Алгоритмы и структуры данных

Задание к лабораторной работе №2. Сортировка слиянием. Метод декомпозиции

Лабораторная работа посвящена сортировке слиянием и в целом методу декомпозиции (Разделяй и властвуй). Аналогично второй лабораторной работе, есть два способа ее выполнения и защиты:

1 **Базовый уровень.** Решается 3 задачи по вариантам. Варианты в табличке внизу, номер вашего варианта соответствует вашему номеру в списке группы. Посмотреть свой номер можно, например, в журнале успеваемости по дисциплине. В этом случае максимум за защиту можно получить 4 балла. В сумме с самой работой (0,5 балла) и отчетом (1 балл) получается 5,5 балла, что достаточно для зачета.

2 **Продвинутый уровень.** Решаются все задачи или минимум 5 задач, причем 3 из них - исходя из вашего варианта, а остальные две - по выбору. В этом случае вы сможете получить максимальные 7,5 баллов. *Задача №10 - не обязательная.*

Вариант	Номера задач	Вариант	Номера задач
1	1,2,3	16	1,4,7
2	1,2,4	17	1,4,8
3	1,2,5	18	1,4,9
4	1,2,6	19	1,5,6
5	1,2,7	20	1,5,7
6	1,2,8	21	1,5,8
7	1,2,9	22	1,5,9
8	1,3,4	23	1,6,7
9	1,3,5	24	1,6,8
10	1,3,6	25	1,6,9
11	1,3,7	26	1,7,8
12	1,3,8	27	1,7,9
13	1,3,9	28	1,8,9
14	1,4,5		
15	1,4,6		

Если какая-то из задач вашего варианта кажется вам слишком сложной, вы можете решить другую задачу, которая вам больше нравится, или посмотреть на задачу №10.

1 задача. Сортировка слиянием

1. Используя *псевдокод* процедур `Merge` и `Merge-sort` из презентации к Лекции 2 (страницы 6-7), напишите программу сортировки слиянием на Python и проверьте сортировку, создав несколько рандомных массивов, подходящих под параметры:

   - **Формат входного файла (input.txt).** В первой строке входного файла содержится число n ($1 \leq n \leq 2 \cdot 10^4$) — число элементов в массиве. Во второй строке находятся n различных целых чисел, по модулю не превосходящих 10^9.
   - **Формат выходного файла (output.txt).** Одна строка выходного файла с отсортированным массивом. Между любыми двумя числами должен стоять ровно один пробел.
   - Ограничение по времени. 2сек.
   - Ограничение по памяти. 256 мб.

2. Для проверки можно выбрать наихудший случай, когда сортируется массив размера 1000, $10^4, 10^5$ чисел порядка 10^9, отсортированных в обратном порядке; наилучший, когда массив уже отсортирован, и средний. Сравните, например, с сортировкой вставкой на этих же данных.

3. Перепишите процедуру `Merge` так, чтобы в ней не использовались сигнальные значения. Сигналом к остановке должен служить тот факт, что все элементы массива L или R скопированы обратно в массив A, после чего в этот массив копируются элементы, оставшиеся в непустом массиве.

или перепишите процедуру `Merge` (и, соответственно, `Merge-sort`) так, чтобы в ней не использовались значения границ и середины - p, r и q.

2 задача. Сортировка слиянием+

Дан массив целых чисел. Ваша задача — отсортировать его в порядке неубывания *с помощью сортировки слиянием*.

Чтобы убедиться, что Вы действительно используете сортировку слиянием, мы просим Вас, после каждого осуществленного слияния (то есть, когда соответствующий подмассив уже отсортирован!), выводить индексы граничных элементов и их значения.

- **Формат входного файла (input.txt).** В первой строке входного файла содержится число n ($1 \leq n \leq 10^5$) — число элементов в массиве. Во второй строке находятся n различных целых чисел, по модулю не превосходящих 10^9.

- **Формат выходного файла (output.txt).** Выходной файл состоит из нескольких строк.

– В **последней строке** выходного файла требуется вывести отсортированный в порядке неубывания массив, данный на входе. Между любыми двумя числами должен стоять ровно один пробел.

– Все предшествующие строки описывают осуществленные слияния, по одному на каждой строке. Каждая такая строка должна содержать по четыре числа: I_f, I_l, V_f, V_l, где I_f — индекс начала области слияния, I_l — индекс конца области слияния, V_f — значение первого элемента области слияния, V_l — значение последнего элемента области слияния.

– Все индексы начинаются с единицы (то есть, $1 \le I_f \le I_l \le n$). **Индексы области слияния должны описывать положение области слияния в исходном массиве!** Допускается не выводить информацию о слиянии для подмассива длиной 1, так как он отсортирован по определению.

- Ограничение по времени. 2сек.

- Ограничение по памяти. 256 мб.

- **Приведем небольшой пример:** отсортируем массив $[9, 7, 5, 8]$. Рекурсивная часть сортировки слиянием (процедура SORT(A, L, R), где A — сортируемый массив, L — индекс начала области слияния, R — индекс конца области слияния) будет вызвана с $A = [9, 7, 5, 8]$, $L = 1$, $R = 4$ и выполнит следующие действия:

  – разделит область слияния $[1, 4]$ на две части, $[1, 2]$ и $[3, 4]$;

  – выполнит вызов SORT($A, L = 1, R = 2$):

    * разделит область слияния $[1, 2]$ на две части, $[1, 1]$ и $[2, 2]$;
    * получившиеся части имеют единичный размер, рекурсивные вызовы можно не делать;
    * осуществит слияние, после чего A станет равным $[7, 9, 5, 8]$;
    * выведет описание слияния: $I_f = L = 1,\ I_l = R = 2,\ V_f = A_L = 7,\ V_l = A_R = 9$.

  – выполнит вызов SORT($A, L = 3, R = 4$):

    * разделит область слияния $[3, 4]$ на две части, $[3, 3]$ и $[4, 4]$;
    * получившиеся части имеют единичный размер, рекурсивные вызовы можно не делать;
    * осуществит слияние, после чего A станет равным $[7, 9, 5, 8]$;
    * выведет описание слияния: $I_f = L = 3,\ I_l = R = 4,\ V_f = A_L = 5,\ V_l = A_R = 8$.

  – осуществит слияние, после чего A станет равным $[5, 7, 8, 9]$;

  – выведет описание слияния: $I_f = L = 1,\ I_l = R = 4,\ V_f = A_L = 5,\ V_l = A_R = 9$.

- Описания слияний могут идти в произвольном порядке, необязательно совпадающем с порядком их выполнения. Однако, с целью повышения производительности, рекомендуем выводить эти описания сразу, не храня их в памяти. Именно по этой причине отсортированный массив выводится в самом конце.

- Пример:

input.txt	output.txt
10 1 8 2 1 4 7 3 2 3 6	1 2 1 8 3 4 1 2 1 4 1 8 5 6 4 7 1 6 1 8 7 8 2 3 9 10 3 6 7 10 2 6 1 10 1 8 1 1 2 2 3 3 4 6 7 8

Любая корректная сортировка слиянием, делящая подмассивы на две части (необязательно равных!), будет зачтена, если успеет завершиться, уложившись в ограничения.

3 задача. Число инверсий

Инверсией в последовательности чисел A называется такая ситуация, когда $i < j$, а $A_i > A_j$. Количество инверсий в последовательности в некотором роде определяет, насколько близка данная последовательность к отсортированной. Например, в сортированном массиве число инверсий равно 0, а в массиве, сортированном наоборот - каждые два элемента будут составлять инверсию (всего $n(n-1)/2$).

Дан массив целых чисел. Ваша задача — подсчитать число инверсий в нем.

Подсказка: чтобы сделать это быстрее, можно воспользоваться модификацией сортировки слиянием.

- **Формат входного файла (input.txt).** В первой строке входного файла содержится число n ($1 \leq n \leq 10^5$) — число элементов в массиве. Во второй строке находятся n различных целых чисел, по модулю не превосходящих 10^9.

- **Формат выходного файла (output.txt).** В выходной файл надо вывести число инверсий в массиве.

- Ограничение по времени. 2сек.

- Ограничение по памяти. 256 мб.

- Пример:

input.txt	output.txt
10 1 8 2 1 4 7 3 2 3 6	17

4 задача. Бинарный поиск

В этой задаче вы реализуете алгоритм бинарного поиска, который позволяет очень эффективно искать (даже в огромных) списках при условии, что список отсортирован. Цель - реализация алгоритма двоичного (бинарного) поиска.

- **Формат входного файла (input.txt).** В первой строке входного файла содержится число n ($1 \leq n \leq 10^5$) — число элементов в массиве, и последовательность $a_0 < a_1 < ... < a_{n-1}$ из n **различных** положительных целых чисел в порядке возрастания, $1 \leq a_i \leq 10^9$ для всех $0 \leq i < n$. Следующая строка содержит число k, $1 \leq k \leq 10^5$ и k положительных целых чисел $b_0, ...b_{k-1}$, $1 \leq b_j \leq 10^9$ для всех $0 \leq j < k$.
- **Формат выходного файла (output.txt).** Для всех i от 0 до $k-1$ вывести индекс $0 \leq j \leq n-1$, такой что $a_i = b_j$ или -1, если такого числа в массиве нет.
- Ограничение по времени. 2сек.
- Ограничение по памяти. 256 мб.
- Пример:

input.txt	output.txt
5 1 5 8 12 13 5 8 1 23 1 11	2 0 -1 0 -1

  В этом примере есть возрастающая последовательность из $a_0 = 1, a_1 = 5, a_2 = 8, a_3 = 12$ и $a_4 = 13$ длиной в $n = 5$ и пять чисел для поиска: 8 1 23 1 11. Видно, что $a_2 = 8$ и $a_0 = 1$, но чисел 23 и 11 нет в последовательности a, поэтому они имеют индекс -1. В итоге ответ: 2 0 -1 0 -1.

5 задача. Представитель большинства

Правило большинства - это когда выбирается элемент, имеющий больше половины голосов. Допустим, есть последовательность A элементов $a_1, a_2, ...a_n$, и нужно проверить, содержит ли она элемент, который появляется больше, чем $n/2$ раз. Наивный метод это сделать:

```
Majority(A):
for i from 1 to n:
        current_element = a[i]
        count = 0
        for j from 1 to n:
                 if a[j] = current_element:
                        count = count+1
        if count > n/2:
                return a[i]
return "нет элемента большинства"
```

Очевидно, время выполнения этого алгоритма квадратично. Ваша цель - использовать метод "Разделяй и властвуй"для разработки алгоритма проверки, содержится ли во входной последовательности элемент, который встречается больше половины раз, за время $O(n \log n)$.

- **Формат входного файла (input.txt).** В первой строке входного файла содержится число n ($1 \leq n \leq 10^5$) — число элементов в массиве. Во второй строке находятся n положительных целых чисел, по модулю не превосходящих 10^9, $0 \leq a_i \leq 10^9$.
- **Формат выходного файла (output.txt).** Выведите 1, если во входной последовательности есть элемент, который встречается строго больше половины раз; в противном случае - 0.
- Ограничение по времени. 2сек.
- Ограничение по памяти. 256 мб.
- Пример 1:

input.txt	output.txt
5 2 3 9 2 2	1

Число "2"встречается больше 5/2 раз.

- Пример 2:

input.txt	output.txt
4 1 2 3 4	0

Нет элемента, встречающегося больше $n/2$ раз.

6 задача. Поиск максимальной прибыли

Используя *псевдокод* процедур `Find Maximum Subarray` и `Find Max Crossing Subarray` из презентации к Лекции 2 (страницы 25-26), напишите программу поиска максимального подмассива.

Примените ваш алгоритм для ответа на следующий вопрос. Допустим, у нас есть данные по акциям какой-либо фирмы за последний месяц (год, или иной срок).

Проанализируйте этот срок и выдайте ответ, в какой из дней при покупке единицы акции данной фирмы, и в какой из дней продажи, вы бы получили максимальную прибыль? Выдайте дату покупки, дату продажи и максимальную прибыль.

Вы можете использовать любые данные для своего анализа. Например, я набрала в Google *"акции"* и мне поиск выдал акции Газпрома, тут - можно скачать информацию по стоимости акций за любой период. (Перейдя по ссылке, нажмите на вкладку "Настройки"→ "Скачать")

Соответственно, вам нужно только выбрать данные, посчитать ***изменение цены*** и применить алгоритм поиска максимального подмассива.

- **Формат входного файла** в данном случае на ваше усмотрение.
- **Формат выходного файла (output.txt).** Выведите название фирмы, рассматриваемый вами срок изменения акций, дату покупки и дату продажи единицы акции, чтобы получилась максимальная выгода; и сумма этой прибыли.

7 задача. Поиск максимального подмассива за линейное время

Можно найти максимальный подмассив за линейное время, воспользовавшись следующими идеями. Начните с левого конца массива и двигайтесь вправо, отслеживая найденный к данному моменту максимальный подмассив. Зная максимальный подмассив массива $A[1..j]$, распространите ответ на поиск максимального подмассива, заканчивающегося индексом $j + 1$, воспользовавшись следующим наблюдением: максимальный подмассив массива $A[1..j + 1]$ представляет собой либо максимальный подмассив массива $A[1..j]$, либо подмассив $A[i..j + 1]$ для некоторого $1 \leq i \leq j + 1$. Определите максимальный подмассив вида $A[i..j + 1]$ за константное время, зная максимальный подмассив, заканчивающийся индексом j.

В этом случае у вас возможны 2 варианта тестирования: первый предполагает создание рандомного массива чисел, аналогично **задаче №1** (в этом случае формат входного и выходного файла смотрите там). Второй вариант - взять любые данные по акциям какой-либо компании, аналогично **задаче №6**.

8 задача. Умножение многочленов

Выдающийся немецкий математик Карл Фридрих Гаусс (1777—1855) заметил, что хотя формула для произведения двух комплексных чисел $(a + bi)(c + di) = ac - bd + (bc + ad)i$ содержит *четыре* умножения вещественных чисел, можно обойтись и *тремя*: вычислим ac, bd и $(a + b)(c + d)$ и воспользуемся тем, что $bc + ad = (a + b)(c + d) - ac - bd$.

Задача. Даны 2 многочлена порядка $n - 1$: $a_{n-1}x^{n-1} + a_{n-2}x^{n-1} + ... + a_1x + a_0$ и $b_{n-1}x^{n-1} + b_{n-2}x^{n-1} + ... + b_1x + b_0$. Нужно получить произведение:

$c_{2n-2}x^{2n-2} + c_{2n-3}x^{2n-3} + ... + c_1x + c_0$, где:

$$\begin{aligned} c_{2n-2} &= a_{n-1}b_{n-1} \\ c_{2n-3} &= a_{n-1}b_{n-2} + a_{n-2}b_{n-1} \\ ... & \quad ... \\ c_2 &= a_2b_0 + a_1b_1 + a_0b_2 \\ c_1 &= a_1b_0 + a_0b_1 \\ c_0 &= a_0b_0 \end{aligned}$$

Пример. Входные данные: $n = 3,\ A = (3, 2, 5),\ B = (5, 1, 2)$

$$\begin{aligned} A(x) &= 3x^2 + 2x + 5 \\ B(x) &= 5x^2 + x + 2 \\ A(x)B(x) &= 15x^4 + 13x^3 + 33x^2 + 9x + 10 \end{aligned}$$

Ответ: $C = (15, 13, 33, 9, 10)$.

- **Формат входного файла (input.txt).** В первой строке число n - порядок многочленов A и B. Во второй строке коэффициенты многочлена A через пробел. В третьей строке коэффициенты многочлена B через пробел.

- **Формат выходного файла (output.txt).** Ответ - одна строка, коэффициенты многочлена $C(x) = A(x)B(x)$ через пробел.

- Нужно использовать метод "Разделяй и властвуй". Подсказка: любой многочлен $A(x)$ можно разделить на 2 части, например, $A(x) = 4x^3 + 3x^2 + 2x + 1$ разделим на $A_1 = 4x+3$ и $A_2 = 2x+1$. И многочлен $B(x) = x^3+2x^2+3x+4$ разделим на 2 части: $B_1 = x + 2$, $B_2 = 3x + 4$. Тогда произведение $C = A(x) * B(x) = (A_1B_1)x^n + (A_1B_2 + A_2B_1)x^{n/2} + A_2B_2$ - требуется 4 произведения (проверьте правильность данной формулы). Можно использовать формулу Гаусса и обойтись всего тремя произведениями.

9 задача. Метод Штрассена для умножения матриц

Умножение матриц. Простой метод. Если есть квадратные матрицы $X = (x_{ij})$ и $Y = (y_{ij})$, то их произведение $Z = X \cdot Y \Rightarrow z_{ij} = \sum_{k=1}^{n} x_{ik} \cdot y_{kj}$. Нужно вычислить n^2 элементов матрицы, каждый из которых представляет собой сумму n значений.

```
Matrix_Multiply(X, Y)::
  n = X.rows
  Z - квадратная матрица размера n
  for i = 1 to n:
    for j = 1 to n:
      z[i,j] = 0
      for k = 1 to n:
        z[i,j] = z[i,j] + x[i,k]*y[k,j]
  return Z
```

Задачу умножения матриц достаточно легко разбить на подзадачи, поскольку произведение можно составлять из *блоков*. Разобьём каждую из матриц X и Y на четыре блока размера $\frac{n}{2} \times \frac{n}{2}$:

$$X = \begin{bmatrix} A & B \\ C & D \end{bmatrix}, \; Y = \begin{bmatrix} E & F \\ G & H \end{bmatrix},$$

Тогда их произведение выражается в терминах этих блоков по обычной формуле умножения матриц :

$$XY = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \cdot \begin{bmatrix} E & F \\ G & H \end{bmatrix} = \begin{bmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{bmatrix}$$

Вычислив рекурсивно восемь произведений $AE, BG, AF, BH, CE, DG, CF, DH$ и просуммировав их за время $O(n^2)$, мы вычислим необходимое нам произведение матриц. Соответствующее рекуррентное соотношение на время работы алгоритма

$$T(n) = 8T(\frac{n}{2}) + O(n^2).$$

Какое получилось время у предыдущего рекурсивного алгоритма? Да, ничуть не лучше наивного. Однако его можно ускорить с помощью алгебраического трюка: для вычисления произведения XY достаточно перемножить ***семь*** пар матриц размера $\frac{n}{2} \times \frac{n}{2}$, после чего хитрым образом (*и как только Штрассен догадался?*) получить ответ:

$$XY = \begin{bmatrix} P_5 + P_4 - P_2 + P_6 & P_1 + P_2 \\ P_3 + P_4 & P_1 + P_5 - P_3 - P_7 \end{bmatrix}$$

где

$$\begin{gathered} P_1 = A(F - H), \qquad P_5 = (A + D)(E + H), \\ P_2 = (A + B)H, \qquad P_6 = (B - D)(G + H), \\ P_3 = (C + D)E, \qquad P_7 = (A - C)(E + F), \\ P_4 = D(G - E). \end{gathered}$$

- **Цель**. Применить метод Штрассена для умножения матриц и сравнить его с простым методом. *Найти размер матриц n, при котором метод Штрассена работает существенно быстрее простого метода.*
- **Формат входа**. Стандартный ввод или input.txt. Первая строка - размер **квадратных** матриц n для умножения. Следующие строки соответсвенно сами значения матриц A и B.
- **Формат выхода**. Стандартный вывод или output.txt. Матрица $C = A \cdot B$.

10 задача★.

1. Реализуйте сортировку слиянием, учитывая, что можно сэкономить на отсортированных массивах, которые не нужно объединять. Проверьте $A[q]$, меньше он или равен $A[q+1]$, и объедините их, только если $A[q] > A[q+1]$, где q - середина при делении в Merge_Sort.

2. В небольших массивах сортировки методом вставок и методом выбора могут работать быстрее сортировки слиянием. Сравните свои реализации сортировки методом вставок, методом выбора и сортировки слиянием и найдите порог, где сортировка слиянием работает быстрее двух других.