НИУ ВШЭ - Нижний Новгород

November 29, 2024

Алгоритмы и структуры данных

Лекция 4. Алгоритмы поиска

Илья Сергеевич Бычков
ibychkov@hse.ru

Лекция 4.
Алгоритмы поиска

Алгоритмы поиска

План лекции

0. План лекции
1. Задача поиска
2. Линейный поиск
3. Двоичный/бинарный поиск
4. Троичный/тернарный поиск
5. Экспоненциальный поиск
6. Интерполяционный поиск
7. Jump поиск

Задача поиска

Задача поиска заключается в нахождении объектов с заданными свойствами среди конечного количества объектов.

Входные данные: коллекция/контейнер с объектами, искомый критерий (функция/предикат)

Выходные данные: информация о местоположении искомых объектов в коллекции (обычно индекс или набор индексов)

Задача поиска является самой фундаментальной задачей в области компьютерных наук.

Алгоритмы поиска

План лекции

0. План лекции
1. Задача поиска
2. Линейный поиск
3. Двоичный/бинарный поиск
4. Троичный/тернарный поиск
5. Экспоненциальный поиск
6. Интерполяционный поиск
7. Jump поиск

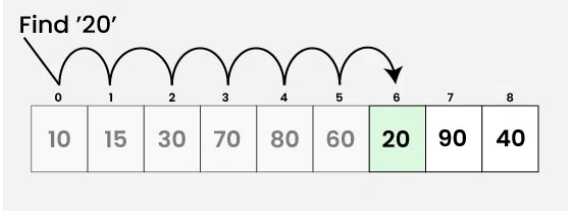
Линейный поиск

Линейный поиск

Линейный поиск (linear search) — простейший алгоритм поиска (полный перебор).

Заключается в последовательной проверке каждого элемента коллекции на соответствие заданному критерию поиска.

Алгоритм выполняется пока не будет найдено совпадение(или все совпадения) или не будут проверены все элементы.



(1) Линейный поиск (linear search), Источник - Geeks4Geeks

Линейный поиск (linear search)

Преимущества:
+ крайне прост в реализации
+ может работать без какой-либо дополнительной информации о данных

Недостатки:
- линейная сложность $\mathcal{O}(n)$ по времени в худшем и среднем случаях

Алгоритм использует константный размер дополнительной памяти $\mathcal{O}(1)$.
Однако, это является общим свойством для большинства алгоритмов поиска.

Алгоритмы поиска

План лекции

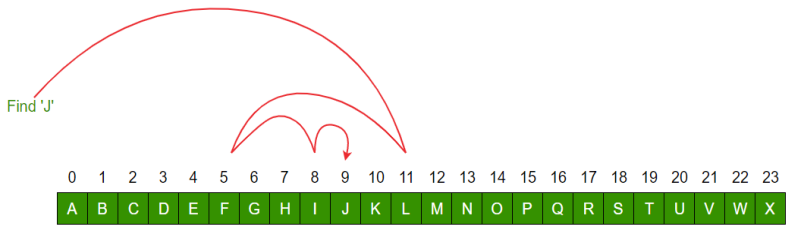
0. План лекции
1. Задача поиска
2. Линейный поиск
3. Двоичный/бинарный поиск
4. Троичный/тернарный поиск
5. Экспоненциальный поиск
6. Интерполяционный поиск
7. Jump поиск

Двоичный/бинарный поиск

Двоичный/бинарный поиск

Двоичный/бинарный поиск (binary search) — классический алгоритм поиска элемента в отсортированном массиве.

Делит текущий рассматриваемый интервал альтернатив на 2 половины. В зависимости от значения элемента в середине интервала рекурсивно решает такую же подзадачу для левой половины или для правой.



(2) Двоичный поиск (binary search), Источник - Geeks4Geeks

Двоичный/бинарный поиск (binary search)

```c
int binarySearch(int arr[], int low, int high, int x)
{
    while (low <= high) {
        int mid = low + (high - low) / 2;

        // Check if x is present at mid
        if (arr[mid] == x)
            return mid;

        // If x greater, ignore left half
        if (arr[mid] < x)
            low = mid + 1;

        // If x is smaller, ignore right half
        else
            high = mid - 1;
    }

    return -1;
}
```

(3) Реализация - Geeks4Geeks

```c
int binarySearch(int arr[], int low, int high, int x)
{
    if (high >= low) {
        int mid = low + (high - low) / 2;

        // If the element is present at the middle
        // itself
        if (arr[mid] == x)
            return mid;

        // If element is smaller than mid, then
        // it can only be present in left subarray
        if (arr[mid] > x)
            return binarySearch(arr, low, mid - 1, x);

        // Else the element can only be present
        // in right subarray
        return binarySearch(arr, mid + 1, high, x);
    }

    return -1;
}
```

(4) Реализация - Geeks4Geeks

Двоичный/бинарный поиск

Двоичный/бинарный поиск

Преимущества:

+ наиболее эффективный алгоритм для поиска
+ сложность по времени $\mathcal{O}(log_2 n)$
+ сложность по памяти $\mathcal{O}(1)$ в случае итеративной релизации

Недостатки:

- сложность в реализации
- требование к данным - предварительная сортировка
- требование к коллекции - эффективная операция произвольного доступа

```
l = -1
r = n
while r - l > 1:
    m = (r + l) // 2
    if a[m] < x:
        l = m
    else:
        r = m
```

Реализация - Algoprog

Задача

В лабиринте есть n комнат. В каждой комнате есть монстр, сила которого равна a_i. Игрок может использовать магию, чтобы уничтожить монстра силой P, но она действует только на монстров силой $\leq P$. Найти минимальную силу P, чтобы за одну атаку уничтожить как минимум k монстров.

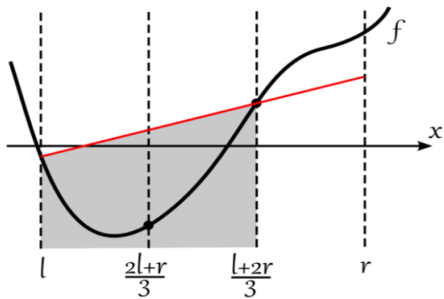
(5) Известный мем

Алгоритмы поиска

План лекции

0. План лекции
1. Задача поиска
2. Линейный поиск
3. Двоичный/бинарный поиск
4. Троичный/тернарный поиск
5. Экспоненциальный поиск
6. Интерполяционный поиск
7. Jump поиск

Троичный/тернарный поиск (ternary search) — алгоритм поиска минимума или максимума функции на отрезке, которая либо сначала строго возрастает, затем строго убывает, либо наоборот.



(6) Троичный поиск - ИТМО

Троичный/тернарный поиск (ternary search)

Итеративный вариант

```
double ternarySearchMin(double f(), double left, double right, double eps):
    while right - left > eps
        a = (left * 2 + right) / 3
        b = (left + right * 2) / 3
        if f(a) < f(b)
            right = b
        else
            left = a
    return (left + right) / 2
```

(7) Троичный поиск - ИТМО

Троичный/тернарный поиск (ternary search)

Итеративный вариант

```
double ternarySearchMin(double f(), double left, double right, double eps):
    if right - left < eps
        return (left + right) / 2
    a = (left * 2 + right) / 3
    b = (left + right * 2) / 3
    if f(a) < f(b)
        return ternarySearchMin(f, left, b, eps)
    else
        return ternarySearchMin(f, a, right, eps)
```

(8) Троичный поиск - ИТМО

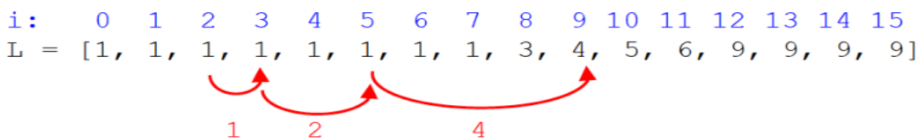
Алгоритмы поиска

План лекции

0. План лекции
1. Задача поиска
2. Линейный поиск
3. Двоичный/бинарный поиск
4. Троичный/тернарный поиск
5. Экспоненциальный поиск
6. Интерполяционный поиск
7. Jump поиск

Экспоненциальный поиск (exponential/doubling/galloping search)

Алгоритм поиска, используемый для поиска в отсортированных массивах.
Идея состоит в том, чтобы определить диапазон, в котором находится целевое значение, и выполнить бинарный поиск в пределах этого диапазона.



(9) Экспоненциальный поиск - Источник

Экспоненциальный поиск (exponential/doubling/galloping search)

```cpp
// Returns the position of key in the array arr of length size.
template <typename T>
int exponential_search(T arr[], int size, T key)
{
    if (size == 0) {
        return NOT_FOUND;
    }

    int bound = 1;
    while (bound < size && arr[bound] < key) {
        bound *= 2;
    }

    return binary_search(arr, key, bound/2, min(bound, size));
}
```

(10) Экспоненциальный поиск - Wiki

Exponential search

Class	Search algorithm
Data structure	Array
Worst-case performance	$O(\log i)$
Best-case performance	$O(1)$
Average performance	$O(\log i)$
Worst-case space complexity	$O(1)$
Optimal	Yes

(11) Сложность - Wiki

Алгоритмы поиска

План лекции

0. План лекции
1. Задача поиска
2. Линейный поиск
3. Двоичный/бинарный поиск
4. Троичный/тернарный поиск
5. Экспоненциальный поиск
6. Интерполяционный поиск
7. Jump поиск

Интерполяционный поиск (interpolation search) — алгоритм поиска, который используется для нахождения элемента в отсортированном массиве данных.

Основан на предположении, что элементы массива распределены равномерно и что искомый элемент можно найти, используя линейную интерполяцию между двумя известными элементами массива.

Interpolation Search

$$pos = lo + \left[\frac{(target - arr[lo]) * (hi - lo)}{(arr[hi] - arr[lo])} \right]$$

(12) Индекс искомого элемента - Geeks4Geeks

Интерполяционный поиск (interpolation search)

```
def interpolationSearch(arr, lo, hi, x):

    # Since array is sorted, an element present
    # in array must be in range defined by corner
    if (lo <= hi and x >= arr[lo] and x <= arr[hi]):

        # Probing the position with keeping
        # uniform distribution in mind.
        pos = lo + (((hi - lo) * (x - arr[lo])) // (arr[hi] - arr[lo]))

        # Condition of target found
        if arr[pos] == x:
            return pos

        # If x is larger, x is in right subarray
        if arr[pos] < x:
            return interpolationSearch(arr, pos + 1,
                                       hi, x)

        # If x is smaller, x is in left subarray
        if arr[pos] > x:
            return interpolationSearch(arr, lo,
                                       pos - 1, x)
    return -1
```

(13) Интерполяционный поиск - Geeks4Geeks

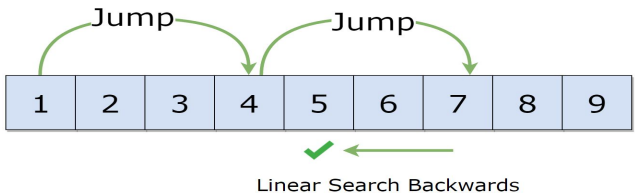
Алгоритмы поиска

План лекции

0. План лекции
1. Задача поиска
2. Линейный поиск
3. Двоичный/бинарный поиск
4. Троичный/тернарный поиск
5. Экспоненциальный поиск
6. Интерполяционный поиск
7. Jump поиск

Jump поиск (jump search) — алгоритм поиска, который используется для нахождения элемента в отсортированном массиве данных.

Основывается на более быстром (по отношению к линейному поиску) переборе элементов и поиске интервала, на котором расположен искомый элемент.



(14) Jump search - Источник