1 Сортировка слиянием

1.1. Даны два отсортированных по неубыванию массива a и b. Определите, есть ли в них одинаковые числа. Время $O(n)$.

1.2. Даны два отсортированных по неубыванию массива a и b. Найдите такие i и j, что разница $|a_i - b_j|$ минимальна. Время $O(n)$.

1.3. Даны два отсортированных по неубыванию массива a и b и число S. Найдите такие i и j, что сумма $a_i + b_j = S$. Время $O(n)$.

1.4. Даны два отсортированных по неубыванию массива a и b. Найдите число пар (i, j), таких, что $a_i = b_j$. Время $O(n)$.

1.5. Даны два отсортированных по неубыванию массива a и b. Найдите число пар (i, j), таких, что $a_i > b_j$. Время $O(n)$.

1.6. Дан массива a. Найдите число пар (i, j), таких, что $i < j$ и $a_i > a_j$. Время $O(n \log n)$

2 Рекуррентные соотношения

2.1. Решите методом итераций:

(2.1.1) $T(n) = T(n-a) + T(a) + n$ (a — константа).
(2.1.2) $T(n) = 2T(\sqrt{n}) + 1$.

2.2. Докажите по индукции, что если $T(n) = T(n/3) + T(2n/3) + n$, то $T(n) = O(n \log n)$.

2.3. Решите, применив мастер-теорему:

(2.3.1) $T(n) = 3T(n/2) + n$.
(2.3.2) $T(n) = 4T(n/2) + n^2$.
(2.3.3) $T(n) = 4T(n/2) + n^3$.
(2.3.4) $T(n) = 2T(n/2) + n \log n$.

2.4. Пусть время работы алгоритма A описывается соотношением $T_A(n) = 7T_A(n/2) + n^2$, а время работы алгоритма B описывается соотношением $T_B(n) = aT_B(n/4) + n$. При каких значениях a второй алгоритм работает асимптотически быстрее первого?

2.5. Пусть время работы алгоритма описывается соотношением $T(n) = 4T(n/a) + n^a$. При каком значении a время работы будет асимптотически минимальным?

2.6. Докажите по индукции, что если $T(n) = 2T(n/2 + 20) + n$, то $T(n) = O(n \log n)$.

2.7. Докажите по индукции, что если $T(n) = 2T(n/2 + \log n) + n$, то $T(n) = O(n \log n)$.

2.8. Докажите по индукции, что если $T(n) = \log n \cdot T(n/\log n) + n$, то $T(n) = O(n \log n)$.

2.9. Докажите по индукции, что если $T(n) = 2T(n/2) + n$, то $T(n) = \Omega(n \log n)$ (оценка снизу).

2.10. Докажите по индукции, что если $T(n) = 2T(\sqrt{n}) + 1$, то $T(n) = O(\log n)$.

3 Куча, сортировка кучей

3.1. Проиллюстрируйте работу алгоритма сортировки кучей на примере массива [5, 2, 7, 3, 4, 2, 8].

3.2. Пусть в куче лежат числа от 1 до 255, по одному разу каждое. Какое минимальное число может лежать в куче на самом нижнем уровне?

3.3. Пусть в куче лежат числа от 1 до n, по одному разу каждое. В каком случае операция `removeMin` будет работать минимальное, а в каком — максимальное время?

3.4. Пусть дерево кучи организовано таким образом, что у каждой вершины (кроме нижнего слоя) не два ребенка, а три. Какие номера будут у детей вершины i в этом случае?

3.5. Пусть дерево кучи организовано таким образом, что у каждой вершины (кроме нижнего слоя) d детей (при этом d — не константа, а параметр). За какое время работают оснавные операции над кучей в этом случае? Приведите оценку зависимости от n и d.

3.6. Давайте считать, что в куче хранятся не числа, а указатели на элементы, у каждого элемента есть некоторый ключ, по которому элементы сравниваются друг с другом. Что изменится в коде от этого? Что нужно сделать, чтобы можно было быстро по элементу найти его положение в основном массиве кучи?

3.7. Добавьте в кучу операцию `setKey(x, k)`, которая изменяет ключ елемента x, делая его равным k.

3.8. Добавьте в кучу операцию `remove(x)`, которая удаляет произвольные элемент x из кучи.

3.9. Как из двух куч сделать структуру данных, которая одновременно может искать и максимум, и минимум.

3.10. Сортировка называется *устойчивой*, если она не меняет порядок равных элементов (то есть, если в массиве есть два элемента с равными ключами и первый стоит левее второго, то после сортировки первый тоже будет стоять левее второго). Какая из изученных нами сортировок будет устойчивой (и почему), а какая — нет (приведите пример)?

4 Разные задачи про сортировки

4.1. Есть k отсортированных массивов, содержащих в сумме n элементов. Слейте их в один отсортированный массив за время $O(n \log k)$.

4.2. В отсортированном массиве размера n изменили k элементов (неизвестно, каких именно). Отсортируйте полученный массив за $O(n + k \log k)$.

4.3. Дан массив из n чисел от 1 до k, разработайте структуру данных, которая за $O(1)$ отвечает на запросы вида «Сколько в массиве элементов от a до b?». Время на предподсчет $O(n + k)$.

4.4. Как с помощью генератора случайных чисел получить случайную перестановку? Нужно, чтобы каждая перестановка появлялась с вероятностью $1/n!$.

4.5. Дан массив из n чисел. Необходимо для каждого элемента найти число элементов, которые меньше его. Время работы $O(n \log n)$.

4.6. Дано два массива: a и b. Найдите такие i и j, что $a_i < a_j$ и $b_i < b_j$, или скажите, что найти невозможно. Время работы $O(n \log n)$.

4.7. Дан массив из n чисел. Необходимо отсортировать его, совершив не более n операций `swap(a[i], a[j])`. Время работы $O(n \log n)$.

4.8. Как с помощью цифровой сортировки сортировать строки (напрмер, состоящие только из латинских букв) в лексикографическом порядке? За какое время будет работать такая сортировка?

5 Стеки и очереди

5.1. Добавьте в стек и очередь операцию `getSum()`, возвращающую сумму элементов в стеке/очереди. Время работы $O(1)$.Дополнительная память $O(1)$.

5.2. Добавьте в стек операцию `getMin()`, возвращающую минимальный элемент в стеке. Время работы $O(1)$. Дополнительная память $O(size)$ ($size$ — число элементов в стеке).

5.3. Известно, что размер очереди не превосходит n в любой момент времени, но при этом число операций по добавлению/удалению элементов существенно больше n. Как реализовать такую очередь, используя только один массив размера n?.

5.4. Реализуйте стек с помощью очереди (время работы операций не важно).

5.5. Реализуйте очередь с помощью двух стеков (время работы операций не важно).

5.6. Используя стек, научитесь вычислять выражения в префиксной записи (это как постфиксная, только наоборот, например, выражение `4-((1+2)*3)` в постфиксной записи выглядит так: `- 4 * + 1 2 3`. При этом читать строку с выражением нужно слева направо (так бывает нужно делать, например, если строку вам передают по сети и у вас нет памяти, чтобы хранить ее целиком).

5.7. Используя стек, научитесь вычислять выражения в инфиксной записи со скобками (обычные выражения). Для простоты можно считать, что в выражении проставлены все скобки (то есть внутри скобок вычисляется только один оператор, например так можно: $(4 - ((1 + 2) * 3))$, а так — нет: $(1 + 2 + 3)$.

5.8. Добавьте в очередь операцию `getMin()`, возвращающую минимальный элемент в очереди. Время работы $O(\log n)$. Дополнительная память $O(n)$ (n — число элементов в очереди).

6 Связные списки

6.1. Напишите процедуру слияния двух отсортированных односвязных списков в один за время $O(n)$ с $O(1)$ дополнительной памяти.

6.2. Придумайте, как отсортировать связный список за время $O(n \log n)$ с $O(1)$ дополнительной памяти.

6.3. Напишите процедуру, которая разворачивает односвязный список за время $O(n)$ с $O(1)$ дополнительной памяти.

6.4. Кольцевой список — это список, в котором элементы выстроены в виде цикла, каждый элемент хранит ссылку на следующий (и на предыдущий, если это двусвязный список). Для чего может быть удобен такой список? Как с помощью кольцевого списка реализовать обычный линейный список?

6.5. Для экономии памяти в двусвязных списках иногда вместо двух ссылок хранят только их битовый XOR. (например, если `prev[i] = 5`, `next[i] = 3`, то вместо них храним `prevnext[i] = 6`). Как работать с таким списком?

6.6. Дан набор из n элементов, в каждом есть ссылка на следующий. Проверьте, правда ли эти элементы образуют линейный список. (Время $O(n)$, память $O(1)$).

6.7. Дан набор из n элементов, в каждом есть ссылка на следующий. Проверьте, правда ли эти элементы образуют кольцевой список. (Время $O(n)$, память $O(1)$)

6.8. Дан набор из n элементов, в каждом есть ссылка на следующий и предыдущий. Проверьте, правда ли эти элементы образуют несколько связных списков и, если да, объедините их в один (в любом порядке). (Время $O(n)$, память $O(1)$)

6.9. Дан набор из n элементов, в каждом есть ссылка на следующий. Проверьте, правда ли эти элементы образуют несколько кольцевых списков. Выведите (Время $O(n)$, память $O(1)$)

6.10. В функциональных языках программирования нет переменных, то есть когда вы, например, создаете узел списка, поменять в нем ссылку на следующий или предыдущий элемент уже не получится. Как сделать функциональный стек?

7 Амортизационный анализ

7.1. Проанализировать саморасширяющийся массив, если расширение происходит в A раз $(1 < A)$.

7.2. Проанализировать стек на саморасширяющемся массиве, если при полном заполнении происходит увеличение в 2 раза, а при заполнении менее чем на $1/4$ — сужение в 2 раза с помощью метода потеницалов. Потенциал должен зависеть только от текущего состояния стека (размера выделенного массива и числа заполненных элементов) и не должен зависеть от истории операций.

7.3. Пусть выделение массива памяти любого размера происходит за время $O(1)$. Разработайте вектор с истинной (не амортизированной) стоимостью всех операций $O(1)$ и памятью $O(n)$.

7.4. Битовый счетчик хранит число в виде массива двоичных цифр. Изначально все цифры равны 0. Операция `add` увеличивает счетчик на 1. Реализуйте операцию `add` и докажите, что амортизационное время ее работы $O(1)$.

7.5. Добавьте битовому счетчику операцию `clear`, сбрасывающую его в 0. Амортизационная стоимость операции должна быть $O(1)$.

7.6. Добавьте в очередь операцию `getMin()`, возвращающую минимальный элемент в очереди. Амортизированное время работы $O(1)$.

7.7. Реализуйте слудующую структуру данных. Есть матрица из нулей и единиц $n \times n$, изначально заполненная нулями. С ней проделывают операции:

- `setBit(i, j)` — установить значение 1 в ячейке (i, j)
- `clearRow(i)` — установить значение 0 во всех ячейках строки i
- `clearColumn(j)` — установить значение 0 во всех ячейках столбца j
- `getBit(i, j)` — узнать значение в ячейке (i, j)

Амортизационная стоимость всех операции должна быть $O(1)$.

7.8. Реализуйте очередь на базе расширяющегося/сужающегося массива с амортизированной стоимостью всех операций $O(1)$ и памятью $(1 + \varepsilon)n$.

7.9. Структура данных «буферное окно» используется в текстовых редакторах и позволяет перемещать курсор по тексту и добавлять/удалять символы в позиции курсора. Для этого текст хранится в виде строки, в которой на позиции курсора есть окно из незаполненных элементов. Когда пользователь вводит символ, он добавляется на первое незаполненное место и размер буфера уменьшется на 1, когда удаляется — наоборот. Как нужно работать с такой структурой, чтобы амортизированное время добавления/удаления одной буквы было $O(1)$? Чем эта структура лучше/хуже, чем связный список?

8 Деревья поиска

8.1. Напишите рекурсивную процедуру, выводящую элементы дерева поиска в отсортированном порядке, за время $O(n)$.

8.2. Напишите нерекурсивную процедуру, выводящую элементы дерева поиска в отсортированном порядке, за время $O(n)$ с $O(1)$ дополнительной памяти (у узлов есть указатели на родителей).

8.3. Докажите, что нельзя построить дерево поиска из заданных n элементов быстрее, чем за $O(n \log n)$.

8.4. Найти в дереве максимальный элемент, меньший заданного x. Время $O(H)$ (H — высота дерева).

8.5. Для каждого узла x посчитайте число $w(x)$, равное числу узлов в его поддереве (включая сам x). Время $O(n)$.

8.6. Используя вычисленные значения $w(x)$, научитесь находить k-й по возрастанию элемент дерева. Время $O(H)$.

8.7. Используя $w(x)$, научитесь находить по заданному ключу x число элементов, меньших x. Время $O(H)$ (H — высота дерева).

8.8. По заданному узлу найдите его номер по возрастанию среди элементов дерева (у узлов есть указатели на родителей). Время $O(H)$.

8.9. Докажите, что k последовательных команд `x = next(x)` работают за время $O(k + H)$.

8.10. Приведите пример дерева, в котором средняя глубина узла (среднее расстояние от узла до корня) $O(\log n)$, а высота дерева (максимальное расстояние от узла до корня) — $\omega(\log n)$ (асимптотически больше).

8.11. Приведите пример АВЛ-дерева, в котором при добавлении совершится более одного поворота.

8.12. Приведите пример АВЛ-дерева, в котором при удалении совершится более одного поворота.

8.13. Приведите пример двух АВЛ-деревьев, хранящих одно и то же множество элементов, но имеющих разную высоту.

8.14. Приведите пример двух 2-3-деревьев, хранящих одно и то же множество элементов, но имеющих разную высоту.

8.15. По аналогии с 2-3 деревом можно сделать X-Y дерево, в котором у каждого узла, кроме листьев (и, возможно, корня), от количество детей лежит в отрезке X до Y, включительно. Какие условия нужно наложить на X и Y, чтобы можно было эффекивно работать с таким деревом?

8.16. Двоичное дерево поиска называется красно-черным, если каждая его вершина раскрашена либо в красный, либо в черный цвет, родитель любой красной вершины черный, и путь до любого листа содержит одно и то же количество черных вершин. Докажите, что высота такого дерева $O(\log n)$.

8.17. Проверить, что заданное дерево является корректным деревом поиска. Время $O(n)$.

8.18. Петя хочет соптимизировать декартово дерево по памяти. Для этого он решил не хранить ключи Y. Вместо этого он там, где раньше было сравнение ключей Y, делает запрос к генератору случайных чисел. (Например, было `if (a.y < b.y) {...} else {...}`, будет `if (random() < random()) {...} else {...}`. Покажите, что после такого изменения некоторая последовательность действий может привести к тому, что высота дерева (вернее ее матожидание) будет $\Omega(n)$.

9 Дерево отрезков

9.1. Построить дерево отрезков по данному массиву за время $O(n)$.

9.2. Есть массив из n ячеек. Каждая ячейка может быть занятой или свободной. Нужно обрабатывать запросы: 1) пометить ячейку как занятую/свободную, 2):

(9.2.1) найти число свободных ячеек на отрезке от l до r. ($O(\log n)$)

(9.2.2) найти k-ю по порядку свободную ячейку. ($O(\log n)$)

(9.2.3) найти ближайшую к i свободную ячейку. ($O(\log n)$)

9.3. Есть массив a из n чисел. Нужно обрабатывать запросы: 1) присвоить значение a_i, 2):

(9.3.1) найти минимальное i для которого $a_i > k$. ($O(\log n)$)

(9.3.2) вывести все i для которых $a_i > k$. ($O(x \log n)$, где x — размер ответа).

(9.3.3) найти по данным l и r значение суммы $a_l - a_{l+1} + a_{l+2} - a_{l+3} + ... \pm a_{r-1}$. ($O(\log n)$)

(9.3.4) найти по данным l и r значение суммы $a_l + 2a_{l+1} + 3a_{l+2} + ... + (r-l)a_{r-1}$. ($O(\log n)$)

(9.3.5) найти такие l и r, что сумма $a_l + a_{l+1} + ... + a_{r-1}$ максимальна. ($O(\log n)$)

9.4. Есть строка из n круглых скобок. В ней иногда меняются символы.

(9.4.1) после каждого изменения проверять, правильная ли получилась последовательность. ($O(\log n)$)

(9.4.2) отвечать на запросы, является ли правильной скобочной последовательностью подстрока с l до r. ($O(\log n)$)

9.5. Есть подвешенное дерево из n узлов. В каждом узле написано число. Обрабатывать запросы 1) задать значение в узле 2) найти максимум в поддереве. Оба запроса за $O(\log n)$.

9.6. Все приемы, применяемые в дереве отрезков, можно применить и в дереве поиска. Покажите, например, как с помощью сбалансированного дерева поиска (любого на выбор), отвечать на запросы: 1) добавить в множество элемент с ключом k и значением v, 2) удалить элемент с ключом k, 3) найти сумму значений элементов, ключи которых лежат в диапазоне от l до r.

10 Дерево отрезков, массовые операции

10.1. Есть массив a из n булеанов. Нужно обрабатывать запросы за $O(\log n)$:

(10.1.1) присвоить значение x всем элементам отрезка, найти ближайшую к i единицу

(10.1.2) изменить значение всех элементов отрезка на противоположное,
найти число единиц на отрезке

(10.1.3) присвоить значение x всем элементам отрезка,
выполнить `for (i = l .. r - 1) : a[i] = a[i] and a[i - 1]`,
выполнить `for (i = l .. r - 1) : a[i] = a[i] or a[i - 1]`,
найти число единиц на отрезке

(10.1.4) присвоить значение x всем элементам отрезка,
найти число непрерывных отрезков из единиц

10.2. Есть массив a из n чисел. Нужно обрабатывать запросы за $O(\log n)$:

(10.2.1) присвоить значение x всем элементам отрезка,
изменить элементы отрезка $a_i = -a_i$,
найти сумму на отрезке

(10.2.2) присвоить значение x всем элементам отрезка,
изменить элементы отрезка $a_i = -a_i$,
найти максимум на отрезке

(10.2.3) присвоить значение x всем элементам отрезка,
изменить элементы отрезка $a_i = -a_i$,
найти отрезок с максимальной суммой

(10.2.4) присвоить значение x всем элементам отрезка,
найти НОД на отрезке

(10.2.5) присвоить значение x всем элементам отрезка,
найти самый длинный отрезок из одинаковых чисел

(10.2.6) изменить элементы отрезка $a_i = a_i + x \cdot i + y$,
найти сумму на отрезке

(10.2.7) присвоить элементам отрезка значения $a_i = x \cdot i + y$,
найти максимум на отрезке

(10.2.8) присвоить элементам отрезка значения $a_i = x \cdot i + y$,
найти отрезок с максимальной суммой

10.3. Про некоторый массив a из n чисел известно m свойств вида $\max(a_l..a_r) = x$ или $\min(a_l..a_r) = x$. Постройте массив, удовлетворяющий всем свойствам или скажите, что это невозможно.

10.4. Про некоторый массив a из n чисел известно m свойств вида $\sum(a_l..a_r) = x$. Постройте массив, удовлетворяющий всем свойствам или скажите, что это невозможно.

11 Еще задачи на дерево отрезков

11.1. В игре на клеточном поле $n \times n$ игрок сбрасывает на поле бомбы. Каждая бомба наносит урон на некотором прямоугольном участке поля. Нужно обрабатывать запросы: 1) Сбросить бомбу с заданным уроном в заданный прямоугольник, 2) вывести суммарное значение урона в ячейке. Оба запроса за $O(\log^2 n)$.

11.2. Манхеттенское расстояние на плоскости задается как сумма расстояний по координатам $|x_1 - x_2| + |y_1 - y_2|$. Даны координаты n макдональдсов на плоскости. Нужно по запросу (x, y, r) за $O(\log^2 n)$ выдавать число макдональдсов, находящихся на манхеттенском расстоянии не больше r от точки (x, y).

11.3. На огромном шахматном поле стоят n ладей. За $O(\log n)$ проверять, правда ли, что все клетки заданного прямоугольника атакуются хотя бы одной ладьей.

11.4. На биржу приходят два типа запросов: 1) продать x акций по y рублей за штуку, этот запрос кладется в очередь на продажу 2) купить x акций, по этому запросу из очереди жадно достаются x акций с минимальной ценой. Оба запроса за $O(\log n)$ (где n — число запросов в очереди).

11.5. Последовательность f_i вычисляется по следующим правилам: $f_{-1} = f_0 = 1$, $f_i = (a_i \cdot f_{i-1} + b_i \cdot f_{i-2}) \pmod M$. Нужно обрабатывать запросы: 1) для заданного i изменить числа a_i и b_i на x и y (и пересчитать последовательность), 2) найти значение f_i. Оба запроса за $O(\log n)$.

11.6. Есть два массива a и b. Нужно обрабатывать запросы: 1) скопировать участок массива a ав масиив b (то есть, сделать $b_{y+q} = a_{x+q}$ для всех q от 0 до k - 1) 2) найти значение b_i. Оба запроса за $O(\log n)$.

12 NEW! Система непересекающихся множеств

12.1. Добавьте в СНМ операцию `getMin(x)`, возвращающую минимальный элемент в множестве x.

12.2. Дан массив a длины n, заполненный нулями, Делают два вида запросов: 1) $a_i := 1$ 2) найти число непрерывных отрезков из единиц.

12.3. Дан массив a из n положительных чисел. Найдите отрезок с максимальным значением произведения (сумма $\times$ минимум).

12.4. Дан массив a из n положительных чисел. Для каждого числа найдите, для скольки разных отрезков оно является минимумом.

В следующих задачах используются термины из теории графов, которые мы пока не проходили. Но вы не пугайтесь, они все несложные, если не знаете какой-то из них — загуглите.

12.5. В изначально пустой граф из n вершин добавляются одно за другим m ребер. После каждого добавления найдите размер самой большой компоненты связности (по числу вершин).

12.6. Есть пустой граф из n вершин. Делают два вида запросов: 1) добавить ребро 2) найти число ребер в компоненте связности x.

12.7. Есть пустой граф из n вершин. Делают два вида запросов: 1) добавить ребро 2) найти число компонент связности, являющихся деревьями.

12.8. Есть пустой граф из n вершин. Делают два вида запросов: 1) добавить ребро 2) найти число компонент связности, являющихся циклами.

12.9. Есть пустой граф из n вершин. Делают два вида запросов: 1) добавить ребро 2) найти число компонент связности, являющихся двудольными графами.

12.10. Докажите, что если для a и b определены $\log_a^* x$ и $\log_b^* x$, то $\log_a^* x = O(\log_b^* x)$.

12.11. Докажите амотризированную оценку $O(\log n)$ для эвристики сжатия путей (без других эвристик).

12.12. Для кластеризации изображений решают следующую задачу. Есть картинка $n \times m$ пикселей (для простоты будем считать, что каждый пиксель задается одним числом). Мы хотим выделить на ней части примерно одинакового цвета. Для этого нужно разбить картинку на k связных областей так, чтобы минимальная разность значений пикселей на границе областей была максимально возможной.