GUÍA DE APRENDIZAJE PARA CONCURSANTES ICPC Y IOI: BÚSQUEDA TERNARIA

1. Introducción

Se nos da una función $f(x)$ que es **unimodal** en un intervalo $[l, r]$. En esta guía veremos como hallar el máximo o mínimo valor según sea la situación dentro del intervalo $[l, r]$

2. Conocimientos previos

2.1. Función unimodal

Una función unimodal es una función matemática que tiene un único máximo o mínimo absoluto en un intervalo determinado. Esto significa que la función aumenta o disminuye hasta alcanzar un punto máximo o mínimo y luego vuelve a cambiar de dirección. En otras palabras, la función tiene una forma de *campana* con un solo pico o valle. Por función unimodal, nos referimos a uno de dos comportamientos de la función:

1. La función primero aumenta estrictamente, alcanza un máximo (en un solo punto o en un intervalo) y luego disminuye estrictamente.
2. La función primero disminuye estrictamente, alcanza un mínimo y luego aumenta estrictamente.

2.2. Teorema de Maestro

En el análisis de algoritmos , el teorema maestro de las recurrencias de divide y vencerás proporciona un análisis asintótico para muchas relaciones de recurrencia que ocurren en el análisis de los algoritmos de divide y vencerás . El enfoque fue presentado por primera vez por Jon Bentley , Dorothea Blostein (de soltera Haken) y James B. Saxe en 1980, donde se describió como un método unificador para resolver tales recurrencias. El nombre *teorema maestro* fue popularizado por el libro de texto de algoritmos ampliamente utilizado Introducción a los algoritmos de Cormen , Leiserson , Rivest y Stein .

3. Desarrollo

Considere 2 puntos cualesquiera m_1, y m_2 en este intervalo: $l < m_1 < m_2 < r$. Evaluamos la función en m_1 y m_2, es decir, encontrar los valores de $f(m_1)$ y $f(m_2)$. Ahora, tenemos una de tres opciones:

1. $f(m_1) < f(m_2)$ El máximo deseado no se puede ubicar en el lado izquierdo de m_1, es decir, en el intervalo $[l, m_1]$, ya que ambos puntos m_1 y m_2 o solo m_1 pertenecen al área donde la función aumenta. En cualquier caso, esto significa que tenemos que buscar el máximo en el intervalo $[m_1, r]$.
2. $f(m_1) > f(m_2)$ Esta situación es simétrica a la anterior: el máximo no puede ubicarse en el lado derecho de m_2, es decir, en el intervalo $[m_2, r]$, y el espacio de búsqueda se reduce al segmento $[l, m_2]$.

3. $f(m_1) = f(m_2)$ Podemos ver que ambos puntos pertenecen al área donde se maximiza el valor de la función, o m_1 está en la zona de valores crecientes y m_2 está en el área de valores descendentes (aquí usamos el rigor de la función creciente/decreciente). Así, el espacio de búsqueda se reduce a $[m_1, m_2]$. Para simplificar el código, este caso se puede combinar con cualquiera de los casos anteriores.

Por lo tanto, basándose en la comparación de los valores en los dos puntos internos, podemos reemplazar el intervalo actual $[l, r]$ con un nuevo intervalo más corto $[l', r']$. Aplicando repetidamente el procedimiento descrito al intervalo, podemos obtener un intervalo arbitrariamente corto. Con el tiempo, su longitud será menor que una determinada constante predefinida (precisión) y el proceso podrá detenerse. Este es un método numérico, por lo que podemos suponer que después de eso la función alcanza su máximo en todos los puntos del último intervalo $[l, r]$. Sin pérdida de generalidad, podemos tomar $f(l)$ como valor de retorno.

No impusimos ninguna restricción en la elección de puntos m_1 y m_2. Esta elección definirá la tasa de convergencia y la precisión de la implementación. La forma más común es elegir los puntos para que divida el intervalo $[l, r]$ en tres partes iguales. Así, tenemos:

$$m_1 = l + \frac{(r-l)}{3}$$

$$m_2 = r - \frac{(r-l)}{3}$$

Si m_1 y m_2 se eligen para que estén más cerca entre sí, la tasa de convergencia aumentará ligeramente.

3.1. El caso de los argumentos enteros

Si $f(x)$ toma un parámetro entero, el intervalo $[l, r]$ se vuelve discreto. Dado que no impusimos ninguna restricción en la elección de puntos m_1 y m_2, la exactitud del algoritmo no se ve afectada m_1 y m_2 todavía se puede elegir dividir $[l, r]$ en 3 partes aproximadamente iguales.

La diferencia se produce en el criterio de parada del algoritmo. La búsqueda ternaria tendrá que detenerse cuando $(r - l) < 3$, porque en ese caso ya no podemos seleccionar m_1 y m_2 ser diferentes entre sí y también de l y r, y esto puede causar un bucle infinito. Una vez $(r - l) < 3$, el grupo restante de puntos candidatos $(l, l + 1, \ldots, r)$. Es necesario comprobarlo para encontrar el punto que produce el valor máximo $f(x)$.

4. Implementación

4.1. C++

```
double f (double x){
   //Evaluacion de x en la funcion
```

```
}

double ternary_search(double l, double r) {
   double eps = 1e-9; //establezca el limite de error aqui
   while (r - l > eps) {
      double m1 = l + (r - l) / 3;
      double m2 = r - (r - l) / 3;
      double f1 = f(m1); //evalua la funcion en m1
      double f2 = f(m2); //evalua la funcion en m2
      if (f1 < f2) l = m1;
      else r = m2;
   }
   return f(l); //devolver el maximo de f(x) en [l, r]
}
```

4.2. Java

```
public static double f (double x){
   //Evaluacion de x en la funcion
}

public static double ternary_search(double l, double r) {
   double eps = 1e-9; //establezca el limite de error aqui
   while (r - l > eps) {
      double m1 = l + (r - l) / 3;
      double m2 = r - (r - l) / 3;
      double f1 = f(m1); //evalua la funcion en m1
      double f2 = f(m2); //evalua la funcion en m2
      if (f1 < f2) l = m1;
      else r = m2;
   }
   return f(l); //devolver el maximo de f(x) en [l, r]
}
```

5. Aplicaciones

A continuación, se presentan algunas aplicaciones comunes de la búsqueda ternaria:

- **Optimización en funciones unimodales:** La búsqueda ternaria se utiliza para encontrar el máximo o mínimo de una función unimodal (una función que tiene un solo máximo o mínimo) en un intervalo dado. Dividiendo el intervalo en tres partes y comparando los valores de la función en los puntos intermedios, se puede encontrar el punto óptimo con mayor precisión que con la búsqueda binaria.
- **Búsqueda en funciones cóncavas o convexas:** En funciones cóncavas o convexas, la búsqueda ternaria puede utilizarse para encontrar el punto en el que la función alcanza su máximo

o mínimo. Dividiendo el intervalo en tres partes y determinando en qué dirección se encuentra el punto óptimo, se puede reducir el espacio de búsqueda de manera eficiente.

- **Determinación de umbrales:** La búsqueda ternaria también se utiliza para determinar umbrales o puntos críticos en una función. Por ejemplo, si se desea encontrar el punto en el que una función supera cierto umbral, la búsqueda ternaria puede ser útil para acercarse a ese valor de manera eficiente.
- **Problemas de optimización numérica:** En problemas de optimización numérica donde se busca maximizar o minimizar una función en un intervalo, la búsqueda ternaria puede ser una herramienta útil para encontrar soluciones cercanas al óptimo.

En resumen, la búsqueda ternaria es una técnica de búsqueda eficiente que puede aplicarse en una variedad de situaciones donde se necesita encontrar el máximo o mínimo de una función, determinar umbrales o puntos críticos, o resolver problemas de optimización numérica. Su capacidad para dividir el espacio de búsqueda en tres partes puede conducir a una convergencia más rápida y precisa en comparación con otros métodos de búsqueda.

6. Complejidad

Se puede visualizar de la siguiente manera: cada vez que se evalúa la función en los puntos m_1 y m_2, esencialmente estamos ignorando aproximadamente un tercio del intervalo, ya sea el izquierdo o el derecho. Por tanto, el tamaño del espacio de búsqueda es $2n/3$ del original.

Aplicando el teorema de Maestro , obtenemos la estimación de complejidad deseada:

$$O(n) = O(2n/3) + O(1) = O(\log n)$$

7. Ejercicios

A continuación una lista de ejercicios que se pueden resolver aplicando este algoritmo:

- Codeforces - Devu and his Brother
- Codeforces - Restorer Distance
- Codeforces - Weakness and Poorness
- Codechef - Race time
- Codechef - Is This JEE
- TIMUS 1719 Kill the Shaitan-Boss
- TIMUS 1913 Titan Ruins: Alignment of Forces
- Hackerearth - Rescuer
- SPOJ - Building Construction

- GYM - Dome of Circus (D)
- GYM - Chasing the Cheetahs (A)
- UVA - Galactic Taxes
- UVA - 12197 - Trick or Treat
- LOJ - Closest Distance