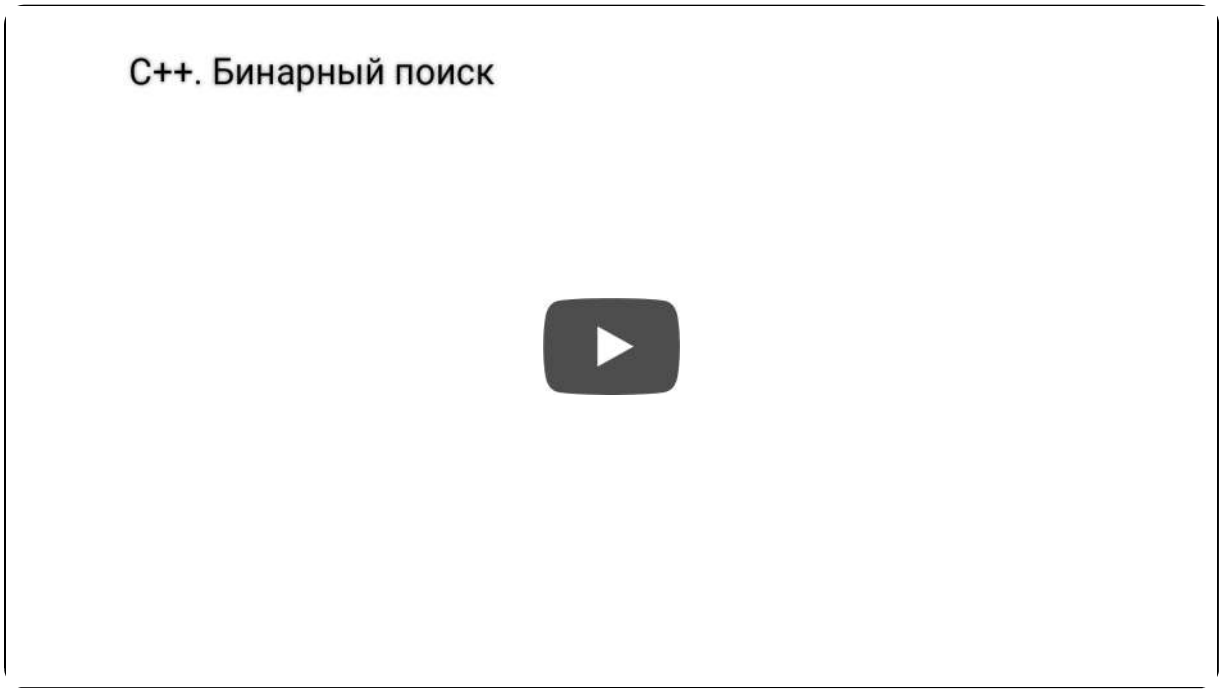
Бинарный поиск



Бинарный поиск

Бинарный поиск — это метод решения задач, который основан на делении пополам.

Одним из примеров бинарного поиска служит поиск слова в словаре. Пусть у нас есть англо-русский словарь и требуется найти перевод какого-то слова. Откроем словарь где-то в середине и сравним слово. написанное на открытой странице, с искомым словом. Если это слово идёт лексикографически (в алфавитном порядке) раньше, чем искомое слово, то искомое слово надо искать во второй половине словаря, а иначе его надо искать в первой половине словаря. Таким образом, за один шаг мы сократили диапазон поиска с целого словаря до половины словаря. Аналогично поступим на втором шаге — откроем такую страницу словаря, которая делит половину словаря, в которой находится искомое слово, на две равные части, и сравним написанное там слово с искомым словом. После второго шага мы сократим диапазон поиска до четверти словаря. Далее будем поступать аналогичным образом, каждый раз сокращая искомый диапазон страниц в два раза, пока он не сократится до одной страницы, на которой находится искомое слово.

Так как каждый раз диапазон поиска сокращается в два раза, то искомая

страница будет найдена за $O(\log n)$ шагов. Стоит отметить, что если на каждом шаге середина будет найдена не совсем точно, то диапазон поиска может сокращаться менее чем в два раза за один шаг, но асимптотика от этого не изменится.

Другой пример применения бинарного поиска — это игра в угадывание числа. Пусть нам загадали натуральное число от 1 до N. Мы должны отгадать это число, задавая вопросы вида: "Верно ли, что загаданное число меньше X?", где X — некоторое натуральное число. Желательно использовать как можно меньше вопросов. Используя бинарный поиск, можно отгадать число за $O(\log n)$ вопросов. Для этого сначала зададим вопрос для $X \approx \frac{N}{2}$, чтобы вне зависимости от полученного ответа сократить диапазон поиска примерно в два раза. И будем поступать так на каждом шаге. Если на текущий момент мы знаем, что наше число находится в диапазоне от L до R, то будем задавать вопрос для $X \approx \frac{R+L}{2}$.

Например, если $N = 16$ и нам загадали число 10, то мы сначала спросим: "Верно ли, что загаданное число меньше 9?". Нам ответят, что это неверно, и наш диапазон поиска сократиться до чисел от 9 до 16. Затем мы спросим: "Верно ли, что загаданное число меньше 13?". Нам ответят, что это верно, и диапазон поиска сократится до чисел от 9 до 12. Затем мы спросим: "Верно ли, что загаданное число меньше 11?". Нам ответят, что это верно, и диапазон поиска сократится до чисел от 9 до 10. Наконец, мы спросим: "Верно ли, что загаданное число меньше 10?". Нам ответят, что это неверно, и мы узнаем, что загаданное число равно 10.

Чтобы написать бинарный поиск для решения задачи, будем опираться на подход, который использует понятие *инварианта*. Любая задача на бинпоиск сводится к тому, что некоторое условие выполнено для всех натуральных (или целых) индексов до какого-то значения, а после этого значения условие не выполнено для всех индексов. И задача состоит в том, чтобы найти либо последний индекс, для которого условие выполнено, либо первый индекс, для которого условие не выполнено. Сначала необходимо найти целые индексы L и R такие, что для индекса L условие выполнено, а для индекса R условие не выполнено — это правило для индексов L и R сохраняется на всех шагах алгоритма и называется *инвариантом*. Далее на каждом шаге алгоритма бинарного поиска мы будем брать число в середине отрезка $[L, R]$, которое

можно вычислить по формуле $M = \left\lfloor \frac{R+L}{2} \right\rfloor$. Если условие для индекса M выполнено, то мы можем левую границу бинарного поиска сдвинуть в M, тем самым перейдя от отрезка $[L, R]$ к отрезку $[M, R]$. Если же условие для индекса M не выполнено, то мы можем правую границу бинарного поиска сдвинуть в M, тем самым перейдя от отрезка $[L, R]$ к отрезку $[L, M]$. В любом случае наш отрезок сократится примерно в два раза, что нам и требуется. Алгоритм бинарного поиска будет выполняться до тех пор, пока границы поиска L и R не станут двумя соседними числами. А так как при выполнении нашего алгоритма мы сохраняли *инвариант*, то на последнем шаге индекс L будет последним индексом, для которого условие выполнено, а индекс R будет первым индексом, для которого условие не выполнено.

Рассмотрим типичную реализацию алгоритма бинарного поиска на языке C++. Пусть некоторое условие выполнено для L = 0 и не выполнено для R = 100. Тогда код бинарного поиска можно реализовать следующим образом.

```
int L = 0;
int R = 100;
while (R - L > 1) {
    int M = (R + L) / 2;
    if (ok(M))
        L = M;
    else
        R = M;
}
```

При этом где-то выше должна быть реализована функция `ok`, которая получает целый индекс и возвращает `true`, если условие задачи для этого индекса выполнено, и `false` в противном случае.