

Разные алгоритмы сортировок. O-символика. Шаблон `STD qsort(...)`

Носов Артём
Направление: программная инженерия
Группа: Б20-514
VK: [@evocrabforever](https://vk.com/@evocrabforever)



O-символика - что это такое и с чем едят

Определения

Пусть $f(x)$ и $g(x)$ – две функции, определенные в некоторой проколотой окрестности точки x_0 , причем в этой окрестности g не обращается в ноль. Говорят, что:

- f является «O» большим от g при $x \rightarrow x_0$, если существует такая положительная константа C , что для всех x из некоторой окрестности точки x_0 имеет место неравенство

$$|f(x)| < C |g(x)|;$$

- f является «o» малым от g при $x \rightarrow x_0$, если для любой положительной константы c найдется такая окрестность точки x_0 , что для всех x из этой окрестности имеет место неравенство

$$|f(x)| < c |g(x)|.$$

Иначе говоря, в первом случае отношение $|f|/|g|$ в окрестности точки x_0 ограничено сверху, а во втором оно стремится к нулю.

[Источник](#)

O-символика - что это такое и с чем едят

Если сложность нашего алгоритма записанного в функции f это $O(n)$ то это значит то же самое что и $|f(x)| < C|g(x)|$ при $g(x) = n$ и вектор строка x аргументов функций f и g лежат в некоторой окрестности x_0

Упражнение на вычисление сложности алгоритма

Для оценки сложности алгоритмов:

1. Находим объем входных данных (N)
2. Оцениваем скорость с точностью до коэффициента
3. Всегда берем наихудший случай

Сложностью алгоритма называется величина $T(n)$, где n – размер задачи.

Упражнение на вычисление сложности алгоритма

Предположим, что у нас есть задача – отсортировать неупорядоченный массив чисел по возрастанию. Для этого можно применить простейший алгоритм сортировки – «пузырьковую» сортировку.

Массив можно упорядочить, если последовательно менять местами элементы стоящие «неправильно», то есть не в той последовательности. Сравнить числа можно как с левого конца массива, так и с правого. Предположим, что мы сравниваем элементы с правого конца массива, тогда можно предложить следующий алгоритм:

1. Сравним выбранный i -й элемент массива с $i-1$ элементом. Если $i-1$ элемент больше, то меняем их местами, иначе оставляем нетронутыми.
2. Продолжаем процесс для i от N до 2.
3. Повторяем пункты 1 и 2 N раз.
4. После того, как мы выполним пункты 1 и 2 в первый раз, в начале массива окажется наименьший его элемент. Повторяя процесс, мы получим полностью отсортированный массив.

Упражнение на вычисление сложности алгоритма



Bubble Sort **in 2**



Упражнение на вычисление сложности алгоритма

Можно заметить, что нет смысла каждый раз сравнивать все элементы, так как после первого прохода массива в начале окажется наименьший элемент, после второго – 2 наименьших, и так далее. В модифицированном виде алгоритм можно записать так:

```
for(int i = 0; i < n; i++)  
    for(int j = 1; j < n - i; j++)  
        if (arr[j-1] > arr[j]) {  
            tmp = arr[j];  
            arr[j] = arr[j-1];  
            arr[j-1] = tmp;  
        }
```

Упражнение на вычисление сложности алгоритма

Такой алгоритм будет делать $(N-1)+(N-2)+\dots+2+1$ шагов, что можно представить как $(N-1)N/2$. Таким образом $T(n) \approx n^2$ для данного алгоритма.

По сложности алгоритмы делятся на

1. Полиномиальные ($T(n)=n^k$)
2. Экспоненциальные ($T(n)=a^n$)
3. Логарифмическая ($T(n)=\log_a n$)

Быстрая сортировка. Компараторы и указатели на функции

Ссылка на стрим:

[Источник информации про qsort](#)

Функции:

ТипВозвращаемогоЗначения ИмяФункции(СписокФормальныхАргументов)

{

ТелоФункции;

return(ВозвращаемоеЗначение);

}

Указатели на функции:

ТипВозвращаемогоЗначения (*ИмяФункции)(СписокФормальныхАргументов);

Сортировки

Алгоритмы сортировки: сортировка пузырьком

[Ссылка](#)

Алгоритмы сортировки: сортировка выбором

[Википедия](#)

Алгоритмы сортировки: сортировка перемешиванием

[Википедия](#)

Алгоритмы сортировки: сортировка Шелла

[Википедия](#)

Алгоритмы сортировки: гномья сортировка

[Википедия](#)

Алгоритмы сортировки: сортировка вставками

[Википедия](#) и [реализация](#)

Алгоритмы сортировки: сортировка слиянием

[Википедия](#)

Практика: таймирование сортировки

Практика: быстро сортируем массивы под разные задачи)

Есть массив содержащий в себе количества посещений спортзала разными людьми, отсортировать число посещений по убыванию.

$N = 2000$ число посещений в диапазоне от 10 до 200.

Практика: быстро сортируем массивы под разные задачи)

Есть массив содержащий в себе количества посещений спортзала разными людьми, отсортировать число посещений по убыванию.

$N = 2000$ число посещений в диапазоне от 10 до 200.

1. Сгенерировать массив случайных чисел для сортировки;
2. Выбрать и применить одну из сортировок (сортируем по убыванию);
3. Распечатать результат в удобном виде в консоль.

ДЗ по сортировкам

Разработать функцию осуществляющую адаптивную сортировку целочисленных массивов, которая на основании данных о массиве будет с помощью условных конструкций выбирать оптимальный алгоритм сортировки. Пример, bubble sort подходит для очень коротких массивов от 10 до 100 элементов, а quicksort для огромных массивов. Чем глубже проработаете тему и больше сортировок используете тем лучше :з