ЛАБОРАТОРНАЯ РАБОТА №2

По дисциплине: Алгоритмы и структуры данных
Тема работы: **Алгоритмы методов сортировки: сортировка слиянием (Merge sort)**
Цель работы: выполнить программную реализацию и визуализацию алгоритма сортировки слиянием
Количество часов: 2

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

Описание сортировки

Эта сортировка – хороший пример использования метода «разделяй и властвуй». Сначала сложная задача разбивается на несколько простых, которые подобны исходной задаче, но имеют меньший объем. Далее эти задачи решаются рекурсивным методом, после чего полученные решения комбинируются для решения исходной задачи.

Для решения задачи сортировки эти три этапа выглядят так:

- сортируемая последовательность разбивается на две части размером $n/2$;
- каждая из получившихся частей сортируется отдельно, например – тем же самым алгоритмом;
- две упорядоченные подпоследовательности соединяются в одну.

Рекурсия достигает своего нижнего предела, когда длина сортируемой последовательности становится равной 1 (любая последовательность длины 1 считается упорядоченной), как показано на рис. 2.

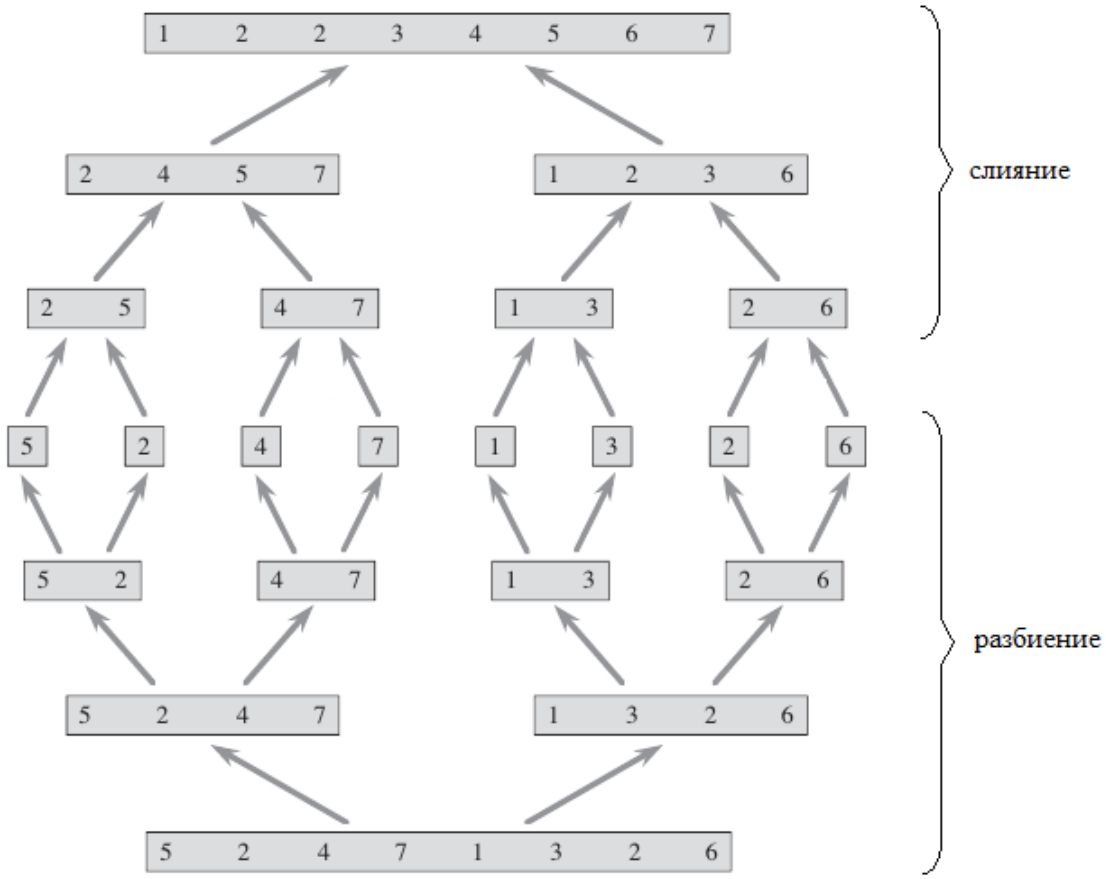

Рис. 2. Выполнение алгоритма Merge_Sort над массивом A={54,2,74,75,4}

Ключевая операция – объединение двух отсортированных последовательностей. Это делается с помощью функции *Merge (A, p, q, r)*, где *A* – массив, *p, q, r* – индексы массива (*p<=q<r*). Здесь предполагается, что элементы подмассивов *A[p..q]* и *A[q+1..r]* упорядочены. Она сливает эти два подмассива в один отсортированный (временный массив), элементы которого заменяют текущие элементы подмассива *A[p..r]*. Для ее выполнения требуется время *Θ(n).*

```
Merge (A,p,q,r)
1.  i=p
2.  j=q+1
3.  k=1
4.  // создать массив C длинной r-p+1
5.  while (i ≤ q and j ≤ r)
6.     if A[i] < A[j]
7.          C[k] = A[i]
8.          i= i+1
9.     else C[k] = A[j]
10.         j=j+1
11.    k=k+1
12. if i ≤ q
13.    l=i
14. else l=j
15. while k ≤ r-p
16.    C[k] = A[l]
17.    k=k+1
18.    l=l+1
19. for l=1 to k
20.    A[l+p-1] = C[l]
```

Теперь функцию *Merge()* можно использовать в качестве подпрограммы в алгоритме сортировки слиянием. Функция *Merge_sort (A, p, r)* выполняет сортировку элементов в подмассиве *A[p..r]*. При этом вычисляется число *q*, которое делит массив на две примерно равные части *A[p..q]* и *A[q+1..r]*. Рекурсивный вызов осуществляется пока p<r. Данное условие нарушается и разбиение останавливается, когда правя и левая границы подмассива начнут указывать на один и тот же элемент (p=r), то есть когда в подмассивах останутся единичные элементы. После этого начинается слияние. Функция *Merge()* соединяет два отсортированных фрагмента массива в один. Весь массив можно отсортировать, вызвав функцию *Merge_Sort(A, 1, n).*

```
Merge_Sort (A,p,r)
1.  if p < r
2.     q= (p+r)/2
3.     Merge-Sort(A,p,q)
4.     Merge-Sort(A,q+1,r)
5.     Merge(A,p,q,r)
```

Особенности алгоритма. Алгоритм работает с применением дополнительной памяти размером *n* для хранения вспомогательного массива – слияние требует дополнительного массива для записи результатов слияния. Алгоритм работает на основе метода «разделяй и властвуй».

Время работы. Для больших *n* сортировка слиянием оказывается эффективнее сортировки вставками. Время, затраченное на разбиение, составляет *Θ(1)*. Рекурсивный спуск до единичных подмассивов включает *lg n +1* уровней. Слияние подмассивов на каждом уровне включает *n* операций. Таким образом, пренебрегая незначительными и не влияющими на рост элементами, время работы алгоритма является логарифмической функцией *Θ(n lg n).*

Задачи

1. Покажите работу функции *Merge_Sort (A)* по сортировке массива *A = {7, 3, 4, 9, 5, 1, 2, 6}* в неубывающем порядке.

2. Покажите работу функции *Merge_Sort (A)* по сортировке массива *A = {5, 2, 8, 4, 7, 3, 1, 9}* в неубывающем порядке.

3. Дана функция $f(n) = 120n^2 + 7$. Показать, что $f(n) = O(n^3)$.

4. Покажите последовательность слияний, выполняемых функцией *Merge_Sort (A)* при сортировке ключей *E, A, S, Y, S, O, L, U, T, I, O, N*.
5. Выделяйте память массиву C[] динамически и размером, достаточным только для обработки его в цикле while в 5-11 строках. Затем добавьте обработку оставшихся элементов в исходном массиве.

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 сортировки слиянием.
2. Какие особенности у данного алгоритма.
3. Для чего используется дополнительная память при работе данного алгоритма.
4. Обязательно ли оба входных подмассива в функции Merge() должны быть отсортированы? Обоснуйте ответ и приведите свой контрпример.
5. Какие способы экономии памяти в работе данного алгоритма вы можете предложить?

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/.