < m_p$. El jugador que se lleva el último es declarado ganador (en otras palabras, el jugador que no puede hacer un movimiento es el perdedor). La pregunta es: ¿para qué n ganará el primer jugador si ambos juegan de manera óptima ?.

Podemos ver que $n = m_1, m_2, m_3, \ldots, m_p$ son posiciones ganadoras para el primer jugador, porque simplemente puede tomar todas las monedas. Para $n = 0$ no hay movimientos posibles (el juego está terminado), por lo que es la posición perdedora para el primer jugador, ya que no puede moverse de él.

Supongamos que se puede quitar 1, 3 y 4 monedas podemos ver que n = 1, 3, 4 son posiciones ganadoras para el primer jugador, porque simplemente puede tomar todas las monedas. Para n = 0 no hay movimientos posibles (el juego está terminado), por lo que es la posición perdedora para el primer jugador, ya que no puede moverse de él. Si n = 2, el primer jugador tiene solo una opción, para eliminar 1 moneda. Si n = 5 o 6, un jugador puede moverse a 2 (eliminando 3 o 4 monedas), y está en una posición ganadora. Si n = 7, un jugador puede mover solo a 3, 4, 6, pero de todos ellos su oponente puede ganar.

Las posiciones tienen las siguientes propiedades:

- Todas las posiciones terminales están perdiendo.
- Si un jugador puede moverse a una posición perdedora, entonces está en una posición ganadora.
- Si un jugador puede moverse solo a las posiciones ganadoras, entonces está en una posición perdedora.

Estas propiedades podrían usarse para crear un algoritmo WinLose-Algoritmo recursivo simple:

```
logico esGanador(entero pos)
   //Posibles posiciones a las que puedo moverme desde la posicion pos;
   movs[]
   Por cada x que pertenece movs:
      Si esGanador(x) == Falso:
         retorno Verdadero
   retorno Falso
```

n	0	1	2	3	4	5	6	7	8	9	10	11
posición	L	W	L	W	W	W	W	L	W	L	W	W

Este juego podría jugarse también con una regla (generalmente llamada la regla de juego misere) de que el jugador que quita la última moneda es declarado perdedor. Solo necesita cambiar el comportamiento de las posiciones de terminal en el algoritmo WL. La tabla cambiará a esto:

n	0	1	2	3	4	5	6	7	8	9	10	11
posición	W	L	W	L	W	W	W	W	L	W	L	W

Se puede ver que si una posición está ganando o perdiendo depende solo de las últimas k posiciones, donde k es el número máximo de monedas que podemos quitar. Si bien solo hay 2^k valores posibles para las secuencias de la longitud k, nuestra secuencia se volverá periódica.

3.2. El juego del Nim

El juego matemático más famoso es probablemente el Juego de Nim. Este es el juego que probablemente encontrarás más veces y hay muchas variaciones en él, así como juegos que se pueden resolver utilizando el conocimiento de cómo jugar el juego. Aunque estos problemas a menudo requieren una idea inteligente, generalmente son muy fáciles de codificar.

Históricamente, este juego fue popular en la antigüedad. Su origen probablemente esté en China, o al menos el juego Jianshizi es muy similar a él. En Europa las primeras referencias son del siglo XVI. El nombre se lo dio Charles Bouton, quien en 1901 publicó un análisis completo de este juego.

El juego clásico plantea que: Tenemos una pila con n piedras, y los jugadores pueden tomar de dicha pila, desde 1 hasta n piedras. El jugador que en su turno, no pueda retirar más piedras del bulto, se considera perdedor.

Está claro que siempre gana el jugador A, retirando todas las piedras, es decir, el jugador B alcanza un estado terminal que es un bulto con 0 piedras, por lo que podemos definir que la posición p = 0 es perdedora y que cualquier posición p >0 es ganadora, porque podemos retirar todas las piedras y ganar.

Pero hagamos el juego más interesante, supongamos que cada jugador puede remover una o dos piedras solamente. La posición p = 0 sigue siendo una posición perdedora porque no podemos retirar ninguna piedra, fácilmente podemos ver que las posiciones p = 1 y p = 2 son ganadoras, simplemente retiramos todas las piedras. ¿Pero, qué sucedería si el bulto tiene 3 piedras?

De la posición p= 3 nos podemos mover a p = 2 retirando una piedra, y a p = 1 retirando dos piedras, ambas posiciones ganadoras para el próximo jugador a jugar, por lo tanto, p = 3 es perdedora.

Y si el bulto tuviera 4 piedras, ¿Cuántas piedras se podrían quitar?, ¿una o dos?. Si quitamos dos alcanzamos la posición p= 2 que es ganadora para el próximo jugador, y si quitamos una alcanzamos la posición p= 3 que es perdedora para el próximo jugador, así que la jugada óptima es quitar una sola piedra.

Entonces ya podemos definir dos conceptos importantes:

Posición ganadora (G): Una posición es ganadora, si de esta podemos movernos al menos a una posición perdedora.

Figura 1: Juego del Nim

Posición perdedora (P): Una posición es perdedora si de esta solo podemos movernos a posiciones ganadoras. Todos los estados terminales son posiciones perdedoras.

n	0	1	2	3	4	5	6	7	8	9
posición	P	G	G	P	G	G	P	G	G	P

En esta tabla se puede ver el patrón PGG, y llegar a la siguiente conclusión:

Definiendo P(n) como la posición del juego en el estado n.

P(n) = P (perdedora) si n es divisible por 3.

P(n) = G (ganadora) si n no es divisible por 3.

Variante del juego de Nim: Hay n pilas de monedas. Cuando es el turno de un jugador, él elige una pila y toma al menos una moneda de ella. Si alguien no puede moverse, pierde (por lo tanto, el que saca la última moneda es el ganador).

El estado del juego se describe sin ambigüedades mediante un conjunto múltiple de números enteros positivos. Un movimiento consiste en disminuir estrictamente un número entero elegido (si se convierte en cero, se elimina del conjunto).

La solución de Charles L. Bouton se ve así:

Teorema: El jugador actual tiene una estrategia ganadora si y solo si la suma xor de los tamaños de pila es distinta de cero. La suma xor de una sucesión a es $a_1 \oplus a_2 \oplus \ldots \oplus a_n$, dónde $\oplus$ es el bit a bit exclusivo o.

Prueba. La clave de la prueba es la presencia de una estrategia simétrica para el oponente . Mostramos que una vez en una posición con la suma xor igual a cero, el jugador no podrá hacer que no sea cero a largo plazo, si hace la transición a una posición con una suma xor distinta de cero, el oponente siempre tendrá un movimiento que devuelve la suma xor a cero.

Probaremos el teorema por inducción matemática.

Para un Nim vacío (donde todas las pilas están vacías, es decir, el conjunto múltiple está vacío), la suma xor es cero y el teorema es verdadero.

Ahora supongamos que estamos en un estado no vacío. Usando el supuesto de inducción (y la aciclicidad del juego) asumimos que el teorema está probado para todos los estados alcanzables desde el actual.

Entonces la demostración se divide en dos partes: si para la posición actual el xor-sum $s = 0$, tenemos que demostrar que este estado está perdiendo, es decir, todos los estados alcanzables

tienen xor-sum $t \neq 0$. Si $s \neq 0$, tenemos que probar que hay un movimiento que conduce a un estado con $t = 0$.

- Dejar $s = 0$ y consideremos cualquier movimiento. Este movimiento reduce el tamaño de una pila x a un tamaño y. Usando propiedades elementales de $\oplus$, tenemos

$$t = s \oplus x \oplus y = 0 \oplus x \oplus y = x \oplus y$$

  Desde $y < x$,$y \oplus x$ no puede ser cero, entonces $t \neq 0$. Eso significa que cualquier estado alcanzable es ganador (por la suposición de inducción), por lo que estamos en una posición perdedora.

- Dejar $s \neq 0$ considere la representación binaria del número s. Dejar d sea el índice de su bit principal (mayor valor) distino de cero. Nuestro movimiento será en una pila cuyo tamaño es el número de bit d está establecido (debe existir, de lo contrario, el bit no se establecería en s). Reduciremos su tamaño x a $y = x \oplus s$. Todos los bits en posiciones mayores que d en x y y emparejar y morder d está preparado x pero no establecido y. Por lo tanto, $y < x$, que es todo lo que necesitamos para que una mudanza sea legal. Ahora tenemos:

$$t = s \oplus x \oplus y = s \oplus x \oplus (s \oplus x) = 0$$

  Esto significa que encontramos un estado perdedor alcanzable (por la suposición de inducción) y el estado actual es ganador.

Corolario. Cualquier estado de Nim puede ser reemplazado por un estado equivalente siempre que la suma xor no cambie. Además, al analizar un Nim con varias pilas, podemos reemplazarlo con una sola pila de tamaño s.

4. Implementación

4.1. C++

4.1.1. Ganar o perder

El la siguiente implementación la función devolverá verdadero siempre y cuando el jugador que juegue primero tenga una estrategia ganadora y falso en caso contrario.

```
bool winOrLoss(int K, int *moves) {
   bool *isWinnig = new bool[K + 1];
   fill(isWinnig, isWinnig + (K + 1), false);
   for (int i = 1; i <= K; i++) {
      for (int j = 0; j < sizeof(moves) / sizeof(moves[0]); j++) {
         if (i - moves[j] >= 0 && isWinnig[i - moves[j]] == false) {
            isWinnig[i] = true; break;
                        }
               }
```

```
        }
        return isWinnig[K] == true;
}
```

4.1.2. Juego del Nim

El la siguiente implementación la función devolverá verdadero siempre y cuando el jugador que juegue primero tenga una estrategia ganadora y falso en caso contrario.

```
bool nim(vector<int> stacks) {
   int sums=0;
   if(stacks.empty()==false) {
      sums = stacks[0];
      for(int i=1;i<stacks.size();i++) sums = sums ^ stacks[0];
   }
   return (sums!=0);
}
```

4.2. Java

4.2.1. Ganar o perder

El la siguiente implementación la función devolverá verdadero siempre y cuando el jugador que juegue primero tenga una estrategia ganadora y falso en caso contrario.

```
public boolean winOrLoss(int K,int [] moves) {
   boolean [] isWinnig = new boolean[K+1];
   Arrays.fill(isWinnig, false);
   for(int i=1;i<=K;i++) {
      for(int j=0;j<moves.length;j++) {
         if(i-moves[j]>=0 && isWinnig[i-moves[j]]==false) {
            isWinnig[i]=true;break;
         }
      }
   }
   return isWinnig[K]==true;
}
```

4.2.2. Juego del Nim

El la siguiente implementación la función devolverá verdadero siempre y cuando el jugador que juegue primero tenga una estrategia ganadora y falso en caso contrario.

```
public boolean nim(List<Integer> stacks) {
   int sums=0;
   if(stacks.isEmpty()==false) {
```

```
        sums = stacks.get(0);
        for(int i=1;i<stacks.size();i++) sums = sums ^ stacks.get(i);
    }
    return (sums!=0);
}
```

5. Aplicaciones

La teoría de juegos tiene una amplia gama de aplicaciones, que incluyen psicología, biología evolutiva, guerra, política, economía y negocios. A pesar de sus muchos avances, la teoría de juegos sigue siendo una ciencia joven y en desarrollo.

Como método de matemática aplicada, la teoría de juegos se ha utilizado para estudiar una amplia variedad de comportamientos humanos y animales. Inicialmente se desarrolló en economía para comprender una gran colección de comportamientos económicos, incluidos los comportamientos de empresas, mercados y consumidores. El primer uso del análisis de teoría de juegos fue por Antoine Augustin Cournot en 1838 con su solución del duopolio de Cournot. El uso de la teoría de juegos en las ciencias sociales se ha expandido, y la teoría de juegos también se ha aplicado a los comportamientos políticos, sociológicos y psicológicos.

Aunque los naturalistas anteriores al siglo XX, como Charles Darwin, hicieron declaraciones de tipo teórico de juegos, el uso del análisis teórico de juegos en biología comenzó con los estudios de Ronald Fisher sobre el comportamiento animal durante la década de 1930. Este trabajo es anterior al nombre de «teoría de juegos», pero comparte muchas características importantes con este campo. Los desarrollos en economía fueron aplicados más tarde a la biología en gran parte por John Maynard Smith en su libro Evolution and the Theory of Games.

Además de usarse para describir, predecir y explicar el comportamiento, la teoría de juegos también se ha utilizado para desarrollar teorías del comportamiento ético o normativo y para prescribir dicho comportamiento. En economía y filosofía, los académicos han aplicado la teoría de juegos para ayudar a comprender el comportamiento bueno o apropiado. Los argumentos teóricos del juego de este tipo se pueden encontrar desde Platón. Una versión alternativa de la teoría de juegos, llamada teoría de juegos químicos, representa las elecciones del jugador como moléculas reactivas químicas metafóricas llamadas «knowlecules». La teoría del juego químico luego calcula los resultados como soluciones de equilibrio para un sistema de reacciones químicas.

6. Complejidad

No se preocupe si ve un problema de teoría de juego durante un concurso: puede ser similar a uno de los juegos descritos anteriormente o puede reducirse a uno de ellos. Si no, solo piénselo en ejemplos concretos. Una vez que lo descubres, la parte de codificación suele ser muy simple, directa y con una complejidad bastante asequible para la restricciones.

En el caso de la implementación de ganar o perder su complejidad es $O(nk)$ donde n es la cantidad inicial y k la cantidad de posibles movimientos. En el caso de la implementación del

Nim su complejidad es O(n) donde n es la cantidad de pilas iniciales del juego.

7. Ejercicios propuestos

A continuacion una lista de ejercicios que se resuelven aplicando estas nociones basicas de teoría de juego:

- DMOJ - Juego AB
- DMOJ - Un juego interesante
- DMOJ - Juego de Dígitos Vacuno
- DMOJ - Piedras