АиСД у2023. Второй семестр

Домашние задания М3134-М3137

⟨Версия от 24 мая 2024 г.⟩

Темы

Неделя 1. Дерево отрезков — 1

Устная часть

В заданиях с 1.1 по 1.9 дан массив a длины n. Требуется придумать, как при помощи дерева отрезков выполнять две операции. Первая операция — присвоить элементу a_i значение x. Вторая операция описана в каждом задании. Обе операции должны работать за $\mathcal{O}(\log n)$.

1.1. Найти минимум на отрезке $[l, r)$, а также вычислить количество элементов, равных минимуму.

1.2. Найти минимум на отрезке $[l, r)$, а также найти позицию самого левого элемента отрезка, который равен минимуму.

1.3. Найти значение суммы $a_l - a_{l+1} + a_{l+2} - \ldots + (-1)^{r-l} a_{r-1}$.

1.4. Найти значение суммы $a_l + 2a_{l+1} + 3a_{l+2} + \ldots + (r-l)a_{r-1}$.

1.5. Найти подотрезок $[l_1, r_1)$, такой что $l \le l_1 \le r_1 \le r$ и сумма на подотрезке $[l_1, r_1)$ максимальна среди всех таких отрезков. Достаточно найти значение самой суммы, хотя восстановить отрезок также не составит труда.

1.6. Найти минимальное i $(1 \le i \le n)$, такое что $a_i \ge k$. Здесь k — параметр, который задается в запросе. То есть, в разных запросах значения k могут различаться.

1.7. Найти минимальное i на отрезке $[l, r)$ $(l \le i < r)$, такое что $a_i \ge k$. Здесь k, l и r — параметры, которые задаются в запросе. То есть, в разных запросах значения k могут различаться.

1.8. Дан массив из 0 и 1. Найти количество непрерывных отрезков из единиц.

1.9. Дан массив из 0 и 1. Найти самый длинный непрерывный отрезок из единиц.

1.10. Научитесь искать НВП массива длины n за $\mathcal{O}(n \log n)$, используя дерево отрезков. Считайте, что элементы массива — натуральные числа, не превосходящие n.

1.11. Решите задачу 1.10, при условии, что элементы массива — произвольные целые числа.

1.12. Вычислите количество инверсий в массиве длины n за $\mathcal{O}(n \log n)$, используя дерево отрезков.

1.13. Дана строка из n открывающих и закрывающих круглых скобок. Придумайте, как при помощи дерева отрезков отвечать на следующие запросы за $\mathcal{O}(\log n)$. Первый запрос — изменить i-ю скобку. Второй запрос — проверить, является ли скобочная последовательность $a_l a_{l+1} \ldots a_r$ правильной.

1.14. Дана строка из n открывающих и закрывающих круглых скобок. Придумайте, как при помощи дерева отрезков отвечать на следующие запросы за $\mathcal{O}(\log n)$. Первый запрос — изменить i-ю скобку. Второй запрос — найти длину наибольшего префикса отрезка $[l, r)$, который является правильной скобочной последовательностью.

1.15. Дан массив длины n, элементы которого являются натуральными числами, не превосходящими n. Научитесь отвечать на запрос: даны l, r, x и y, требуется вычислить количество элементов на отрезке $[l, r)$, которые лежат в диапазоне от x до y (то есть количество таких i, что $l \le i < r$ и $x \le a_i \le y$). В данной задаче считайте, что все запросы известны заранее, то есть можно решать задачу в Offline. Время работы: $\mathcal{O}((n + q) \log n)$.

1.16. Дан массив длины n, элементы которого являются натуральными числами, не превосходящими n. Научитесь отвечать на запрос: даны l, r, требуется вычислить количество различных элементов, которые встречаются на отрезке $[l, r)$. В данной задаче считайте, что все запросы известны заранее, то есть можно решать задачу в Offline. Время работы: $\mathcal{O}((n + q) \log n)$.

1.17. Марио собирается проходить уровень, состоящий из n последовательно расположенных труб, высота i-й трубы — a_i. Он может переместиться с трубы i на трубу j, если $|i - j| = 1$ и $a_j - a_i \le 1$. Требуется выполнять операции двух типов за $\mathcal{O}(\log n)$:

(a) Определить, может ли Марио добраться от трубы с номером x до трубы с номером y.

(b) Увеличить высоты труб с l-й до r-й на величину d.

Неделя 2. Дерево отрезков — 2

Устная часть

В заданиях с 2.18 по 2.24 дан массив a длины n. Требуется придумать, как при помощи дерева отрезков выполнять указанные операции за $\mathcal{O}(\log n)$.

2.18. (a) Присвоить значение x всем элементам отрезка
(b) Умножить все элементы отрезка на -1 (то есть заменить a_i на $-a_i$)
(c) Найти сумму на отрезке

2.19. (a) Присвоить значение x всем элементам отрезка
(b) Умножить все элементы отрезка на -1 (то есть заменить a_i на $-a_i$)
(c) Найти максимум на отрезке

2.20. (a) Присвоить значение x всем элементам отрезка
(b) Умножить все элементы отрезка на -1 (то есть заменить a_i на $-a_i$)
(c) Найти подотрезок с максимальной суммой

2.21. (a) Присвоить значение x всем элементам отрезка
(b) Прибавить значение x ко всем элементам отрезка
(c) Найти значение элемента

2.22. (a) Присвоить значение x всем элементам отрезка
(b) Прибавить значение x ко всем элементам отрезка
(c) Найти сумму на отрезке

2.23. (a) Заменить на отрезке a_i на $\max(a_i, x)$
(b) Заменить на отрезке a_i на $\min(a_i, x)$
(c) Найти значение элемента

2.24. (a) Присвоить значение x всем элементам отрезка
(b) Найти подотрезок максимальной длины, состоящий из одинаковых чисел

В заданиях с 2.25 по 2.28 дан массив a длины n, состоящий из булевых значений. Требуется придумать, как при помощи дерева отрезков выполнять указанные операции за $\mathcal{O}(\log n)$.

2.25. (a) Присвоить значение x всем элементам отрезка
(b) Найти ближайшую к i-му элементу единицу

2.26. (a) Изменить все значения на отрезке на противоположные
(b) Найти количество единиц на отрезке

2.27. (a) Присвоить значение x всем элементам отрезка
(b) Найти количество непрерывных отрезков из единиц

2.28. (a) Присвоить значение x всем элементам отрезка

(b) Найти самый длинный непрерывный отрезок из единиц

2.29. Даны n прямоугольников на плоскости, стороны которых параллельны осям координат. Найдите точку, покрытую максимальным количеством прямоугольников (если таких точек несколько, можно найти любую). Время работы: $\mathcal{O}(n \log n)$.

2.30. Даны n прямоугольников на плоскости, стороны которых параллельны осям координат. Координаты углов прямоугольников не превосходят C. Найдите площадь той части плоскости, которая покрыта хотя бы одним прямоугольником. Время работы: $\mathcal{O}(n \log n + n \log C)$.

2.31. Решите предыдущую задачу за $\mathcal{O}(n \log n)$.

2.32. Даны n прямоугольников на плоскости, стороны которых параллельны осям координат. Найдите площадь той части плоскости, которая покрыта максимальным количеством прямоугольников. Время работы: $\mathcal{O}(n \log n)$.

2.33. Даны n прямоугольников на плоскости, стороны которых параллельны осям координат, и m точек. Для каждого прямоугольника вычислите, сколько точек он покрывает. Время работы: $\mathcal{O}((n + m) \log(n + m))$.

2.34. Дан массив длины n. Вычислите количество возрастающих подпоследовательностей массива длины k. Время $\mathcal{O}(kn \log n)$.

2.35. Есть шахматное поле $n \times n$. Требуется обрабатывать следующие запросы за $\mathcal{O}(\log n)$:

(a) Добавить ладью на поле

(b) Удалить ладью с поля

(c) Найти количество клеток, которые не бьет ни одна ладья

Неделя 3. Двоичное дерево поиска. AVL-дерево

Устная часть

3.36. Придумайте, как реализовать рекурсивную функцию, выводящую все элементы двоичного дерева поиска в порядке сортировки, за $\mathcal{O}(n)$.

3.37. Придумайте, как реализовать нерекурсивную функцию, выводящую все элементы двоичного дерева поиска в порядке сортировки, за $\mathcal{O}(n)$, используя $\mathcal{O}(1)$ дополнительной памяти. Можно считать, что у каждой вершины хранится ссылка на родительскую вершину.

3.38. Докажите, что не существует алгоритма, который строит двоичное дерево поиска по заданному набору ключей быстрее, чем за $\Omega(n \log n)$ в худшем случае.

3.39. Научитесь находить k-й в порядке возрастания ключ в двоичном дереве поиска за $\mathcal{O}(H)$. Разрешается хранить в каждой вершине $\mathcal{O}(1)$ дополнительной памяти.

3.40. Научитесь по данному ключу x находить количество элементов в дереве, не превосходящих x за $\mathcal{O}(H)$. Разрешается хранить в каждой вершине $\mathcal{O}(1)$ дополнительной памяти.

3.41. По заданной вершине в дереве найдите следующие k вершин в порядке сортировки за $\mathcal{O}(H + k)$.

3.42. Приведите пример дерева, в котором средняя глубина вершины (среднее расстояние от корня до вершины) $\mathcal{O}(\log n)$, а высота дерева (максимальное расстояние от корня до вершины) асимптотически больше, чем $\log n$.

3.43. Приведите пример AVL-дерева и операции добавления элемента, при которой операцию балансировки будет выполнена более, чем в одной вершине.

3.44. Приведите пример AVL-дерева и операции удаления элемента, при которой операция балансировки будет выполнена более, чем в одной вершине.

3.45. Приведите пример двух AVL-деревьев, хранящих одно множество ключей, но имеющих разную высоту.

3.46. Научитесь проверять, что заданное двоичное дерево является корректным двоичным деревом поиска.

3.47. Пусть в двоичном дереве поиска для каждой вершины высота ее детей отличается не более, чем на 5. Правда ли, что высота такого дерева $\mathcal{O}(\log n)$?

3.48. Пусть в двоичном дереве поиска для каждой вершины высота ее детей отличается не более, чем в два раза. Правда ли, что высота такого дерева $\mathcal{O}(\log n)$?

3.49. Научитесь при помощи AVL-дерева отвечать на следующие запросы за $\mathcal{O}(\log n)$:

(a) Добавить число x в множество

(b) Удалить число x из множества

(c) Вычислить сумму всех x из множества, таких что $l \leq x \leq r$

3.50. Рассмотрим два AVL-дерева. Пусть любой ключ в первом дереве меньше, чем любой ключ во втором дереве. Предложите способ слить эти два дерева в одно AVL-дерево за истинные $\mathcal{O}(\log n)$, где n — сумма размеров деревьев.

3.51. Модифицируйте AVL-дерево таким образом, чтобы в каждой вершине хранить, помимо ссылок на детей, $\mathcal{O}(1)$ бит дополнительной информации (обратите внимание, что стандартная реализация с хранением высот вершин использует больше памяти).

Неделя 4. Splay-дерево

Устная часть

4.52. Докажите по аналогии с другими операциями, что операция `зиг-заг` имеет амортизированное время работы $\widetilde{T}(\texttt{зиг-заг}(x)) = 3 \cdot (r'(x) - r(x))$.

4.53. Пусть была сделана операция `splay` для вершины x, находящейся на глубине h. Как изменится сумма глубин вершин на пути от вершины x до корня дерева?

4.54. Упростим операцию `splay`: будем всегда делать операцию `зиг`, забыв про существование `зиг-зиг` и `зиг-заг`. Покажите, что существует последовательность из n операций, которая в этом случае будет работать дольше, чем за $n \log n$.

4.55. Упростим операцию `splay`: будем вместо операции `зиг-заг` делать операцию `зиг-зиг`. Покажите, что существует последовательность из n операций, которая в этом случае будет работать дольше, чем за $n \log n$.

4.56. Что можно сказать о времени работы Splay дерева, если вместо операции `зиг-зиг` делать операцию `зиг-заг`?

4.57. Пусть в Splay дереве находятся числа от 1 до n. За какое суммарное время отработает последовательность операций: `find(1)`, `find(2)`, ..., `find(n)`?

4.58. Пусть в Splay дереве находятся числа от 1 до n. Будем делать m операций в следующей последовательности: `splay(1)`, `splay(n)`, `splay(1)`, `splay(n)`, ... За какое суммарное время они отработают?

4.59. Пусть в Splay дереве размера n вы совершаете много операций над небольшим подмножеством элементов размера k. Как будет выглядеть дерево? За какое время будут работать операции?

4.60. Пусть в Splay дереве находятся числа от 1 до n. Далее поступают m запросов, суммарное количество запросов к i-му элементу равно p_i. Докажите, что суммарное время всех запросов равно $\mathcal{O}(m + \sum p_i \log \frac{m}{p_i})$. Подсказка: на лекции мы ввели $s(x)$ как количество вершин в поддереве вершины x. Теперь можно придумать для каждой вершины некоторый вес $w(x)$ (подумайте, какой) и обозначить за $s(x)$ сумму весов всех вершин в поддереве вершины x.

4.61. Придумайте способ построить Splay дерево за $\mathcal{O}(n)$ по заданному отсортированному массиву.

4.62. Пусть в Splay дереве находятся числа от 1 до n. Далее поступают m запросов к вершинам $x_1, x_2, \ldots, x_m$. При этом известно, что $|x_i - x_{i-1}| \le d$. Докажите, что суммарное время запросов равно $\mathcal{O}(m \log d)$.

4.63. Докажите, что если сделать операцию `splay` ко всем элементам в порядке от 1 до n, то образуется дерево, у каждой внутренней вершины которого есть только левый сын.

4.64. Постройте пример Splay дерева из хотя бы семи вершин, которое после применения операции `splay` к одному из самых глубоких листьев становится бамбуком.

Неделя 5. Декартово дерево

Устная часть

5.65. Дан массив пар (x, y), отсортированный по возрастанию значений x. Постройте по нему декартово дерево за $\mathcal{O}(n)$.

5.66. Можно ли построить декартово дерево за $\mathcal{O}(n)$, если массив пар не отсортирован по x?

5.67. Дан массив пар (x, y). Все x попарно различны, а y могут совпадать. Как проверить, что декартово дерево можно построить единственным образом?

5.68. Дан массив пар (x, y). Все x попарно различны, а y могут совпадать. Необходимо вычислить количество способов построить декартово дерево.

5.69. Придумайте, как сделать СНМ на базе дерева поиска по неявному ключу с временем работы операций $\mathcal{O}(\log n)$.

5.70. Пусть у дерева поиска нет вершин с одним ребенком (то есть у всех вершин 0 или 2 детей). Правда ли, что высота такого дерева $\mathcal{O}(\log n)$?

5.71. Пусть у дерева поиска размер каждого поддерева равен $2^k - 1$ для некоторого целого k (для разных вершин могут быть разные значения k). Правда ли, что высота такого дерева $\mathcal{O}(\log n)$?

5.72. Дан массив чисел длины n. При помощи декартова дерева научитесь выполнять следующие операции за $\mathcal{O}(\log n)$:

(a) Развернуть отрезок массива $[l, r]$ задом наперед

(b) Найти минимум на отрезке $[l, r]$

5.73. Дан массив чисел длины n. При помощи декартова дерева научитесь выполнять следующие операции за $\mathcal{O}(\log n)$:

(a) Для данных l и r поменять местами следующие пары элементов: l и $l + 1$, $l + 2$ и $l + 3$, и так далее до r

(b) Найти минимум на отрезке $[l, r]$

5.74. Будем строить декартово дерево, не храня приоритеты. В тех местах, где производится сравнение приоритетов, будем выбирать случайный вариант с вероятностью $\frac{1}{2}$. Покажите, что при такой реализации декартова дерева некоторая последовательность операций может привести к тому, что высота дерева станет $\Omega(n)$.

5.75. В дереве поиска, в отличие от дерева отрезков, исходные элементы множества хранятся не только в листьях, но и в промежуточных узлах. Иногда это неудобно, поэтому делают другую версию дерева поиска: исходные элементы множества хранятся только в листьях, а в промежуточных вершинах хранится максимальный ключ в поддереве. Покажите, как осуществлять операции поиска и добавления элемента в таком дереве.

5.76. Дерево поиска вывели на экран при помощи такой функции:

```
print_tree(t):
    if (t == null):
        return
    print(t.key)
    print_tree(t.l)
    print_tree(t.r)
```

По массиву, который вывела данная процедура, восстановите дерево поиска за $\mathcal{O}(n)$.

5.77. Даны n окружностей. Они не пересекаются, но могут быть вложены друг в друга. Постройте дерево вложенности окружностей за $\mathcal{O}(n \log n)$.

Неделя 6. Дерево Фенвика. Sparse Table

Устная часть

6.78. На лекции был рассмотрен способ построить дерево Фенвика за $\mathcal{O}(n)$. Научитесь строить дерево Фенвика за $\mathcal{O}(n)$ с использованием $\mathcal{O}(1)$ дополнительной памяти.

6.79. Дано построенное дерево Фенвика для некоторого массива. Восстановите по нему исходный массив, используя $\mathcal{O}(1)$ дополнительной памяти.

6.80. Как изменится время работы и необходимый объем памяти Sparse Table, если хранить значения функции на отрезках длины x^k, а не 2^k $(x > 2)$? Приведите оценки, зависящие от n и x.

6.81. Добавьте в дерево Фенвика операцию, позволяющую найти префикс максимальной длины, сумма на котором не превосходит x (все числа в массиве неотрицательные). Время работы: $\mathcal{O}(\log n)$.

6.82. Научитесь отвечать на запросы за $\mathcal{O}(\log n)$ с использованием дерева Фенвика:

(a) Прибавить x ко всем элементам отрезка $[l, r)$.

(b) Найти значение элемента a_i.

6.83. Научитесь отвечать на запросы за $\mathcal{O}(\log n)$ с использованием дерева Фенвика:

(a) Прибавить x ко всем элементам отрезка $[l, r)$.

(b) Найти сумму элементов на отрезке $[l, r)$.

6.84. Научитесь отвечать на запросы за $\mathcal{O}(\log n)$ с использованием дерева Фенвика:

(a) Найти минимум на отрезке $[0, r)$.

(b) Заменить i-й элемент на число d.

Можно ли решать данную задачу для произвольных значений d?

6.85. Научитесь обрабатывать следующие запросы при помощи дерева Фенвика за $\mathcal{O}(\log n)$:

(a) Присвоить значение d всем элементам отрезка $[i, n)$.

(b) Найти значение i-го элемента массива.

6.86. Последовательность f_n задана следующим образом: $f_{-2} = 1$, $f_{-1} = 1$, $f_n = (a_n \cdot f_{n-1} + b_n \cdot f_{n-2}) \pmod M$ для всех $n \geq 0$. Научитесь обрабатывать следующие запросы за $\mathcal{O}(\log n)$:

(a) Заменить пару (a_i, b_i) на пару (x, y).

(b) Вычислить f_i для данного i.

6.87. Даны два массива a и b длины n. Научитесь обрабатывать следующие запросы за $\mathcal{O}(\log n)$:

(a) Скопировать участок массива a длины k в массив b. То есть, заменить b_{x+i} на a_{y+i} для всех i от 0 до $k-1$.

(b) Найти значение b_i.

6.88. Даны массивы a и b длины n. Найдите количество отрезков $[l, r)$, таких что $\min(a_l, a_{l+1}, \ldots, a_{r-1}) = \max(b_l, b_{l+1}, \ldots, b_{r-1})$. Время работы: $\mathcal{O}(n \log n)$.

6.89. Дан массив a длины n. Найдите количество отрезков $[l, r)$, таких что $(a_l \text{ or } a_{l+1} \text{ or } \ldots \text{ or } a_{r-1}) > \max(a_l, a_{l+1}, \ldots, a_{r-1})$. Здесь or означает побитовое «ИЛИ». Время работы: $\mathcal{O}(n \log n \log C)$, где C — максимальное из a_i. Подсказка: используйте Sparse Table!

6.90. Рассмотрим следующий код альтернативной реализации дерева Фенвика:

```
sum(r):
    i = r + 1
    ans = 0
    while i > 0:
        ans += t[i]
        i -= i & -i
    return ans

add(pos, d):
    i = pos + 1
    while i <= n:
        t[i] += d
        i += i & -i
```

Здесь массив t имеет длину $n+1$, а i — signed переменная. Докажите, что данные функции работают за $\mathcal{O}(\log n)$ и являются корректной реализацией дерева Фенвика.

6.91. Подумайте, безопасно ли писать код из предыдущего задания на языке C++?

Неделя 7. Хеш-таблицы

Устная часть

7.92. Будем разрешать коллизии в хеш-таблице с помощью списков, но хранить списки в отсортированном по ключам порядке. Как это повлияет на время работы в худшем случае и в среднем?

7.93. Добавьте в хеш-таблицу возможность проитерироваться по всем ее элементам в том порядке, в котором они добавлялись за $\mathcal{O}(n)$.

7.94. Множество хеш-функций называется 2-универсальным, если для любой пары ключей (x, y) $(x \neq y)$ пара $(h(x), h(y))$ может принять любую из возможных m^2 пар значений равновероятно (будем считать, что число m — простое). Докажите, что множество хеш-функций $h_a(x) = ax \pmod m$ не является 2-универсальным.

7.95. Докажите, что множество хеш-функций $h_{ab}(x) = ax + b \pmod m$ является 2-универсальным.

7.96. Научитесь удалять ключи из хеш-таблицы с открытой адресацией за $\mathcal{O}(1)$ в среднем.

7.97. Найдите реализацию метода `Long.hashCode()` в языке Java. Как быстро сгенерировать много объектов типа `Long` с одинаковым хешом?

7.98. Найдите реализацию метода `String.hashCode()` в языке Java. Как быстро сгенерировать много объектов типа `String` с одинаковым хешом?

7.99. На лекции было доказано, что если для произвольной пары (x, y) $(x \neq y)$ $p(h(x) = h(y)) = \frac{1}{m}$, то операции с хеш-таблицей в среднем работают за $\mathcal{O}(\frac{n}{m})$, где n — количество элементов в таблице. Рассмотрим другой критерий для хеш-функции. Пусть $p(h(x) = k) = \frac{1}{m}$ для любого k. Хорош ли такой критерий? Гарантирует ли он, что операции с хеш-таблицей будут работать за $\mathcal{O}(\frac{n}{m})$? Если да, докажите. Если нет, приведите пример плохой хеш-функции, для которой данный критерий выполняется. Хорош ли такой критерий, если гарантируется, что ключи, добавляемые в хеш-таблицу, выбираются случайно?

7.100. Придумайте, как реализовать операцию `merge` для объединения двух множеств, хранящихся в виде хеш-таблиц. Амортизированная стоимость данной операции должна быть $\mathcal{O}(\log n)$.

7.101. Пусть изначально мы не знаем, сколько элементов будут добавлены в хеш-таблицу. Придумайте способ реализовать хеш-таблицу таким образом, чтобы амортизированная стоимость добавления элемента в среднем равнялась $\mathcal{O}(1)$.

7.102. Какой размер множества одинаковых равномерно распределенных от 1 до n случайных величин необходим, чтобы вероятность того, что хотя бы две из них совпадут, была хотя бы $\frac{1}{2}$? Сделайте вывод о вероятности коллизии в хеш-таблице, если при ее реализации мы игнорируем коллизии.

Неделя 8. LCA

Устная часть

8.103. Дано дерево. Заранее известны q запросов вида: «прибавить ко всем ребрам на пути от u до v число x». Выведите вес каждого ребра после выполнения всех запросов. Подсказка: вспомните аналогичную задачу на массиве и используйте ту же технику.

8.104. Дано дерево, которое иногда меняется. Имеются две операции:

(a) Добавить новую вершину u и подвесить ее к вершине v

(b) Удалить лист дерева с номером u

Покажите, что можно пересчитывать двоичные подъемы без потери производительности по времени.

8.105. Дано дерево, на каждом ребре написано число. Требуется отвечать на запросы: «найти сумму чисел на пути от u до v» за $\mathcal{O}(\log n)$. Числа не меняются.

8.106. Дано дерево, на каждом ребре написано число. Требуется отвечать на запросы: «найти минимум чисел на пути от u до v» за $\mathcal{O}(\log n)$. Числа не меняются.

8.107. Дано дерево, у каждого ребра есть длина. Требуется отвечать на запросы: «найти ближайшего предка v, длина пути до которого не меньше d» за $\mathcal{O}(\log n)$. Длины не меняются.

8.108. Дано дерево, у каждого ребра есть длина. Требуется отвечать на запросы: «найти середину пути от u до v» за $\mathcal{O}(\log n)$. Длины не меняются. Середина пути — это либо вершина, либо точка на ребре. Во втором случае достаточно найти ребро, на котором находится середина.

8.109. Дано дерево. Требуется отвечать на запросы: «найти длину пересечения двух путей» за $\mathcal{O}(\log n)$. Пути задаются парами вершин (u_1, v_1) и (u_2, v_2).

8.110. Дано подвешенное дерево. Требуется отвечать на запросы двух типов за $\mathcal{O}(\log n)$:

(a) Найти LCA вершин u и v

(b) Взять целиком поддерево вершины u и подвесить его к вершине v

8.111. Дано подвешенное дерево. Требуется отвечать на запросы двух типов за $\mathcal{O}(\log n)$:

(a) Найти LCA вершин u и v

(b) Переподвесить дерево за вершину v

8.112. Дано подвешенное дерево. Требуется отвечать на запросы: «дано множество из k вершин, найти их LCA» за $\mathcal{O}(k \log n)$.

8.113. Дано дерево, у каждого ребра есть длина. Требуется отвечать на запросы двух типов за $\mathcal{O}(\log n)$ (препроцессинг $\mathcal{O}(n)$):

(a) Изменить длину ребра (u, v)

(b) Найти расстояние между вершинами u и v

8.114. Дано дерево. Требуется отвечать на запросы: «даны две вершины u и v, найдите количество вершин в дереве, равноудаленных от вершин u и v» за $\mathcal{O}(\log n)$.

8.115. Дано подвешенное дерево. Требуется отвечать на запросы: «является ли вершина u предком вершины v» за $\mathcal{O}(1)$. Препроцессинг: $\mathcal{O}(n)$.

Неделя 9. Задачи на деревья

Устная часть

9.116. Дано ориентированное дерево на n вершинах (у каждого ребра есть ориентация, можно перемещаться только в одну сторону). Необходимо отвечать на запросы: «можно ли добраться из вершины u в вершину v, учитывая ориентацию ребер» за $\mathcal{O}(\log n)$.

9.117. Независимым множеством называется множество вершин, таких что никакие две вершины множества не соединены ребром. Найдите максимальное по размеру независимое множество в дереве за $\mathcal{O}(n)$.

9.118. Паросочетанием называется множество ребер, такое что никакие два ребра из множества не имеют общей вершины. Найдите максимальное по размеру паросочетание в дереве за $\mathcal{O}(n)$.

9.119. Дано дерево из n вершин. В вершине с номером 1 располагается ваш дом, а в листьях дерева находятся рестораны. В некоторых вершинах дерева сидят коты (вам известно, в каких именно вершинах). Вы хотите посетить ресторан, и для этого можете перемещаться по ребрам дерева. Однако, вы можете выбирать только такие маршруты, чтобы не посещать **подряд** m вершин, в которых находятся коты, так как у вас аллергия. Найдите количество ресторанов, до которых вы сможете добраться за $\mathcal{O}(n)$.

9.120. Дано дерево на n вершинах, у каждого ребра есть длина. Диаметром называется наидлиннейший простой путь в дереве. Научитесь искать диаметр в дереве за $\mathcal{O}(n)$.

9.121. Два дерева T_1 и T_2 называются изоморфными, если существует такая биекция f из множества вершин первого дерева в множество вершин второго дерева, что в дереве T_1 существует ребро (u, v) тогда и только тогда, когда в дереве T_2 существует ребро $(f(u), f(v))$. Два подвешенных дерева называются изоморфными, если f также переводит корень первого дерева в корень второго дерева. Научитесь проверять подвешенные деревья на изоморфизм за $\mathcal{O}(n \log n)$.

9.122. Научитесь проверять неподвешенные деревья на изоморфизм за $\mathcal{O}(n \log n)$.

9.123. Дано дерево на n вершинах. Необходимо записать в каждой вершине число $f(v)$, не превосходящее $\log n$, таким образом, чтобы было выполнено следующее условие. Для любых двух вершин u и v, таких что $u \neq v$ и $f(u) = f(v)$ существует такая вершина w, лежащая на пути из u в v, что $f(w) > f(u)$. Время работы: $\mathcal{O}(n)$.

9.124. Дано дерево на n вершинах, у каждого ребра есть вес. Научитесь отвечать на запросы «найти ребро минимального веса на пути из u в v» за $\mathcal{O}(1)$ с препроцессингом за $\mathcal{O}(n)$.

9.125. Дано дерево на n вершинах. Изначально все вершиины не помечены. Требуется отвечать на запросы за $\mathcal{O}(\log n)$ в Online:

(a) Пометить вершину v

(b) Найти самого глубокого не помеченного общего предка вершин u и v

Препроцессинг: $\mathcal{O}(n \log n)$.

9.126. На лекции про LCA мы научились сводить задачу LCA к задаче RMQ при помощи Эйлерова обхода дерева. Придумайте способ выполнить обратное сведение. Иными словами, дан произвольный массив. Требуется построить некоторое дерево таким образом, чтобы запрос минимума на любом подотрезке массива соответствовал нахождению LCA некоторой пары вершиин. Выполнить сведение (то есть построить дерево) необходимо за $\mathcal{O}(n)$.

9.127. В дереве каждый лист покрашен в какой-то цвет. За время $\mathcal{O}(n \log n)$ определите для каждой вершины количество различных цветов в ее поддереве.

9.128. По аналогии с Long Path Decomposition у каждой вершины выберем ребро, ведущее в ребенка с максимальным размером поддерева (в Long Path Decomposition мы выбирали вместо этого ребенка с максимальной высотой поддерева). Докажите, что для любой вершины v на пути из v в корень дерева мы посетим не более, чем $\mathcal{O}(\log n)$ различных путей.

Неделя 10. Heavy-Light декомпозиция и задачи на деревья

Устная часть

10.129. Какое максимальное и минимальное количество путей может получиться в Heavy-Light декомпозиции дерева?

10.130. Будем выбирать в качестве тяжелого ребра ребро, ведущее в поддерево, имеющее максимальную высоту (а не максимальное количество вершин). Оцените количество легких ребер, которое может быть на пути от вершины до корня дерева.

10.131. Пусть веса в дереве есть не на вершинах, а на ребрах. Покажите, как можно решать задачи при помощи HLD в таком случае.

10.132. Научитесь обрабатывать следующие запросы:

(a) Изменить вес ребра

(b) Найти самый далекий от корня лист

10.133. Дано дерево, каждое ребро которого может быть включено или выключено. Научитесь обрабатывать следующие запросы:

(a) Изменить состояние всех ребер на пути от вершины u до вершины v

(b) Найти количество компонент связности, если перемещаться можно только по включенным ребрам

10.134. Дано дерево, каждое ребро которого может быть включено или выключено. Научитесь обрабатывать следующие запросы:

(a) Изменить состояние всех ребер на пути от вершины u до вершины v

(b) Найти самый длинный путь из корня, если перемещаться можно только по включенным ребрам

10.135. Дано дерево. Требуется для каждой вершины вычислить сумму расстояний от данной вершины до всех остальных. Время работы: $\mathcal{O}(n)$.

10.136. Дерево поджигают в некоторой вершине. Огонь распространяется по ребру за одну единицу времени. Вычислите для каждой вершины, за сколько времени сгорит все дерево, если поджечь его в данной вершине.

10.137. Научитесь отвечать на следующие запросы: «дерево подожгли в вершинах u и v, найти время, за которое все дерево сгорит».

10.138. Покажите, как при помощи HLD вычислять значение не коммутативной функции на пути.

10.139. Научитесь при помощи HLD вычислять значение функции на пути, без использования алгоритмов поиска LCA.

10.140. *Центроидом* называется вершина дерева, при удалении которой размеры всех оставшихся кусков дерева не превосходят $\frac{n}{2}$. Найдите центроид дерева за $\mathcal{O}(n)$.

10.141. Докажите, что любое дерево имеет не более двух центроидов.

10.142. Дано подвешенное дерево на n вершинах. Для каждой вершины найдите центроид ее поддерева. Время работы: $\mathcal{O}(n)$.

Неделя 11. Корневые оптимизации

Устная часть

11.143. Пусть дана задача о рюкзаке: даны n предметов с весами $w_1, w_2, \ldots, w_n$. Требуется определить, можно ли набрать суммарный вес, в точности равный W. Также известно, что $\sum w_i = S$. Мы уже умеем решать задачу за $\mathcal{O}(n \cdot W)$. Научитесь решать задачу за $\mathcal{O}(S\sqrt{S})$.

11.144. Дан неориентированный граф на n вершинах и m ребрах. В каждой вершине записано целое число. Необходимо отвечать на запросы двух типов:

- ▷ Прибавить ко всем соседям вершины v число x
- ▷ Вывести значение, хранящееся в вершине v

Асимптотика: $\mathcal{O}((m+q)\sqrt{m})$.

11.145. Дано дерево на n вершинах. Изначально все вершины не покрашены. Необходимо отвечать на запросы двух типов:

- ▷ Покрасить вершину v.
- ▷ Найти ближайшую к v покрашенную вершину.

Асимптотика: $\mathcal{O}((n+q)\sqrt{n+q})$. Можно сделать препроцессинг.

11.146. Дан массив из n целых чисел. Требуется ответить на q запросов: «найти MEX множества чисел $a_l, a_{l+1}, \ldots, a_r$». MEX множества — это минимальное неотрицательное целое число, не содержащееся в множестве. Время работы: $\mathcal{O}((n+q)\sqrt{n+q})$.

11.147. Решите предыдущую задачу на дереве (в каждой вершине записано целое число, нужно находить MEX на пути).

11.148. Дан массив из n целых чисел и q запросов одного из трех типов:

- ▷ Найти минимум на отрезке $[l, r]$.
- ▷ Развернуть подотрезок массива $[l, r]$.
- ▷ Прибавить ко всем элементам отрезка $[l, r]$ число x.

Время работы: $\mathcal{O}((n+q)\sqrt{n+q})$.

11.149. Дан массив из n целых чисел. Есть q запросов к массиву, каждый запрос описывается парой целых чисел l_j и r_j. Для каждого запроса (l_j, r_j) необходимо посчитать количество таких чисел x, что число x встречается ровно x раз среди чисел $a_{l_j}, a_{l_j+1}, \ldots, a_{r_j}$.

11.150. Имеется массив натуральных чисел a длины n. Рассмотрим некоторый его подмассив $a_l, a_{l+1}, \ldots, a_r$, и для каждого натурального числа s обозначим через K_s число вхождений числа s в этот подмассив. Назовем *мощностью* подмассива сумму произведений $K_s \cdot K_s \cdot s$ по всем различным натуральным s. Научитесь находить мощность подмассива для заданных l и r.

11.151. Есть n лунок, расположенных в ряд, пронумерованных слева направо числами от 1 до n. У каждой лунки изначально установлена своя сила выброса (у лунки с номером i она равна a_i). Если вбросить шарик в лунку i, то он тут же вылетит из нее и попадет в лунку $i + a_i$, после чего он опять вылетит и так далее. Если же лунки с таким номером нет, то он просто вылетит за край ряда. Есть q запросов:

- ▷ Установить силу выброса лунки a равной b;
- ▷ Вбросить шарик в лунку a и посчитать количество прыжков шарика, прежде чем он вылетит за край ряда, а так же выяснить номер лунки, после выпрыгивания из которой шарик вылетел за край поля.