

Занятие 11

Оценка сложности алгоритмов. Поиск и сортировка данных

Поиск и сортировка данных

Линейный поиск

Пусть имеется целочисленный массив размера N . Необходимо найти все индексы, по которым в массиве расположено число 4.

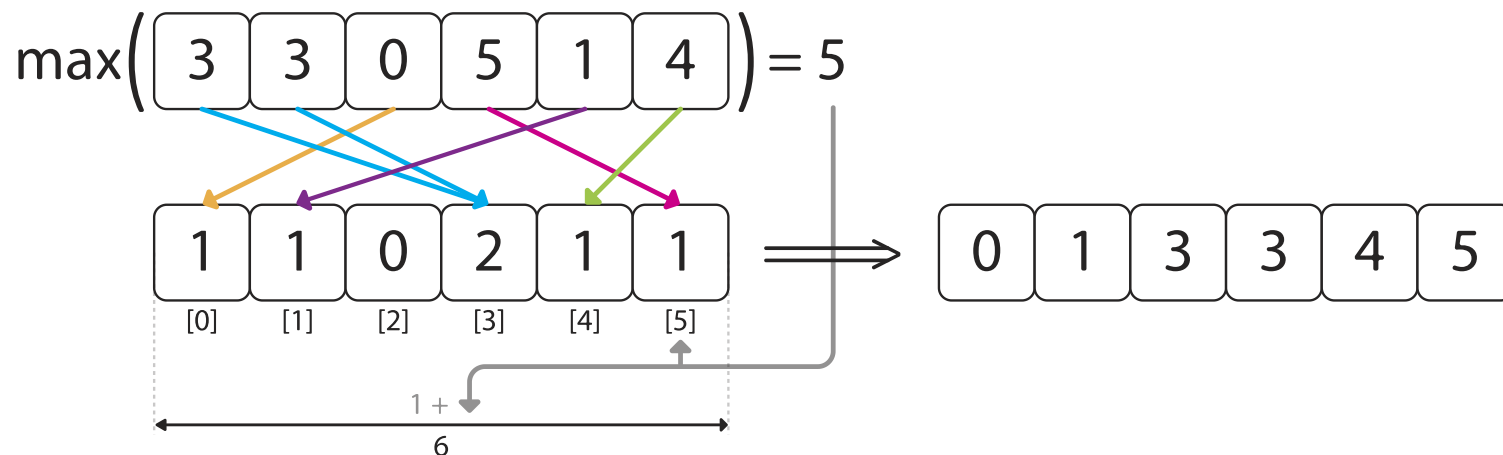
Используем линейный поиск, сравнивая каждый элемент массива с числом 4 и выводя все индексы, в которых было встречено совпадение:

```
for (int i = 0; i < N; ++i) {  
    if (arr[i] == 4) { cout << i << '\n'; }  
}
```

Время выполнения линейного поиска оценивается, как $O(N)$, потому что каждый элемент массива проверяется единожды.

Поиск и сортировка данных

Сортировка подсчетом



Сложность алгоритма

- Время выполнения — $O(N)$
- Используемая память — $O(N)$

Поиск и сортировка данных

Сортировка подсчетом

Шаги сортировки подсчетом

1. Найдем K — максимальное число в коллекции C
2. Создадим дополнительную коллекцию T размера $K + 1$
3. Пройдемся по коллекции C . Будем прибавлять единицу к элементам коллекции T с индексами, равными значениям элементов коллекции C .

```
for(size_t i = 0; i < N; ++i) { ++T[C[i]]; }
```

4. Пройдемся по коллекции T . Теперь вернем в коллекцию C каждый элемент из T столько раз, сколько он встречался в C .

```
for(size_t i = 0; i <= K; ++i) {  
    int currentIndex = 0;  
    for(size_t j = 0; j < T[i]; ++j) { C[currentIndex++] = i; }  
}
```

Случайные числа

Функция `rand()`

В C++ имеется встроенная функция `rand()`, возвращающая целое число из отрезка `[0, INT_MAX]`.

Для того, чтобы получить "случайное" число из определенного отрезка, используется следующая конструкция:

```
a + rand() % (b - a + 1);
```

Где `a` — левая граница отрезка, а `b` — правая граница отрезка.

Случайные числа

Функция `srand()`

В C++ имеется встроенная функция `srand()`, задающая зерно ГПСЧ.

Для того, чтобы получать различные последовательности случайных чисел при независимых запусках программы, в качестве зерна используется текущее время, которое легко получить, используя функцию `time()` из библиотеки `<time.h>` с аргументом `NULL`: `time(NULL)`.

```
#include <time.h>

...

srand(time(NULL));
int r = a + rand() % (b - a + 1);
```

Практические задания

1. Лине́йный поиск

Массив заполняется N случайными числами из отрезка $[a, b]$. Необходимо найти все индексы массива, в которых находится значение k .

Пользователю предлагается ввести N , k , a и b .

Вывести на экран сгенерированный массив и все индексы элементов массива, в которых находится число k . Вывести -1 , если число не было найдено в массиве.

Практические задания

1. Лине́йный поиск

Пример 1

```
Enter N, k, a, b:  
5 8 -10 10  
  
-1 -8 -3 8 -5  
Indices of 8: 3
```

Пример 2

```
Enter N, k, a, b:  
5 8 -10 10  
  
5 -7 -6 -3 9  
Indices of 8: -1
```


Практические задания

2. Сортировка подсчетом

Массив заполняется N случайными числами из отрезка $[a, b]$. Необходимо отсортировать массив по возрастанию и по убыванию значений.

Пользователю предлагается ввести N , a и b .

Вывести на экран сгенерированный массив, отсортированный по возрастанию массив и отсортированный по убыванию массив.

Практические задания

2. Сортировка подсчетом

Пример 1

```
Enter N, a, b:  
5 50 100
```

```
59 98 93 68 74  
59 68 74 93 98  
98 93 74 68 59
```

Пример 2

```
Enter N, a, b:  
5 -50 50  
'a' must be bigger than 0!
```