ЛАБОРАТОРНЫЕ РАБОТЫ №4-5

По дисциплине: Алгоритмы и структуры данных
Тема работы: **Алгоритмы методов сортировки: быстрая сортировка и рандомизированная быстрая сортировка**
Цель работы: выполнить программные реализации и визуализации алгоритмов быстрой и рандомизированной сортировок
Количество часов: 4

Содержание работы:

1. Описание главной функции
2. Добавление функций алгоритма сортировки
3. Анализ времени работы алгоритма для разных наборов чисел
4. Выводы

Методические указания по выполнению

Алгоритм быстрой сортировки (Quicksort)

Описание сортировки. Быстрая сортировка применяет метод «разделяй и властвуй». И реализуется функцией *QuickSort()*.

Массив *A[p..r]* разбивается на два подмассива *A[p..q-1]* и *A[q+1..r]*, таких, что каждый элемент первого подмассива меньше или равен элементу *A[q]*, а он в свою очередь не превышает любой элемент второго подмассива.

```
QuickSort(A,p,r)
1. if p < r
2.     q=Partition(A,p,r)
3.     QuickSort(A,p,q-1)
4.     QuickSort(A,q+1,r)
```

Подмассивы *A[p..q-1]* и *A[q+1..r]* сортируются с помощью рекурсивного вызова функции быстрой сортировки *QuickSort()*.

Весь массив можно отсортировать, вызвав функцию *QuickSort(A,1,n)*.

Ключевой функцией алгоритма является *Partition()*, которая изменяет порядок следования элементов *A[p..r]*.

```
Partition(A,p,r)
1. x=A[r]
2. i=p-1
3. for j=p to r-1
4.     if A[j] ≤ x
5.         i=i+1
6.         обменять A[i] и A[j]
7. обменять A[i+1] и A[r]
8. return i+1
```

Функция выбирает элемент *x=A[r]* в качестве опорного.

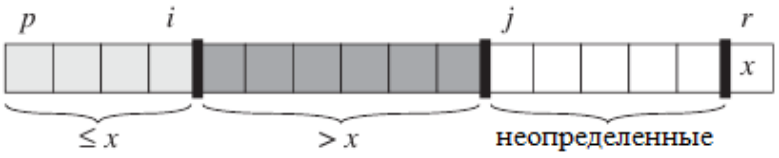

Рис. 4. Области, поддерживаемые функцией *Partition()*.

Затем подмассив *A[p..r]* разбивается относительно него на 4 области, как показано на рис. 4. Сначала, все элементы относятся к 3й и 4й областям (строки 1-2).

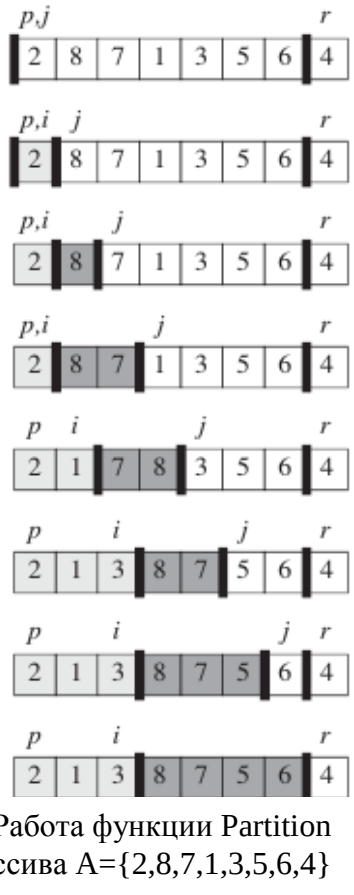

Рис. 5. Работа функции Partition на примере массива A={2,8,7,1,3,5,6,4}

Затем функция осуществляет разделение массива, которое перемещает все ключи, меньшие, либо равные A[j], в первую область, а все ключи, большие, либо равные A[j] – во вторую область (стр. 3-6). В строке 7 последний, опорный, элемент меняется местами с тем, который находится между двумя областями. Таким образом, функция возвращает позицию опорного элемента в переменную *q*, относительно которого все элементы, меньшие его, находятся слева от него, а большие – справа.

Особенности алгоритма. Последовательность сортируется на месте (in place), без использования дополнительной памяти. Алгоритм работает на основе метода «разделяй и властвуй».

Время работы. Время работы алгоритма зависит от того, на какие части разбивается массив. Если массив разбивается на примерно одинаковые части, то время работы алгоритма является *логарифмической функцией Θ(n lg n)*. Наихудший случай получается когда одна часть содержит *n-1* элемент, а вторая – только 1. Это происходит, когда массив на входе уже отсортирован. Программа просто вызовет сама себя *n* раз, каждый раз с меньшим на один элемент массивом. Время сортировки в худшем случае является *квадратичной функцией* $\Theta(n^2)$.

Рандомизированная быстрая сортировка (Randomized Quicksort)

Описание сортировки

Применяется метод случайной выборки: вместо опорного элемента *A[r]*, такой элемент выбирается в массиве *A[p..r]* случайным образом. Это обеспечит равную вероятность оказаться опорным любому из элементов подмассива. Благодаря случайному выбору опорного элемента можно ожидать, что разбиение входного массива окажется хорошо сбалансированным.

```
Randomized_Partition(A,p,r)
1. x = Random(p,q)
2. обменять A[r] и A[i]
3. return Partition(A,p,r)
```

```
Randomized_Quicksort(A,p,r)
1. if p < r
2.     q = Randomized_Partition(A,p,r)
3.     Randomized_Quicksort(A,p,q-1)
4.     Randomized_Quicksort(A,q+1,r)
```

Особенности алгоритма. Последовательность сортируется на месте (in place), без использования дополнительной памяти. Алгоритм работает на основе метода «разделяй и властвуй».

Время работы. Время работы алгоритма не зависит от порядка элементов на входе. Ожидаемое время функции *Randomized_Quicksort(A,p,r)* равно *O(n lg n)* и практически никакие входные данные не могут вызвать наихудшее поведение алгоритма.

Задачи

1. Покажите работу функции *Partition()* для массива *A={25,17,12,21,8,14,9,27,10,4}* при граничном элементе *x=12*.
2. Покажите работу функции *Quicksort(A)* по сортировке массива *A={4,7,9,2,6,1,3,5}* в неубывающем порядке, выбирая в качестве опорного первый элемент подмассива.
3. Укажите, что необходимо изменить в алгоритме быстрой сортировки, чтобы она выполнялась в невозрастающем порядке.
4. Какое значение q возвращает функция *Partition()*, если все элементы массива одинаковы.
5. Покажите работу функции *Randomized_Quicksort(A)* по сортировке массива *A={4,7,9,2,6,1,3,5}* в невозрастающем порядке.

Пособия и инструменты

1. MS Visual Studio 2008 / 2010 / 2012 / 2013
2. Data Structure Visualizations. [Электронный ресурс] – Режим доступа: http://www.cs.usfca.edu/~galles/visualization/java/download.html.

Вопросы для защиты лабораторной работы

1. Дайте асимптотическую оценку работы алгоритма быстрой сортировки.
2. Дайте асимптотическую оценку работы алгоритма быстрой рандомизированной сортировки.
3. Опишите особенности обоих алгоритмов.
4. Как поведет себя алгоритм быстрой сортировки, если на вход подать отсортированную в обратном порядке последовательность.
5. Обдумайте, какие модификации вы бы внесли для исключения переполнения стека.
6. Как изменится количество рекурсий, если разбивать массив на три части?
7. Опишите оптимизацию Боба Седжвика для алгоритма данной сортировки.

Литература

1. Кормен Т.Х. Алгоритмы: построение и анализ, 3-е издание. : Пер. с англ. / Т.Х. Кормен, Ч.И. Лейзерсон, Р.Л. Ривест, К. Штайн. – М.: Издательский дом «Вильямс», 2013. – 1328 с.
2. Algorithms and Data Structures with implementations in Java and C++ [Electronic Resource]. – URL: http://www.algolist.net/.
3. Data Structure Visualizations / David Galles, Department of Computer Science // University of San Francisco [Electronic Resource]. – URL: http://www.cs.usfca.edu/~galles/visualization/Algorithms.html.
4. Data Structures and Algorithms / Java Applets Centre [Electronic Resource]. – URL: http://www.cosc.canterbury.ac.nz/mukundan/dsal/appldsal.html.
5. Анимированные визуализации структур данных / VISUALGO [Electronic Resource]. – URL: http://ru.visualgo.net/.