**Бинарный (двоичный) поиск**

Используется для поиска определенного значения в монотонной функции.

Монотонная функция – либо только возрастает, либо только убывает.

Примеры монотонной функции:

* отсортированные массивы;
* какие-либо математические выражения, значения которых будут только возрастать или только убывать на всём диапазоне поиска.

Могут пригодиться в основном 5 видов бинпоиска:

Точный – когда в диапазоне поиска находим в точности искомое значение.

Левосторонний включающий (ЛВ) – когда ищем значение, которое МЕНЬШЕ искомого, но максимально близко к нему (если искомого значения нет в диапазоне поиска), либо в точности искомое значение (если оно есть в диапазоне поиска).

Левосторонний исключающий (ЛИ) – когда ищем значение, которое МЕНЬШЕ искомого, но максимально близко к нему, и при этом ОБЯЗАТЕЛЬНО НЕ РАВНО искомому.

Правосторонний включающий (ПВ) – когда ищем значение, которое БОЛЬШЕ искомого, но максимально близко к нему (если искомого значения нет в диапазоне поиска), либо в точности искомое значение (если оно есть в диапазоне поиска).

Правосторонний исключающий (ПИ) – когда ищем значение, которое БОЛЬШЕ искомого, но максимально близко к нему, и при этом ОБЯЗАТЕЛЬНО НЕ РАВНО искомому.

Примеры результатов:

|  |  |
| --- | --- |
| **Массив a = {2, 5, 17, 84, 136, 1000} (индексация с 0)** | **Функция f(x) = 720/x, где x - целое число от 1 до 6 включительно** |
| точный(5) = 1 (a[1] = 5) | точный(240) = 3 (f(3) = 720/3 = 240) |
| ЛВ(85) = 3 (a[3] = 84, 84 < 85) | ЛВ(150) = 5 (f(5) = 720/5 = 144, 144 < 150) |
| ЛВ(84) = 3 (a[3] = 84, 84 = 84) | ЛВ(144) = 5 (f(5) = 720/5 = 144, 144 = 144) |
| ЛИ(85) = 3 (a[3] = 84, 84 < 85) | ЛИ(150) = 5 (f(5) = 720/5 = 144, 144 < 150) |
| ЛИ(84) = 2 (a[2] = 17, 17 < 85) | ЛИ(144) = 6 (f(6) = 720/6 = 120, 120 < 144) |
| ПВ(100) = 4 (a[4] = 136, 136 > 100) | ПВ(310) = 2 (f(2) = 720/2 = 360, 360 > 310) |
| ПВ(136) = 4 (a[4] = 136, 136 = 136) | ПВ(360) = 2 (f(2) = 720/5 = 360, 360 = 360) |
| ПИ(100) = 4 (a[4] = 136, 136 > 100) | ПИ(310) = 2 (f(2) = 720/2 = 360, 360 > 310) |
| ПИ(136) = 5 (a[5] = 1000, 1000 > 136) | ПИ(360) = 1 (f(1) = 720/1 = 720, 720 > 360) |

Реализации для поиска ПО ВОЗРАСТАЮЩИМ ФУНКЦИЯМ (для поиска по убывающим функциям необходимо в цикле «while» заменить все строки «low = mid + 1» на «high = mid - 1» и наоборот) представлены ниже. Условимся, что диапазон поиска у нас от start до finish включительно, вместо type используется либо int, либо long long в зависимости от задачи, f(mid) – либо элемент массива с индексом mid, либо какая-то функция, вычисленная по значению mid. Ищем мы значение key.

1. Точный.

type low = start, high = finish, mid, binSearchResult = start - 1;

while (low <= high)

{

mid = low + (high - low + 1) / 2;

if (f(mid) < key)

{

low = mid + 1;

}

else if (f(mid) > key)

{

high = mid - 1;

}

else

{

binSearchResult = mid;

break;

}

}

if (binSearchResult != start - 1)

{

// что делать, если нашли key

}

else

{

// key не нашли

}

1. Левосторонний включающий

type low = start, high = finish, mid, binSearchResult = start - 1;

while (low <= high)

{

mid = low + (high - low + 1) / 2;

if (f(mid) < key)

{

low = mid + 1;

binSearchResult = mid;

}

else if (f(mid) > key)

{

high = mid - 1;

}

else

{

binSearchResult = mid;

break;

}

}

if (binSearchResult != start - 1)

{

// нашёлся элемент, равный key или меньше него, но максимально близкий

}

else

{

// абсолютно все элементы больше key

}

1. Левосторонний исключающий

type low = start, high = finish, mid, binSearchResult = start - 1;

while (low <= high)

{

mid = low + (high - low + 1) / 2;

if (f(mid) < key)

{

low = mid + 1;

binSearchResult = mid;

}

else if (f(mid) >= key)

{

high = mid - 1;

}

}

if (binSearchResult != start - 1)

{

// нашёлся элемент меньше key, но максимально близкий

}

else

{

// все элементы больше key (некоторые могут быть и равны key)

}

1. Правосторонний включающий

type low = start, high = finish, mid, binSearchResult = start - 1;

while (low <= high)

{

mid = low + (high - low + 1) / 2;

if (f(mid) < key)

{

low = mid + 1;

}

else if (f(mid) > key)

{

high = mid - 1;

binSearchResult = mid;

}

else

{

binSearchResult = mid;

break;

}

}

if (binSearchResult != start - 1)

{

// нашёлся элемент, равный key или больше него, но максимально близкий

}

else

{

// абсолютно все элементы меньше key

}

1. Правосторонний исключающий

type low = start, high = finish, mid, binSearchResult = start - 1;

while (low <= high)

{

mid = low + (high - low + 1) / 2;

if (f(mid) <= key)

{

low = mid + 1;

}

else if (f(mid) > key)

{

high = mid - 1;

binSearchResult = mid;

}

}

if (binSearchResult != start - 1)

{

// нашёлся элемент больше key, но максимально близкий

}

else

{

// все элементы меньше key (некоторые могут быть и равны key)

}