АиСД 2023. Первый семестр

Задания на практики M3138-M3139

⟨Версия от 7 января 2024 г.⟩

Темы

Неделя 1. Асимптотики

Определения, которые были на лекции:

▷ $f(n) = \mathcal{O}(g(n))$, если существует константа c, что, начиная с некоторого места, f ограничена сверху $c \cdot g$, или же

$$\exists c > 0, n_0 > 0 : \ \forall n > n_0 : \ f(n) \leqslant c \cdot g(n)$$

▷ $f(n) = o(g(n))$, если для любой константы c, начиная с некоторого места, f ограничена сверху $c \cdot g$, или же

$$\forall c > 0 : \ \exists n_0 > 0 : \ \forall n > n_0 : \ f(n) < c \cdot g(n)$$

Альтернативное определение:

$$\frac{f(n)}{g(n)} \underset{n \to +\infty}{\longrightarrow} 0$$

Стоит отметить, что знак '=' в данном случае не очень корректен. Формально, $\mathcal{O}(n)$ – это *множество функций*, ограниченных сверху линейной. Поэтому корректно было бы писать $f(n) \in \mathcal{O}(g(n))$, но мы будем пользоваться традиционно принятым '=', подразумевая это.

Практика

1.0. Напишите аналогичные написанным выше определения для Ω, Θ и ω. Объясните, почему записанные вами определения соответствуют их смыслу.

1.1. **Альтернативные определения**.

(a) Покажите, что определение $\mathcal{O}$ останется равносильным исходному, если поменять $\leqslant$ на $<$.

(b) Покажите, что определение o останется равносильным исходному, если поменять $<$ на $\leqslant$.

(c) Покажите, что если мы рассматриваем только целые n, то из определения $\mathcal{O}$ можно убрать условие на n_0, то есть что

$$f(n) = \mathcal{O}(g(n)) \iff \exists c > 0 : \ \forall n \ f(n) \leqslant c \cdot g(n).$$

1.2. Для каждой из рассматриваемых дальше функций $f(n)$ найдите наиболее компактно записываемую $g(n)$, что $f(n) = \Theta(g(n))$, и докажите это соотношение.

(a) $f(n) = \log(\sqrt{n}) + \sqrt{\log n}$

(b) $f(n) = n \cdot 3^{n+1} + n^{10}$

(c) $f(n) = \frac{10n^2+2}{7n-1}$

(d) $f(n) = \log(2n \log n)$

1.3. Верно ли, что $n = o\big((\log n)^{\log n}\big)$? Докажите или опровергните.

1.4. Приведите пример двух положительных функций f и g, для которых не выполняется ни $f = \mathcal{O}(g)$, ни $g = \mathcal{O}(f)$.

Уточнение: не привязывайтесь к тому, что мы обычно рассматриваем в качестве функции время работы алгоритма. Подойдут любые положительные функции.

1.5. Докажите, что $\sum\limits_{t=1}^{n} \frac{1}{t} = \Omega(\log n)$. Пользоваться тем, что гармонический ряд приближается логарифмом, без доказательства, **нельзя**.

1.6. Докажите *по определению* следующие утверждения.

(a) Для любых f и g верно, что $f(n) + g(n) = \mathcal{O}(\max(f(n), g(n)))$.

(b) Если $f_1(n) = \mathcal{O}(f_2(n))$ и $g_1(n) = \mathcal{O}(g_2(n))$, то $f_1 + g_1 = \mathcal{O}(f_2 + g_2)$.

(c) Если $f(n)$ – полином степени d, то $f(n) = \Theta(n^d)$.

1.7. **Нестандартные преобразования**. Для каждого из следующих утверждений докажите его или приведите контрпример. Если утверждение не верно, но будет верно при каком-то дополнительном условии, найдите это условие.

(a) Для $g(n) = f(n)^2$ верно $f(n) = \mathcal{O}(g(n))$

(b) Для любой f верно $f(n) = \Omega(f(\frac{n}{2}))$

(c) Если $f(n) = \mathcal{O}(g(n))$, то $e^f = \mathcal{O}(e^g)$

(d) Если $f(n) > 1$ для всех n и $f(n) = o(g(n))$, то $e^f = o(e^g)$

(e) Если $f(n) = \mathcal{O}(g(n))$, то $\log_2(f) = \mathcal{O}(\log_2(g))$

(f) Если $f(n) = o(g(n))$, то $\log_2(f) = o(\log_2(g))$

1.8. Время работы некоторого алгоритма задано следующим рекуррентным соотношением. Найдите Θ-асимптотику времени работы этого алгоритма, *построив дерево рекурсивных вызовов*.

(a) $T(n) = T(n-1) + 2n^2$

(b) $T(n) = 3T(\frac{n}{2}) + 2$

(c) $T(n) = T(\frac{n}{3}) + \log_2 n$

1.9. Докажите следующие утверждения *по индукции*. В качестве базы во всех пунктах можно считать, что $T(1) = 1$.

(a) Если $T(n) = 2T(\sqrt{n}) + 1$, то $T(n) = \mathcal{O}(\log n)$

(b) Если $T(n) = \log(n) \cdot T\left(\frac{n}{\log n}\right) + n$, то $T(n) = \mathcal{O}(n \log n)$

(c) Если $T(n) = 2T(\frac{n}{2} + \log n) + n$, то $T(n) = \mathcal{O}(n \log n)$

1.10. Есть алгоритм `A`, работающий за $T_{\mathtt{A}}(n)$, и алгоритм `B`, работающий за $T_{\mathtt{B}}(n)$. Известно, что

- $T_{\mathtt{A}}(n) = 7T_{\mathtt{A}}(\frac{n}{2}) + n^2$ и
- $T_{\mathtt{B}}(n) = xT_{\mathtt{B}}(\frac{n}{4}) + n$.

При каких значениях x второй алгоритм асимптотически медленнее первого?

1.11. Найдите точную (Θ) асимптотическую оценку для рекуррент:

(a) $T(n) = 2T(\frac{n}{2}) + \sqrt{n}$

(b) $T(n) = 3T(\sqrt{n}) + \log n$

(c) $T(n) = 2T(\frac{n}{2}) + \frac{n}{\log n}$

Неделя 2. Стек, очередь, дек

Практика

2.1. Придумайте модификацию дека, позволяющую за $\mathcal{O}(1)$ времени отвечать на запрос «вернуть сумму всех элементов в деке».

2.2. Придумайте модификацию очереди, позволяющую за $\mathcal{O}(1)$ времени обрабатывать запрос «добавить элемент ровно в середину».

2.3. Докажите, что любая правильная скобочная последовательность – либо пустая, либо имеет вид «$(p)q$», где p и q – тоже правильные скобочные последовательности.

2.4. `[Codeforces #744, E0]` Дана перестановка размера n. Все ее элементы по очереди добавляются в дек, но можно выбрать, добавить каждый следующий элемент в начало или в конец дека. За $\mathcal{O}(n)$ проверить, можно ли таким образом отсортировать перестановку.

2.5. **Реализации на стеке**.

(a) Придумайте реализацию очереди, использующую два стека и $\mathcal{O}(1)$ дополнительной памяти. Покажите, что в вашей реализации среднее время работы одной операции – $\mathcal{O}(1)$.

(b) Придумайте функцию потенциала Φ, позволяющую получить константное амортизированное время работы в предыдущем пункте.

(c) Придумайте реализацию дека, использующую три стека и $\mathcal{O}(1)$ дополнительной памяти. Покажите, что в вашей реализации среднее время работы одной операции – $\mathcal{O}(1)$.

2.6. За время $\mathcal{O}(n)$ найдите в массиве $\mathtt{a}$ длины n отрезок $[l, r]$, на котором максимальна величина $(r - l + 1) \cdot \min\limits_{i \in [l,r]} \mathtt{a}_i$.

2.7. Дан стек, элементы которого образуют последовательность $\mathtt{s}_1$ длины n. Со стеком произвели сколько-то действий, и теперь его элементы образуют последовательность $\mathtt{s}_2$ длины m. За $\mathcal{O}(n+m)$ времени определите какое минимальное количество действий могло потребоваться, чтобы получить $\mathtt{s}_2$ из $\mathtt{s}_1$.

2.8. Вспомните рассмотренный на лекции алгоритм вычисления арифметического выражения. Уточните и дополните его так, чтобы

- ▷ не возникало проблем с вычислением `2 - 3 + 4` в неправильном порядке
- ▷ можно было его обобщить на большее количество **левоассоциативных** операций с разными приоритетами.

Подсказка: Можно почитать здесь

2.9. **Битовый счетчик**. Рассмотрим битовый счетчик, хранящий число в двоичной системе счисления в виде массива бит (младший бит имеет индекс 0, старший – $k - 1$). Для счетчика реализована операция `increment`, увеличивающая его значение на 1.

(a) Оцените худшее и лучшее время работы одной операции `increment`. Оцените среднее время работы операции, по-честному посчитав суммарное время работы.

(b) Придумайте функцию потенциала Φ, позволяющую получить совпадающее с предыдущим пунктом амортизированное время работы `increment`.

(c) Сохранится ли оценка на среднее время работы одной операции, если добавить операцию `decrement`, уменьшающую значение счетчика?

2.10. **Оптимизации динамического массива**.

(a) Покажите, что динамический массив, который

▷ расширяется в 2 раза, когда целиком заполнен,

▷ и сужается в 2 раза, когда заполнен меньше, чем на $\frac{1}{4}$ доступной памяти,

имеет среднее время работы одной операции $\mathcal{O}(1)$.

(b) Та же задача, но оценка среднего времени работы должна выполняться методом потенциалов. Потенциал Φ должен зависеть только от параметров структуры и не должен зависеть от истории операций.

(c) Придумайте реализацию динамического массива, в которой каждая операция работает за **истинное** время $\mathcal{O}(1)$.
Уточнение: в этом пункте считайте, что выделение памяти происходит за $\mathcal{O}(1)$ времени.

Префиксные суммы

2.11. Дан массив целых чисел `a` длины n. Сделав предподсчет за $\mathcal{O}(n)$, научитесь отвечать на запросы «найти сумму чисел на отрезке массива с l_i по r_i» за $\mathcal{O}(1)$ времени.

2.12. Дан массив целых чисел `a` длины n. Найдите в нем отрезок с максимальной суммой

(a) за время $\mathcal{O}(n)$

(b) за **один проход** по массиву с $\mathcal{O}(1)$ дополнительной памяти (изменять массив `a` нельзя)

2.13. Решите задачу 2.11 для k-мерного массива и прямоугольных областей вместо отрезков. Время ответа на запрос – $\mathcal{O}(2^k)$.

2.14. Дан массив целых чисел `a` длины n. Разрешается сделать предподсчет за $\mathcal{O}(n)$, после чего требуется за время $\mathcal{O}(1)$ отвечать на запросы «увеличить все числа на отрезке массива с l_i по r_i на x_i». После всех запросов надо вывести весь `a`.

Идейные задачи не по теме

2.15. Дан массив `a` длины n, каждый элемент которого – целое число от 0 до k. Известно, что каждое возможное значение от 0 до k, кроме одного, встречается в массиве 0 или t раз, а какое-то одно – от 1 до $t-1$ раза.

Найдите это «лишнее» значение за **один проход** по массиву, используя $\mathcal{O}(\log t \cdot \log k)$ бит памяти. Решите задачу для

(a) $t = 2$

(b) произвольного целого $t \geqslant 2$

2.16. Дан массив `a` длины n, каждый элемент которого – целое число от 0 до k. Найдите такой x, что число x встречается в массиве `a` строго больше $\frac{n}{2}$ раз, или скажите, что такого нет.

(a) Решите задачу за **два прохода** по массиву (за один, если гарантируется, что ответ существует), используя не более $\mathcal{O}(\log n + \log k)$ бит памяти.

(b) Докажите, что любое решение за **один проход** по массиву потребует хотя бы $\log_2 \left(C_k^{\frac{n}{2}} \right)$ бит памяти.

Неделя 3. Куча. Сортировка слиянием

Практика

Куча

3.1. В кучу положили в некотором порядке числа от 1 до n (каждое по одному разу).

(a) Какое минимальное число может находиться на нижнем слое кучи?

(b) На каком (если нумеровать слои сверху вниз) минимальном слое может оказаться число n?

3.2. Сколько (точное число) листьев (childfree-вершин) в куче на n элементах?

3.3. **Быстрый** `build_heap()`.

(a) Докажите, что алгоритм

```
for i = 0 .. n-1 {
    sift_down(i)
}
```

не всегда строит корректную кучу на массиве из n элементов.

(b) Покажите, что время работы алгоритма

```
for i = n-1 .. 0 {
    sift_down(i)
}
```

работает за время $\mathcal{O}(n)$, то есть докажите, что $\sum\limits_{k=0}^{\log_2 n} k \cdot \frac{n}{2^k} = \mathcal{O}(n)$.

3.4. **Анка-пулеметчица**. Дан «почти отсортированный» массив длины n: каждый элемент отстоит от своей правильной позиции не более, чем на k.

(a) Придумайте алгоритм сортировки такого массива за $\mathcal{O}(n \log k)$.

(b) Докажите нижнюю оценку на время работы такой сортировки $\Omega(n \log k)$.

3.5. Даны две кучи размеров n и m. Как за время $\mathcal{O}(n+m)$ «слить» эти две кучи в одну?

3.6. Реализуйте для кучи операцию `decrease_key`(i, x) – «уменьшить значение элемента на позиции i на x». Время работы операции $\mathcal{O}(\log n)$. Можно ли аналогично реализовать `increase_key`(i, x)?

3.7. Придумайте структуру данных, поддерживающую выполнение за $\mathcal{O}(\log n)$ следующих операций:

- ▷ добавить элемент;
- ▷ изъять минимальный элемент;
- ▷ изъять максимальный элемент.

3.8. Придумайте структуру данных, поддерживающую выполнение за $\mathcal{O}(\log n)$ следующих операций:

- ▷ добавить элемент;
- ▷ изъять медианный (средний в возрастающем порядке) элемент.

3.9. Дан массив длины n. Найдите в нем k минимальных элементов за время

(a) $\mathcal{O}(n + k \log n)$

(b) $\mathcal{O}(n + k \log k)$

3.10. Дана двоичная куча из $n - k$ элементов. Предложите способ добавить в эту кучу k новых элементов за время $\mathcal{O}(k + \log^2 n)$.

> **Def.** Назовем *d-ичной кучей* дерево, устроенное аналогичным бинарной куче образом, но в которой каждый элемент имеет d детей (кроме, возможно, одной вершины, которой не хватает детей с последнего слоя).

3.11. Какие номера имеют дети и родитель вершины i в d-ичной куче? За какое время (в зависимости от n и d) выполняются операции в такой куче?

3.12. Когда выгодно использовать d-ичную кучу? Пусть с ней выполняются a операций `extract_min` и b операций `decrease_key`. Какое d следует выбрать, чтобы суммарное время работы было оптимальным?

Сортировка слиянием

3.13. **Сортировка вставками**. Предложите алгоритм, который

(a) за время $\mathcal{O}(n \log n)$ находит, сколько `swap`'ов сделает сортировка вставками на данном массиве длины n.

(b) за время $\mathcal{O}(n)$ находит четность количества `swap`'ов, которые делает сортировка вставками на данной перестановке длины n.

3.14. Дан массив длины n. Найдите количество инверсий в этом массиве за время $\mathcal{O}(n \log n)$

3.15. Дано k отсортированных массивов

(a) каждый длины $\frac{n}{k}$

(b) суммарной длины n,

требуется придумать функцию `merge`, которая сливает эти массивы в один отсортированный массив за время $\mathcal{O}(n \log k)$.

3.16. Придумайте, как в алгоритме сортировки слиянием избавиться от рекурсии. Время работы при этом измениться не должно.

3.17. Докажите, что любая сортировка сравнениями, которая работает правильно хотя бы на $\frac{1}{n^{100}}$ всех перестановок, работает за $\Omega(n \log n)$.

3.18. **In-place merge sort**. Придумайте, как делать `in-place merge` двух отсортированных половин массива. Разрешается использовать

(a) $\mathcal{O}(\sqrt{n})$

(b) $\mathcal{O}(1)$

дополнительной памяти.

Неделя 4. Быстрая сортировка, k-я порядковая статистика

Практика

4.1. Упражнения с лекции:

- ▷ Докажите по индукции, что из $\widehat{T}(n) \leqslant \widehat{T}(\frac{n}{3}) + \widehat{T}(\frac{2n}{3}) + 3n$ следует, что математическое ожидание времени работы быстрой сортировки $\widehat{T}(n) = \mathcal{O}(n \log n)$.
- ▷ Докажите по индукции, что из $\widehat{T}(n) \leqslant \widehat{T}(\frac{2n}{3}) + 3n$ следует, что математическое ожидание времени работы k-й порядковой статистики $\widehat{T}(n) = \mathcal{O}(n)$.

4.2. **Медиана медиан**. Упражнение с лекции: докажите, что из $T(n) \leqslant T(\frac{n}{5}) + T(\frac{7n}{10}) + c \cdot n$ следует, что время работы алгоритма Блюма, Флойда, Пратта, Ривеста и Тарьяна $T(n) = \mathcal{O}(n)$. Оцените, насколько «хорошая» константа получается в этом алгоритме по сравнению с рандомизированным алгоритмом.

4.3. Докажите или опровергните, корректен ли приведенный ниже псевдокод `QuckSort` в общем случае? Придумайте как исправить, если некорректен.
Диалект псевдокода – `python3`

```python
def quickSort(arr):
    n = len(arr)
    if n <= 1:
        return arr
    pivot = arr[random.randint(0, n - 1)] # inclusively
    left = [x for x in arr if x < pivot]
    right = [x for x in arr if x >= pivot]
    return quickSort(left) + quickSort(right)
```

4.4. Докажите или опровергните, корректен ли приведенный ниже псевдокод `QuickSort` в общем случае? Придумайте как исправить, если некорректен.
Диалект псевдокода – `Kotlin`

```kotlin
fun quickSort(arr: Int[], l: Int, r: Int) {
    if (l >= r)
        return
    val pivot: Int = arr[random(l, r)] // inclusively
    var i: Int = l
    var j: Int = r
    while (i <= j) {
        while (arr[i] < pivot)
            i++
        while (pivot < arr[j])
            j--
        if (i <= j) {
            swap(arr[i], arr[j])
            i++
            j--
        }
    }
    quickSort(arr, l, j)
    quickSort(arr, i, r)
}
```

4.5. Приведите способ быстрого построения перестановки чисел от 1 до n, на которой быстрая сортировка работает за время $\Omega(n^2)$, если на каждом шаге в качестве `pivot` выбирается

(a) центральный элемент на отрезке

(b) случайный элемент на отрезке, но вам заранее известен `seed` генератора случайных чисел

(c) медиана (второе по величине) из первого, центрального и последнего элементов

4.6. Опишите, как изменится время работы `QuickSort`, если в качестве `pivot` брать

(a) среднее арифметическое элементов

(b) равномерно распределенное случайное число из отрезка $[\min(a), \max(a)]$

4.7. Как изменится рекуррентное соотношение в алгоритме поиска k-й порядковой статистики за линейное время, если разбивать на блоки не по 5 элементов, а по 3 элемента? А по 7 элементов?

(a) Какая асимптотика будет в каждом из этих случаев?

(b) Можно ли добится линейной асимптотики, разбивая на блоки размера меньше 5? (возможно пригодится статья).

4.8. Придумайте работающий за $\mathcal{O}(n)$ алгоритм `in-place partition`,

(a) разбивающий отрезок на две части ($\leqslant$ `pivot`, $>$ `pivot`) с помощью двух указателей, которые можно только увеличивать

(b) разбивающий отрезок на три части ($<$ `pivot`, $=$ `pivot` и $>$ `pivot`) с помощью трех указателей

4.9. У вас есть n болтов и n подходящих к ним гаек, все болты (и, соответственно, все гайки) разного диаметра. Единственная операция, которая у вас есть – взяв какой-то болт и какую-то гайку, сравнить их диаметры (если не пролезает – значит болт больше, если болтается – значит меньше). Сравнивать болты между собой и гайки между собой нельзя. Найдите для каждого болта подходящую гайку за $\mathcal{O}(n \log n)$.

4.10. Сколько дополнительной памяти используется базовым алгоритмом поиска k-й порядковой статистики (без медианы медиан)? А алгоритмом Блюма, Флойда, Пратта, Ривеста и Тарьяна (медиана медиан)? Соптимизируйте память в базовом алгоритме до $\mathcal{O}(\log n)$ бит.

4.11. **Небыстрая сортировка**. В отсортированном массиве длины n изменили k элементов. Отсортируйте полученный массив за время $\mathcal{O}(n + k \log k)$.

4.12. Дан массив длины n и m значений $p_1, p_2, \ldots, p_m$. Найдите все p_i-е порядковые статистики в первом массиве за время $\mathcal{O}(n \log m)$ (можно считать, что $m \leqslant n$).

4.13. На координатной оси заданы n точек с координатами x_i (точки не отсортированы по координате). У каждой точки есть свой вес w_i. За время $\mathcal{O}(n)$ найдите такое x^*, что $\sum\limits_{i=1}^{n} w_i \cdot |x^* - x_i|$ минимальна.

4.14. **Несортировка**. На координатной оси заданы n точек с координатами x_i (точки не отсортированы по координате). За время $\mathcal{O}(n)$ найдите

(a) две точки с максимальным расстоянием между ними

(b) две соседние точки с максимальным расстоянием между ними

4.15. Найдите k минимальных элементов массива (в любом порядке) за один проход по массиву и $\mathcal{O}(n)$ времени, используя $\mathcal{O}(k)$ дополнительной памяти.

Неделя 5. Цифровая сортировка. Продвинутые кучи

Практика

Def. Сортировка называется *стабильной* или *устойчивой*, если элементы, считающиеся «равными» при сравнении, не меняют свой относительный порядок.

5.1. Предложите способ, используя $\mathcal{O}(n)$ дополнительной памяти, сделать стабильной любую сортировку на сравнениях (при этом не изменив асимптотику времени ее работы).

5.2. Дан массив чисел от 1 до k. Имея $\mathcal{O}(n+k)$ времени на предподсчет, научитесь отвечать на запросы «сколько в массиве чисел от l до r?» за время $\mathcal{O}(1)$.

5.3. Дан массив целых чисел длины n, все числа лежат в отрезке $[0, n-1]$. Отсортируйте циклические сдвиги этого массива в лексикографическом порядке, используя $\mathcal{O}(n)$ дополнительной памяти, за время

(a) $\mathcal{O}(n^2)$

(b) $\mathcal{O}(n \log n)$

5.4. **MSD done right**. Рассмотрим следующую реализацию MSD Radix Sort:

```
def sort(a: list[bigint], pos: int = 0):
    if pos == d or a.empty():
        return

    buckets = [[] for _ in 0..b-1]
    for item in a:
        c = item.get_digit(pos)
        buckets[c].add(item)

    a.clear()
    for c = 0..b-1:
        sort(buckets[c], pos + 1)
        a.add(all of buckets[c])
```

Здесь b – основание системы счисления, d – количество разрядов.

(a) Покажите, что такая реализация может работать за $\Omega(nbd)$.
Подсказка: подсчет работает не чисто за $\mathcal{O}$(`количества элементов`).

(b) Покажите, что если

▷ добавить к рекурсивному вызову условие `if (buckets[c].size() > 1)`;

▷ пропускать этап разбиения, если все текущие цифры чисел в `a` одинаковы,

то сортировка будет работать за $\mathcal{O}(n(b+d))$. Нужна ли вторая оптимизация?

(c) Пусть мы сортируем n **равномерно распределенных** на отрезке $[0, m]$ целых чисел. Какое b стоит выбрать (исходя из того, что $d \approx \log_b m$), чтобы время работы сортировки было оптимально?
Считайте, что перевод между системами счисления не потребуется.

Def. В следующей задачей под *хэш-таблицей*, если вы не знаете, что это, можете понимать структуру-данных, поддерживающую операции

▷ `set(key, value)` – присвоить по ключу `key` значение `value` за время $\mathcal{O}(1)$;

▷ `get(key)` – вернуть значение, ассоциированное с ключом `key`, за время $\mathcal{O}(1)$.

5.5. **Сортировка Киркпатрика**. Положим, что числа в этой задаче укладываются в машинное слово. Рассмотрим следующий алгоритм $\texttt{sort}(a, k)$ сортировки n целых чисел от 0 до $2^k - 1$:

- ▷ обозначим за $\texttt{high}(x)$ старшие $\frac{k}{2}$ бит числа x, а за $\texttt{low}(x)$ – младшие;
- ▷ разобьем все числа в массиве на корзины по старшей половине: для каждого a_i сделаем $\texttt{buckets}[\texttt{high}(a_i)].\texttt{add}(\texttt{low}(a_i))$;
- ▷ выпишем все использованные старшие части в массив `highs` и рекурсивно отсортируем с помощью $\texttt{sort}(\texttt{highs}, \frac{k}{2})$;
- ▷ отсортируем каждую корзину с помощью $\texttt{sort}(\texttt{buckets}[h], \frac{k}{2})$;
- ▷ имея информацию о порядке старших частей и младших частей в каждом бакете, восстановим целиком порядок исходных чисел.

Сейчас этот алгоритм работает дольше, чем мы бы хотели, его рекуррента выглядит как

$$T(n,k) = T\left(|\texttt{highs}|, \frac{k}{2}\right) + \sum_{h \in \texttt{highs}} T\left(|\texttt{buckets}[h]|, \frac{k}{2}\right) + n.$$

(a) Внесите модификацию в алгоритм, чтобы сумма аргументов на месте $*$ в $T(*, \frac{k}{2})$ в правой части не превосходила n.

(b) Покажите, что с такой модификацией время работы сортировки – $\mathcal{O}(n \log k)$, то есть $\mathcal{O}(n \log \log \max(a))$.

Скучные обычные сортировки

5.6. Дан массив натуральных чисел длины n. Можно уменьшать элементы массива, пока они остаются натуральными. За время $\mathcal{O}(n \log n)$ определите, какое максимальное количество различных элементов можно получить?

5.7. Даны n точек на прямой с координатами x_i. Выберите такую точку x^*, чтобы минимизировать величину D. Решите для D, равной $\sum\limits_{i=1}^{n}(x_i - x^*)^2$, $\sum\limits_{i=1}^{n}|x_i - x^*|$, $\max\limits_{i=1}^{n}(x_i - x^*)^2$ и $\max\limits_{i=1}^{n}|x_i - x^*|$. В каждом случае решите не более, чем за $\mathcal{O}(n)$.

5.8. Даны n точек на плоскости в координатах (x_i, y_i). Выберите такую точку (x^*, y^*) на плоскости, чтобы величина

(a) $\max\limits_{i=1}^{n}(\max(|x_i - x^*|, |y_i - y^*|))$

(b) $\max\limits_{i=1}^{n}(|x_i - x^*| + |y_i - y^*|)$

(c) $\sum\limits_{i=1}^{n}(\max(|x_i - x^*|, |y_i - y^*|))$

(d) $\sum\limits_{i=1}^{n}(|x_i - x^*| + |y_i - y^*|)$

(e) $\sum\limits_{i=1}^{n}(\min(|x_i - x^*|, |y_i - y^*|))$

была минимальна. Время работы $\mathcal{O}(n \log n)$ (в принципе можно и за $\mathcal{O}(n)$).

5.9. Вам дан массив из n элементов от 0 до $m - 1$.

(a) Известны p_i – частоты вхождения всех значений от 0 до $m-1$ в массив. Докажите нижнюю оценку $n \cdot (\mathcal{H}(p) - 1) \cdot \log e$ на число сравнений в худшем случае при сортировке сравнениями. Здесь $\mathcal{H}(p)$ – *энтропия*, то есть $\sum\limits_{i=0}^{m-1} p_i \ln \frac{1}{p_i}$.

Полезный факт: $\ln n! = n \ln n - n + \mathcal{O}(\ln n)$.

(b) Докажите оценку $n \log \frac{m}{e}$ в пределе $m \ll n$.

Кучи

5.10. Покажите, как сделать `build_heap()` для скошенной влево кучи за время $\mathcal{O}(n)$.

5.11. По данному вам n приведите последовательность действий с изначально пустой скошенной влево кучей, которая превращает ее в бамбук высоты n.

5.12. **Доказательство QuickHeap**. Поверим в доказанные в статье леммы с конечным итогом, что

$$\mathbb{E}\left[\sum(p_i - \texttt{head})\right] = \mathcal{O}(\texttt{size}) \text{ и}$$

$$\mathbb{E}\left[|p|\right] = \mathcal{O}(\log \texttt{size}),$$

где p – стек позиций разделяющих элементов, а `head` – позиция первого элемента.

Используя потенциал $\Phi = \sum(p_i - \texttt{head})$ и $T^* = T - \Delta\Phi$, докажите, что

(a) амортизированное время операции `insert` – это $\mathcal{O}(1)$

(b) амортизированное время операции `extract_min()` – $\mathcal{O}(\log \texttt{size})$

Стоит уделить внимание тому, что здесь T^*_{average} получается меньше T_{average}.

5.13. **Биномиальные кучи**. Пусть $\texttt{B}_n$ (биномиальное дерево порядка n) определено следующим образом:

▷ при $n = 0$ это дерево из одной вершины.

▷ при $n > 0$ это $\texttt{B}_{n-1}$, к которому первым ребенком подвешено еще одно $\texttt{B}_{n-1}$.

(a) Докажите, что $\texttt{B}_n$ имеет высоту n и содержит в точности 2^n вершин. Определим *биномиальную кучу* как набор биномиальных деревьев со свойством кучи, в котором нет двух деревьев одного порядка.

(b) Покажите, что для любого n существует биномиальная куча с n вершинами. Пусть на всех деревьях биномиальной кучи выполняется свойство кучи на минимум. Придумайте `get_min()` за $\mathcal{O}(\log n)$.

(c) Придумайте `merge` за $\mathcal{O}(\log n)$.

(d) Придумайте `insert` за $\mathcal{O}(\log n)$, `extract_min()` и `decrease_key()` по ссылке на узел за $\mathcal{O}(\log n)$.

(e) Придумайте `delete` по ссылке на узел за $\mathcal{O}(\log n)$.

Неделя 6. Двоичный поиск и его друзья

Практика

6.1. Дан массив чисел длины n, отсортированный по неубыванию. За $\mathcal{O}(\log n)$ найдите количество его элементов, равных заданному x.

6.2. Дан массив различных неотрицательных целых чисел длины n, отсортированный по неубыванию. За $\mathcal{O}(\log n)$ найдите минимальное неотрицательное целое число, которого в этом массиве нет.

6.3. Имея $\mathcal{O}(n \log n)$ времени на предподсчет, научитесь находить количество чисел, равных x на отрезке массива $[l, r]$, за время $\mathcal{O}(\log n)$.

6.4. Целочисленный двоичный поиск запускается от границ $l_0 = -1$, $r_0 = n$. За $\mathcal{O}(\log n)$ определите, могут ли в его процессе получиться границы (l, r).

6.5. `[В этой задаче считайте, что любые битовые и арифметические операции работают за` $\mathcal{O}(1)$`]`. Целочисленный двоичный поиск запускается от границ $l_0 = 0$, $r_0 = 2^k$. Определите за $\mathcal{O}(1)$ времени,

(a) могут ли в его процессе получиться границы (l_1, r_1)

(b) сколько различных пар значений (l, r) может быть встречено при таком бинпоиске

Def. *Циклическим сдвигом* массива $\mathtt{a}$ на величину d называется массив, получаемый из исходного переносом последних d элементов в начало:

$$\mathtt{a}^{\to d} = [\mathtt{a}_{n-d}, \mathtt{a}_{n-d+1}, \ldots, \mathtt{a}_{n-1}, \mathtt{a}_0, \mathtt{a}_1, \ldots, \mathtt{a}_{n-d-1}]$$

6.6. Отсортированный по неубыванию массив длины n циклически сдвинули на некоторую величину. За $\mathcal{O}(\log n)$ определите (или покажите, что это не всегда возможно), есть ли в нем элемент, равный x, если

▷ известно, что все элементы в массиве различны

▷ элементы могли повторяться

6.7. К отсортированному по неубыванию массиву приписали справа

(a) еще один отсортированный по неубыванию массив

(b) отсортированный по невозрастанию массив

и получили массив длины n. За $\mathcal{O}(\log n)$ определите (или покажите, что это не всегда возможно), есть ли в нем элемент, равный x, если

▷ известно, что все элементы в массивах различны

▷ элементы могли повторяться

6.8. **И снова точки...** В продолжение задачи 5.8. Выберите (за время $\mathcal{O}(n \log^2 \text{ans})$ в первых двух пунктах, $\mathcal{O}(n \log n + n \log \text{ans})$ в третьем) точку, минимизирующую величину

(a) $\max\limits_{i=1}^{n} \left((x_i - x^*)^2 + (y_i - y^*)^2\right)$

(b) $\sum\limits_{i=1}^{n} \sqrt{(x_i - x^*)^2 + (y_i - y^*)^2}$

(c) $\max\limits_{i=1}^{n} w_i(|x_i - x^*| + |y_i - y^*|)$, если у каждой точки есть вес $w_i \geqslant 0$

6.9. Дан массив чисел длины n, все числа не превосходят C. Найдите в нем отрезок длины не меньше k с максимальным средним арифметическим за время $\mathcal{O}(n \log C)$.

6.10. В игре есть n типов ресурсов, для постройки одного юнита требуется a_i единиц ресурса i для всех i. У игрока есть b_i единиц ресурса i для каждого i, и еще c единиц золота. Одну единицу золота можно обменять на d_i единиц ресурса i для произвольного i. Сколько юнитов может построить игрок?

6.11. В выборах участвуют n кандидатов. По последним опросам, за кандидата i готовы проголосовать a_i избирателей. Вы хотите, чтобы ваш кандидат победил (набрал больше голосов, чем любой другой кандидат). За 1 рубль можно изменить мнение одного избирателя. Сколько надо потратить денег на такую избирательную компанию?

6.12. **Коровы и стойла**. На

(a) прямой

(b) окружности

расположены m стойл с координатами x_i. Требуется выбрать такое максимальное k, чтобы можно было расставить n коров в стойла (не более одной коровы в стойле), и любые две соседние коровы находились на расстоянии $\geqslant k$. Время $\mathcal{O}(m \log \max(x_i) + \texttt{sort}(m))$.

6.13. Придумайте модификацию двоичной кучи, в которой `siftUp` делает только $\mathcal{O}(\log \log n)$ *сравнений* элементов. Количество обменов элементов, разумеется, уменьшить не получится.

6.14. Даны два отсортированных массива длины n, которые нельзя модифицировать. За

(a) $\mathcal{O}(\log^2 n)$ времени

(b) $\mathcal{O}(\log n)$ времени

и $\mathcal{O}(1)$ дополнительной памяти найдите k-ю порядковую статистику в объединении этих массивов.

6.15. Соптимизируйте время работы целочисленного бинпоиска в массиве с $\mathcal{O}(\log n)$ до $\mathcal{O}(\log k)$, где k – индекс ответа на запрос. Иными словами, если вас m раз попросят найти в массиве один из первых элементов, ваше решение должно отработать за $\mathcal{O}(m)$.

6.16. **Не бинпоиск**. Найдите второй максимум в массиве длины n за $n + \mathcal{O}(\log n)$ сравнений.

6.17. **K-Best**. Есть n предметов, у каждого есть стоимость $c_i \leqslant C$ и вес w_i. За время $\mathcal{O}(n \log C)$ выберите ровно k предметов с максимальной удельной стоимостью, то есть с $\max \frac{\sum c_i}{\sum w_i}$.

6.18. Дан многочлен степени n. В предположении, что мы умеем делать достаточно точные вычисления с нецелыми числами, найдите все его корни на отрезке $[-M, M]$ с точностью не менее ε за время $\mathcal{O}(n^3 \log \frac{M}{\varepsilon})$.

6.19. **Нижняя оценка**. Докажите, что если мы делаем и `lower_bound`, и `upper_bound` одновременно, нельзя найти корректный ответ в общем случае, используя меньше $2 \log_2 n - \mathcal{O}(1)$ сравнений.

Подсказка: посмотрите на количество возможных исходов, которые нам надо отличать.

6.20. Покажите, что если есть k массивов $\texttt{a}^i$ суммарной длины n, и мы строим $\texttt{b}^i$ как $\texttt{merge}(\texttt{a}^i, \texttt{b}^{i+1}[::2])$ (где $[::2]$ означает выбор каждого второго элемента), то сумма длин массивов $\texttt{b}^i$ – это $\mathcal{O}(n)$.

Неделя 7. Связные списки. Персистентность

Практика

7.1. Опишите реализацию операции `swap(a, b)`, меняющей в двусвязном списке местами элементы, на которые ссылаются указатели `a` и `b`.

7.2. Разверните односвязный список за $\mathcal{O}(n)$ времени и $\mathcal{O}(1)$ дополнительной памяти.

7.3. Даны n узлов, каждый из которых имеет ссылку на какой-то другой (возможно, `null`). За $\mathcal{O}(n)$ времени и $\mathcal{O}(1)$ дополнительной памяти проверьте, что эти узлы образуют

(a) линейный

(b) кольцевой

список длины n (менять ссылки в узлах нельзя).

7.4. Даны n узлов, каждый из которых имеет две ссылки на другие узлы (возможно, `null`). За $\mathcal{O}(n)$ времени и $\mathcal{O}(1)$ дополнительной памяти определите, образуют ли эти узлы несколько

(a) линейных

(b) кольцевых

двусвязных списков. Менять ссылки в исходных n узлах при этом нельзя.

7.5. Дан связный список из n элементов, каждый элемент которого имеет указатель на следующий и еще один вспомогательный указатель `data` на какой-то из узлов списка. Требуется построить копию этого списка такую, что если в исходном списке был указатель из `a` в `b`, то в новом есть такой же указатель из $\texttt{a}_{\text{copy}}$ в $\texttt{b}_{\text{copy}}$. Время работы $\mathcal{O}(n)$, дополнительная память $\mathcal{O}(1)$.

7.6. Представим, что мы храним двусвязный список в модели RAM-машины. Покажите, как можно вместо двух указателей `prev` и `next` в каждой вершине хранить только одно число того же размера.

7.7. Опишите реализацию функции `merge()` для слияния двух отсортированных двусвязных списков, работающую за $\mathcal{O}(n)$ времени с $\mathcal{O}(1)$ дополнительной памяти.

7.8. Отсортируйте связный список за $\mathcal{O}(n \log n)$ времени и

(a) $\mathcal{O}(\log n)$ дополнительной памяти

(b) $\mathcal{O}(1)$ дополнительной памяти

7.9. Придумайте модификацию множества двусвязных списков, позволяющую выполнять все следующие операции за $\mathcal{O}(1)$ времени:

- ▷ `create(data)` – создать новый список из одного узла, хранящего данные `data`
- ▷ `link(a, b)` – «склеить» два списка `a` и `b` (присоединить `b.head` к `a.tail`)
- ▷ `reverse(u, v)` – «развернуть» отрезок списка между узлами, на которые указывают `u` и `v` (гарантируется, что эти узлы лежат в одном списке)

Гарантируется, что после выполнения всех этих операций образуется один список размера n. Требуется вывести его элементы в актуальном порядке за $\mathcal{O}(n)$.

7.10. Есть таблица размера $n \times n$, с которой производят m действий вида «вырезать прямоугольник, развернуть на 180° и вклеить на место». Выведите состояние таблицы после всех действий, время на обработку – $\mathcal{O}(nm)$.

7.11. Есть полоска из n клеток одинакового размера. Совершается m действий вида «отсчитать от левого края t_i клеток вправо, согнуть полоску в этом месте, и положить правую часть на левую». Выведите состояние полоски после всех действий, время на обработку – $\mathcal{O}(n+m)$.

7.12. Дан массив длины $n+1$ из чисел от 1 до n. Требуется найти любое число, которое встречается в массиве хотя бы дважды, за $\mathcal{O}(n)$ времени и $\mathcal{O}(1)$ дополнительной памяти, если исходный массив доступен только на чтение.

7.13. Счетчик – целое число, изначально равное нулю, с которым можно производить операции инкремента `cnt += 1` и декремента `cnt -= 1`. Предложите реализацию

(a) полностью персистентного счетчика с $\mathcal{O}(1)$ времени на операцию и $\mathcal{O}(1)$ дополнительной памяти на версию

(b) частично персистентного счетчика с $\mathcal{O}(n)$ времени на операцию `get()`, $\mathcal{O}(1)$ времени на инкремент и декремент, и *одним битом* дополнительной памяти на версию

7.14. Предложите реализацию

(a) частично персистентной

(b) полностью персистентной

двоичной кучи на узлах и указателях, требующей $\mathcal{O}(\log n)$ времени на операцию и $\mathcal{O}(\log n)$ дополнительной памяти на каждую версию.

Неделя 8. Фибоначчиева куча. Quake-куча

Практика

8.1. Пусть подряд выполняется n операций `insert` в пустую биномиальную кучу. Какое будет среднее время работы одной операции?

8.2. Пусть мы создали n биномиальных куч размера 1, а затем выполняли с ними только операциии `merge`. Какое будет среднее время работы одной операции?

8.3. Придумайте потенциал Φ, позволяющий доказать, что

- ▷ амортизированное время работы операции `insert` в биномиальной куче равно $\mathcal{O}(1)$;
- ▷ амортизированное время работы остальных операций – $\mathcal{O}(\log n)$, где n – размер кучи.

8.4. Модифицируйте биномиальную кучу так, чтобы **истинное** время работы операции `insert` составляло $\mathcal{O}(1)$, а амортизированная стоимость остальных операций не поменялась.

8.5. **Тонкие кучи**. Пусть B_n определяется так же, как и в биномиальных кучах. Тогда *тонким деревом* ранга n назовем B_n, в которых у некоторых вершин могут быть удалены дети максимального ранга. *Тонкой кучей* тогда назовем коллекцию тонких деревьев со свойством кучи.

(a) Покажите, в чем отличие в определениях тонкой кучи и фибоначчиевой кучи.

(b) Придумайте реализации `merge` и `extract_min` для тонкой кучи. Амортизированное время `merge` должно быть равно $\mathcal{O}(1)$, `extract_min` – $\mathcal{O}(\log n)$.

(c) Придумайте реализацию `decrease_key` для тонких куч с амортизированным временем работы $\mathcal{O}(1)$.
Подсказка: используйте потенциал, похожий на потенциал в фибоначчиевых кучах.

(d) Покажите, что истинное время работы `decrease_key` – это $\mathcal{O}(\log n)$.

(e) Покажите, что можно добиться истинного времени работы `extract_min` $\mathcal{O}(\log n)$, если делать `consolidate()` каждый раз, когда в корневом списке становится хотя бы $2\log n$ элементов.

8.6. Для данного n предложите последовательность операций над изначально пустой фибоначчиевой кучей, образующей кучу из одного дерева, являющегося

(a) бамбуком высоты n

(b) биномиальным деревом ранга n

8.7. Разрешим вершинам фибоначчиевых деревьев иметь от нуля до d '`*`' в ранге: вершина ранга k с x звездочками будет соответствовать вершине ранга k с x удаленными детьми. При каких d время оценка на время работы операций в фибоначчиевой куче изменится, а при каких – останется такой же?

8.8. **Слияние Quake куч**.

(a) Покажите, как делать слияние двух Quake куч за время $\mathcal{O}(1)$, где n – их суммарный размер.

(b) Покажите, что при слиянии корректных Quake куч не понадобится вызывать `quake()`.

8.9. **Quake III** `[задача не имеет отношения к теме лекции]`. Прочитайте про алгоритм `Fast Inverse Square Root` и расскажите его.

Пополняемые структуры данных

> В этой секции данную вам структуру данных стоит воспринимать как «черный ящик». Иными словами, нельзя опираться на предположения о том, как она устроена внутри.
>
> В каждом задании будет указан интерфейс структуры, которая вам дана, и интерфейс структуры, которую необходимо на ее основе получить. Считайте, что помимо этих операций у структуры может быть произвольный `get`-интерфейс, который требуется сохранить.

8.10. **Ленивое удаление**.

- ☐ `insert` и `extract_min` за $T(n)$
- → `insert`, `extract_min` и `delete` по ключу за $T(n)$ **в среднем**

8.11. **Small-to-Large**.

- ☐ `insert` за $T(n)$
- → `insert` за $T(n)$, `merge` за $nT(n)\log n$ **в сумме**

8.12. **Rebuild**. В этой задаче может быть полезно посмотреть на пример отсортированного массива.

- ☐ `build` за $T(n)$
- → `build` за $T(n)$, `insert` за $\frac{T(n)}{\sqrt{n}}$ **в среднем**

Замечание: стоит отметить, что это работает для выпуклых вниз $T(n)$, а также что в частных случаях обычно можно эту оценку немного улучшить.

8.13. **Rebuild-2**. В этой задаче может быть полезно воспользоваться идеей битового счетчика.

- ☐ `build` за $T(n)$
- → `build` за $T(n)$, `insert` за $T(n\log_2 n)$ **в сумме** (но все `get` потенциально в $\log n$ раз дольше, чем были)

8.14. **Оптимизации кучи**.

(a) Дана куча, поддерживающая `insert`, `merge`, `extract_min` за $\mathcal{O}(\log n)$ и `build` за $\mathcal{O}(n)$. Постройте на ее основе кучу, поддерживающую все то же самое, но `insert` за $\mathcal{O}(1)$.

(b) Дана куча, поддерживающая `insert` за $\mathcal{O}(1)$, `merge`, `extract_min` за $\mathcal{O}(\log n)$ и `build` за $\mathcal{O}(n)$. Постройте на ее основе кучу, поддерживающую все то же самое, но `merge` за $\mathcal{O}(1)$.

Неделя 9. Динамическое программирование

Практика

Кузнечик

9.1. Кузнечик умеет прыгать со ступеньки с номером i на ступеньки с номерами $i+1$ и $2i$. Найдите количество способов добраться с нулевой ступеньки до n-й за $\mathcal{O}(n)$ времени.

9.2. Кузнечик умеет прыгать на $1, \ldots, k$ ступенек вперед. Найдите количество способов добраться с нулевой ступеньки до n-й за $\mathcal{O}(n)$ времени.

9.3. Кузнечик умеет прыгать на две или три ступеньки вперед, либо на одну *назад*. Но при этом прыжок назад можно совершать не более одного раза подряд. Найдите количество способов добраться с нулевой ступеньки до n-й за $\mathcal{O}(n)$ времени.

9.4. Кузнечик умеет прыгать на $1, \ldots, k$ ступенек вперед, у каждой ступеньки есть стоимость попадания на нее. Найдите способ добраться с нулевой ступеньки до n-й, потратив как можно меньше денег,

(a) за $\mathcal{O}(n \log k)$ времени

(b) за $\mathcal{O}(n)$ времени

9.5. Кузнечик умеет прыгать на одну или две ступеньки вперед, у каждой ступеньки есть стоимость попадания на нее. Найдите *количество* способов добраться с нулевой ступеньки до n-й, потратив как можно меньше денег, за $\mathcal{O}(n)$ времени.

9.6. Дана оптимизационная задача о кузнечике (надо минимизировать некоторый критерий). Предложите способ за то же время решить ту же задачу, но с условием, что если есть несколько одинаково оптимальных путей, требуется выбрать тот, в котором совершается наименьшее число прыжков.

Черепахматы

> Во всех заданиях этого блока черепашка перемещается по полю размера $n \times m$ и имеет цель добраться из клетки $(1, 1)$ (слева сверху) в клетку (n, m) (справа снизу), такое перемещение дальше будет просто называться *путем*.

9.7. Черепашка умеет перемещаться ровно на одну клетку вправо или вниз. Каждая клетка поля – либо черная, либо белая (не обязательно в шахматной раскраске). Найдите количество путей, проходящих по нечетному числу черных клеток, за $\mathcal{O}(nm)$ времени.

9.8. Черепаходья умеет прыгать на любое количество клеток вправо или вниз. Попадание в каждую клетку имеет свою стоимость. Найдите путь, проход по которому требует наименьшее число денег,

(a) за $\mathcal{O}(nm(n+m))$

(b) за $\mathcal{O}(nm)$

9.9. Черепахороль умеет перемещаться за один ход на один вправо и/или на один вниз (то есть в любую из трех клеток справа-снизу). Про некоторые клетки поля известно, что король не может в них вставать. Найдите количество путей за $\mathcal{O}(nm)$ времени.

9.10. Черепахонь умеет перемещаться ходом коня на $(+2, -1)$, $(+2, +1)$, $(+1, +2)$ или $(-1, +2)$. Найдите количество путей за $\mathcal{O}(nm)$ времени. В решении запрещается использовать условный оператор (`if`) и больше двух циклов (`for`).

9.11. Черепешка умеет перемещаться как обычная черепашка, но, достигая нижнего или правого края доски, получает возможность перемещаться на любое расстояние вправо или вниз. Попадание в каждую клетку имеет свою стоимость. Найдите путь, проход по которому требует наименьшее число денег, за $\mathcal{O}(nm)$ времени.

Наибольшие последовательности

9.12. **Наибольшая последовательнократная подпоследовательность**. В массиве размера n за $\mathcal{O}(n^2)$ времени найдите наибольшую подпоследовательность, каждый элемент которой делится на предыдущий.

Подсказка: используйте динамику, похожую на ту, которой решалась НВП.

9.13. В массиве размера n за $\mathcal{O}(n^2)$ времени найдите наибольшую подпоследовательность, каждые два соседних элемента которой отличаются не более, чем на d.

9.14. В массиве размера n за $\mathcal{O}(n^2)$ времени найдите наибольшую подпоследовательность, которая до некоторого момента монотонно возрастает, а затем – монотонно убывает.

9.15. **Наибольшая палиндромная подпоследовательность**. В массиве размера n за $\mathcal{O}(n^2)$ времени найдите наибольшую подпоследовательность, являющуюся палиндромом.

> **Def.** *Редакционное расстояние* или *расстояние Левенштейна* между двумя последовательностями – это минимальное количество действий «заменить элемент на другой», «вставить элемент» или «удалить элемент», которое необходимо совершить, чтобы получить вторую последовательность из первой.

9.16. **Задача о редакционном расстоянии**.

(a) Докажите, используя определение, что $\rho(a, b) = \rho(b, a)$, где ρ – редакционное расстояние

(b) Даны две последовательности длин n и m, найдите редакционное расстояние между ними за время $\mathcal{O}(nm)$

Подсказка: используйте динамику, похожую на ту, которой решалась НОП.

9.17. Даны две последовательности `a` длины n и b длины m. Известно, что все элементы последовательности `b` различны. Найдите НОП этих последовательностей за время $\mathcal{O}(n \log n)$.

Вечная классика

9.18. **Задача Иосифа Флавия**. Древняя игра: n человек стоят по кругу, каждый p-й по счету человек покидает круг и сбрасывется в яму с крокодилами. Даны n и p, найдите, кто останется в кругу последним за время $\mathcal{O}(n)$. Например, при $n = 6$ и $p = 2$ люди уходят в порядке $2, 4, 6, 3, 1$.

9.19. **Нетривиальные наблюдения**. Черепашка умеет перемещаться ровно на одну клетку вправо или вниз. Попадание в каждую клетку имеет свою стоимость (либо положительную, либо отрицательную). За $\mathcal{O}(nm)$ найдите минимальное число денег, которое должно быть у черепашки в начале пути, чтобы она могла добраться до правого-нижнего угла таблицы, не уходя в отрицательный баланс.

9.20. **Нетривиальные наблюдения-2**. Черепашка умеет перемещаться ровно на одну клетку вправо или вниз. Каждая клетка поля – либо черная, либо белая. Найдите любой путь, на котором количество черных и белых клеток одинаково, за $\mathcal{O}(nm)$ времени.

Неделя 10. Демоническое программирование

Практика

Рюкзаки

10.1. **Непрерывный рюкзак**. Решите задачу о рюкзаке за время $\mathcal{O}(n \log n)$, если можно взять произвольную долю от 0 до 1 каждого предмета.

10.2. **Ограниченный рюкзак**. Решите задачу о рюкзаке, если i-й из n предметов присутствует в b_i экземлярах (соответственно, может быть взят от 0 до b_i раз), за время

(a) $\mathcal{O}\left(W \cdot \sum_{i=1}^{n}(1 + \log b_i)\right)$

(b) $\mathcal{O}(nW)$

10.3. **Рюкзак с мультивыбором**. Решите задачу о рюкзаке за время $\mathcal{O}(nW)$, если предметы разбиты на группы, и из каждой группы нужно выбрать ровно один предмет,

(a) с использованием $\mathcal{O}(nW)$ памяти

(b) с использованием $\mathcal{O}(W)$ памяти

10.4. Несколько (t) друзей выиграли n призов на олимпиаде, у каждого приза есть стоимость c_i. Суммарная стоимость призов – C. Поделите призы между друзьями так, чтобы

(a) разница в суммарной стоимости была минимальна, за время $\mathcal{O}(nC)$ при $t = 2$

(b) каждый получил поровну, а разница в суммарной стоимости была минимальна, за время $\mathcal{O}(n^2C)$ при $t = 2$

(c) разница между максимальной и минимальной суммарной стоимостью призов была минимальна, за время $\mathcal{O}(nC^2)$ при $t = 3$

10.5. **Рюкзак на отрезке**. Даны n предметов с весами w_i и рюкзак вместимости W. Научитесь, используя $\mathcal{O}(nW)$ предподсчета, за $\mathcal{O}(1)$ отвечать на запрос «можно ли предметами с l-го по r-й» набрать суммарный вес $w \leqslant W$?».

Разное

10.6. Даны две последовательности a и b длины n. Найдите их наибольшую общую возрастающую подпоследовательность за время

(a) $\mathcal{O}(n^3)$

(b) $\mathcal{O}(n^2)$

10.7. Дана таблица $n \times m$, в каждой клетке написано число. Найдите путь минимального веса из $(1, 1)$ в (n, m), если разрешено двигаться вниз, влево и вправо. Решите задачу с восстановлением ответа за $\mathcal{O}(nm)$ времени и

(a) $\mathcal{O}(nm)$ памяти

(b) $\mathcal{O}(nm^{\frac{1}{2}})$ памяти

(c) $\mathcal{O}(nm^{\frac{1}{3}})$ памяти

10.8. Дан шаблон длины n, состоящий из букв латинского алфавита, а также символов '`?`' (вайлкард на любой символ) и '`*`' (вайлдкард на произвольную подстроку). Проверьте соответствие данной строки s длины m этому шаблону за время $\mathcal{O}(nm)$.

10.9. Даны два таких же шаблона длин n и m. Найдите минимальную по длине строку, соответствующую обоим шаблонам, за время $\mathcal{O}(nm)$.

10.10. На прямой даны n точек с координатами x_i. Требуется покрыть все эти точки зоной WiFi-сигнала, для этого можно устанавливать WiFi-точки. Установка точки с радиусом действия r стоит $a + br^2$. Определите минимальную стоимость покрытия всех точек сигналом за время $\mathcal{O}(n^2)$.

10.11. Два игрока играют в игру на массиве длины n. За ход можно забрать себе одно из крайних чисел. Определите максимальную сумму, которую может себе обеспечить каждый из игроков при оптимальной игре, за время $\mathcal{O}(n^2)$.

10.12. Даны n дубов, стоящих в ряд, у i-го дуба высота h_i. Дуб можно срубить только если ближайшие к нему еще не срубленные дубы либо оба выше, либо оба ниже (крайние рубить нельзя). Определите за время $\mathcal{O}(n^3)$, какое минимальное число дубов надо срубить, чтобы высоты оставшихся образовывали возрастающую последовательность.

10.13. **Взвешенный бинарный поиск**. Известно, что обращение к i-му элементу массива требует t_i времени. Сам массив длины n при этом неизвестен. Определите за время $\mathcal{O}(n^3)$, какие элементы стоит выбирать в качестве «центральных» в бинпоиске, чтобы минимизировать суммарное время работы.

10.14. **Быстрая покраска забора**. Требуется из массива из n нулей получить массив a с элементами от 0 до C за минимальное число действий вида «присвоить значение на отрезке». Определите минимальное число действий за время

(a) $\mathcal{O}(n^3C)$

(b) $\mathcal{O}(n^3)$

10.15. Дано подвешенное дерево на n вершинах, у каждого ребра есть вес. За время $\mathcal{O}(n)$ найдите максимальное по весу паросочетание в этом дереве.

10.16. Дано подвешенное дерево на n вершинах, у каждой вершины есть (возможно, отрицательный) вес. Требуется за время $\mathcal{O}(n)$ определить, какие ребра (вместе с их целыми поддеревьями) следует удалить, чтобы максимизировать вес оставшейся части.

> **Def.** *Центроидом* дерева размера n называется такая вершина, при удалении которой все оставшиеся компоненты, на которые дерево распадается, имеют размер $\leqslant \frac{n}{2}$.

10.17. **Центроид**. За время $\mathcal{O}(n)$ найдите для каждого поддерева данного дерева размера n его центроид.

10.18. Дано подвешенное дерево на n вершинах. Найдите количество его поддеревьев размера k, содержащих корень, за время

(a) $\mathcal{O}(n^3)$

(b) $\mathcal{O}(n^2)$ или $\mathcal{O}(nk)$

Поддеревом в данной задаче стоит считать любое связное множество вершин.

10.19. **Взорвать дерево!** Выберите во взвешенном дереве размера n минимальное по весу множество вершин, чтобы расстояние от любой вершины до ближайшей выбранной было не больше d, за время $\mathcal{O}(n^2)$.

Неделя 11. Больше динамики

Практика

Просто ДП

11.1. **Рюкзак с большими весами**. Решите задачу о рюкзаке с большими весами, исходя из ограничений $n \leqslant 1000$, $w_i \leq 10^9$, $c_i \leq 100$. Унести в рюкзаке $W \leq 10^9$ предметы максимальной суммарной стоимости.

11.2. **Пираты!**. Флот пиратов плывет по направлению к берегу. Для каждого пиратского судна известны x_i – его координата вдоль берега и t_i – время до достижения берега. На берегу есть одна пушка, которая всегда направлена перпендикулярно берегу и может перемещаться вдоль него со скоростью v. На выстрел время не тратится. Определите порядок уничтожения пиратов, при котором ни одно судно не достигнет берега, за $\mathcal{O}(n^2)$.

11.3. **Умные переливания**. Есть три стакана объемами a, b и c от 0 до n. Требуется переливаниями воды между ними получить объем ровно x за время $\mathcal{O}(n^2)$.

11.4. Найдите количество чисел ровно из k цифр из множества $a_1, \ldots, a_d$ (считайте d константой),

(a) кратных M, за время $\mathcal{O}(kM)$

(b) лежащих в отрезке $[l, r]$, за время $\mathcal{O}(k)$

11.5. Придумайте цикл, перебирающий маски

(a) всех подмножеств множества в возрастающем порядке

(b) всех надмножеств множества

11.6. Придумайте, как вычислять `popcount` от маски за $\mathcal{O}(1)$ (можете рассказать принцип работы стандартной реализации из C++).

11.7. Даны n гирь, про некоторые их пары известно, какая гиря в паре тяжелее. Посчитайте число способов упорядочить гири по массе, не противоречя известной информации, за время $\mathcal{O}(2^n n^2)$.

11.8. Даны множество A и m множеств B_i, все $\subset \{1, \ldots, n\}$. Найдите минимальный по размеру набор B_{i_j} такой, что $\bigcup B_{i_j} = A$, за время $\mathcal{O}(2^m)$.
Можно считать, что $n, m \leq 64$, но нельзя считать их за константы в асимптотике.

11.9. Дана строка длины n над алфавитом размера k. Требуется выбрать некоторое множество букв и заменить их все на пробелы так, чтобы среди любых m символов подряд был хотя бы один пробел. Найдите такое минимальное по размеру множество за время

(a) $\mathcal{O}(n + 2^k \cdot \mathrm{poly}(k))$

(b) $\mathcal{O}(n + 2^k)$

11.10. Найдите число замощений доски $n \times m$ полосками 1×3 и 3×1 за время $\mathcal{O}(nm3^n)$.

11.11. Найдите число способов расставить k коней на шахматной доске $n \times n$ так, чтобы они не били друг друга, за время

(a) $\mathcal{O}(4^n \cdot \mathrm{poly}(n))$

(b) $\mathcal{O}(3^n \cdot \mathrm{poly}(n))$

(c) $\mathcal{O}((3 - \varepsilon)^n)$

Оптимизации ДП

11.12. Есть последовательность из n человек, для каждой пары (i, j) известен уровень их несовместимости $u_{i,j} = u_{j,i}$. Разбейте последовательность на k отрезков, чтобы суммарный уровень несовместимости по всем парам, попавшим в один отрезок, был минимальным, за время $\mathcal{O}(nk \log n)$.

11.13. Дан массив длины n. Стоимость отрезка вычисляется как сумма $\texttt{last}(x) - \texttt{first}(x)$ по всем различным значениям x на отрезке (где `first` и `last` – позиции первого и последнего вхождения, соответственно). Разбейте массив на k отрезков минимальной суммарной стоимости за время $\mathcal{O}(nk \log n)$.

> **Def.** *Бинарным деревом поиска* назовем дерево с ключами в вершинах такое, что для любой вершины все ключи в левом поддереве меньше ее ключа, а в правом – больше.

11.14. Даны n различных ключей от 1 до n. Известно количество обращений к каждому ключу f_i. Расположите ключи в бинарное дерево поиска так, чтобы минимизировать $\sum\limits_{i=1}^{n} f_i \cdot \texttt{depth}(i)$, за время $\mathcal{O}(n^2)$.

> **Def.** Назовем схему кодирования, сопоставляющую каждому символу новую последовательность бит, *сохраняющей порядок*, если лексикографический порядок кодов совпадает с алфавитным порядком соответствующих символов.

11.15. Дан алфавит из n символов и частота вхождения каждого символа в текст f_i. Найдите оптимальный префиксный сохраняющий порядок код для этого текста за время $\mathcal{O}(n^2)$.

11.16. Решите задачу о разрезании бревна (разрез отрезка имеет стоимость, равную текущей длине, требуется сделать n разрезов в определенных местах), за время $\mathcal{O}(n^2)$.

11.17. Вас преследует робот. Изначально вы находитесь вместе с роботом в точке 0, и движетесь со скоростью 1, а робот – со скоростью 2. Если вы одновременно с роботом находитесь в целой точке x, вы можете применить специальное устройство (произвольное число раз), затратив q_x энергии, тем самым останавливая робота и откладывая его движение на t_x времени. За время $\mathcal{O}(n \log n)$ найдите минимальное количество энергии, необходимое, чтобы добраться до точки n строго раньше робота.

11.18. Вдоль улицы есть n мест для строительства остановок, i-е из них находится на расстоянии x_i от начала улицы и вызывает у жителей суммарное недовольство c_i при постройке на этом месте остановки. При этом j-й из m жителей имеет любимое число d_j, и для каждых двух соседних построенных остановок на расстоянии Δx его уровень удовлетворенности увеличивается на $|d - \Delta x|$. За время $\mathcal{O}((n + m) \log(n + m))$ найдите максимальный возможный уровень суммарной удовлетворенности жителей.

11.19. Дано дерево на n вершинах с корнем в вершине 1. В вершине i записана пара чисел (a_i, b_i). Прыжок из вершины u в вершину в ее поддереве v требует $a_u \cdot b_v$ сил. Допрыгайте из корня до листа, затратив как можно меньше сил, за время $\mathcal{O}(n \log n)$.

Неделя 12. Жадные алгоритмы

Практика

12.1. Постройте код Хаффмана за $\mathcal{O}(n)$, если частоты букв уже даны в отсортированном порядке.

12.2. В фирму поступило n заказов, которые можно выполнять в произвольном порядке. На выполнение заказа i необходимо время t_i. В каждый момент времени можно работать ровно над одним заказом. Пусть e_i – момент окончания выполнения заказа номер i. Распределите работу над заказами так, чтобы минимизировать $\sum_i e_i$, за время $\mathcal{O}(n \log n)$.

12.3. Даны n гномов. Если i-го гнома укладывать спать a_i минут, он потом спит b_i минут. Определите, можно ли сделать так, чтобы в какой-то момент все гномы спали, за время $\mathcal{O}(n \log n)$.

12.4. Машина тратит единицу топлива на километр, имеет бак объема k и находится в начале прямой дороги в точке 0. Для всех $i \in \mathbb{N} \cup \{0\}$ в i километрах от нее есть заправочная станция со своей положительной ценой c_i за литр топлива. Определите за время $\mathcal{O}(n)$, как проехать n километров за минимальную стоимость.

12.5. Даны n монеток, у каждой есть своя вероятность выпадения орла p_i. Нужно выбрать подмножество размера k из них, для которого вероятность выпадения ровно $\frac{k}{2}$ орлов при одновременном подбрасывании максимальна (k – четное), за время

(a) $\mathcal{O}(n \log n + k^3)$

(b) $\mathcal{O}(n + k^2)$

12.6. Маршрутка совершает рейс от первой до n-й остановки. В маршрутке m мест для пассажиров. Есть k человек, про каждого заранее известно, что он хочет доехать от остановки s_i до f_i. Проезд для пассажира стоит 1 вне зависимости от расстояния между остановками. Максимизируйте прибыль, при условии, что можно выбирать, кого сажать в маршрутку на каждой остановке, за время $\mathcal{O}((n + m + k) \log m)$.

> **Def.** *Независимым множеством* (или *антикликой*) в графе назовем подмножество вершин графа, никакие две из которых не связаны ребром.

12.7. Пусть в графе G есть n вершин и m ребер, а максимальная степень равна d. Найдите в нем независимое множество размера хотя бы

(a) $\frac{n}{d+1}$ за время $\mathcal{O}(n + m)$

(b) $\sum\limits_{v=0}^{n-1} \frac{1}{\deg(v)+1}$ за время $\mathcal{O}(n \log n + m)$

Считайте, что граф уже дан в памяти в виде массива, где для каждой вершины хранится список ее соседей.

12.8. Имеется n деталей и два станка. Каждая деталь должна сначала пройти обработку на первом станке, затем – на втором. При этом i-я деталь обрабатывается на первом станке за a_i времени, а на втором – за b_i времени. Каждый станок в каждый момент времени может работать только с одной деталью. Требуется за время $\mathcal{O}(n \log n)$ составить такой порядок подачи деталей на станки, чтобы итоговое время обработки всех деталей было бы минимальным.

12.9. Преподаватели сделали n заявок на занятие. Каждое занятие начинается в момент b_i и кончается в момент e_i (занимает интервал $[b_i, e_i)$). Два занятия в одной аудитории быть не могут. Распределите заявки по аудиториям так, чтобы общее число аудиторий было минимально, за $\mathcal{O}(n \log n)$.

12.10. **Оценки на Хаффмана**.

(a) Пусть в тексте встречается символ с частотой большей $\frac{2}{5}$. Докажите, что код Хаффмана будет содержать слово длины 1.

(b) Пусть в тексте все символы встречаются реже $\frac{1}{3}$. Докажите, что при кодировании Хаффмана любое кодовое слово будет иметь длину не меньше 2.

12.11. Ровно n атлетов хотят выстроить из своих тел башню максимальной высоты. Башня – это цепочка атлетов, первый стоит на земле, второй стоит у него на плечах, третий стоит на плечах у второго и т.д. Каждый атлет характеризуется силой s_i и массой m_i. Сила – это максимальная масса, которую атлет способен держать у себя на плечах. Известно, что если атлет тяжелее, то он и сильнее, но атлеты равной массы могут иметь различную силу. Определите максимальную высоту башни за время $\mathcal{O}(n \log n)$.

12.12. **Пятачок, у тебя есть дома ружье?**. Даны n непересекающихся кругов на плоскости. Мы стоим в точке $(0, 0)$ и можем стрелять по прямой. Определите, как минимальным числом выстрелов проткнуть все круги, за время $\mathcal{O}(n \log n)$.

12.13. Ровно n школьников упали в яму глубины S. Каждый школьник имеет рост (от ног до плеч) h_i и длину рук l_i. Школьники могут вставать друг другу на плечи, верхний школьник может вытянуть руки. Чтобы выбраться из ямы, необходимо дотянуться руками до уровня земли. За время $\mathcal{O}(n \log n)$ определите,

(a) могут ли выбраться все школьники

(b) какое максимальное число школьников может выбраться

Неделя 13. Хеш-таблицы и повторение

Письменное ДЗ

★ Правила сдачи

- ▷ Письменное ДЗ оформляется в LaTeX, написанные от руки решения приниматься не будут
- ▷ Решения надо прислать **до 17 января 2024, 23:59** на почту `algorithms.teaching+ct2023@gmail.com`
- ▷ Тема письма должна быть указана в виде «`Группа HW Фамилия Имя`» (без кавычек), например: «`M3140 HW Иванов Иван`». Обратите внимание, что буква «M» в названии группы – латинская
- ▷ К письму должны быть приложены как сгенерированный `.pdf`-файл, так и исходный `.tex`-файл

Хеш-таблицы

Def. Семество хеш-функций $\mathcal{H}$, действующих из U в $[m]$, называется *универсальным*, если для каждой пары различных ключей $x, y \in U$ количество хеш-функций $h \in \mathcal{H}$, для которых $h(x) = h(y)$, не превосходит $\frac{|\mathcal{H}|}{m}$.

Def. Семество хеш-функций $\mathcal{H}$, действующих из U в $[m]$, называется *попарно независимым*, если для каждой пары возможных значений $p, q \in [m]$ и каждой пары различных ключей $x, y \in U$ количество функций, для которых $h(x) = p$ и $h(y) = q$, не превосходит $\frac{1}{m^2}(|\mathcal{H}| + o(|\mathcal{H}|))$.

1. Объясните разницу между универсальностью и попарной независимостью. Верно ли, что одно свойство строго сильнее другого?
2. Докажите или опровергните, что семество хеш-функций
   (a) $\mathcal{H}_p = \{h_a(x) = (ax \bmod p) \bmod m\}$ для некоторого $p \in \mathbb{Z}$
   (b) $\mathcal{H}_p = \{h_a(x) = (ax \bmod p) \bmod m\}$ для некоторого $p \in \mathbb{P}$
   (c) $\mathcal{H}_p = \{h_{a,b}(x) = (ax + b \bmod p) \bmod m\}$ для некоторого $p \in \mathbb{P}$

   является
   - ▷ универсальным;
   - ▷ попарно независимым.
3. Есть множество вещественных чисел. Требуется выполнять две операции за $\mathcal{O}(1)$: «добавить число x» и «найти **любое** число в диапазоне $(x - \varepsilon, x + \varepsilon)$ (или сказать, что таких нет)». Значение ε отдинаковое для всех запросов.
4. Предложите хэш-функцию для объектов вида
   (a) пара чисел от 0 до $p - 1$ при $p \in \mathbb{P}$ и $p < 2^{32}$
   (b) битовый вектор фиксированной длины n
   (c) строка произвольной длины из символов от ‘a’ до ‘z’

   Хэш-функция должна возвращать 32-битное число и должна минимизировать вероятность коллизии в случае равновероятного выбора объекта.

Бонус (теория расписаний)

5. В фирму поступают заказы, которые можно выполнять в произвольном порядке. В каждый момент времени можно работать ровно над одним заказом. Изначально заказов нет, i-й заказ поступает в момент времени r_i, работать над ним нужно t_i времени.

   (a) Пусть e_i – момент окончания выполнения заказа номер i. Распределите работу над заказами так, чтобы минимизировать $\sum e_i$.

   (b) Все заказы объединены в проекты (один заказ относится к одному проекту, заказы из одного проекта могут поступать не подряд). Пусть e_i – момент окончания выполнения последнего (в порядке выполнения) из заказов в проекте с номером i. Нужно распределить работу над заказами так, чтобы минимизировать $\sum e_i$. Придумайте решение, которое не более чем в два раза хуже оптимального.

   Переходить от одного заказа к другому можно в любой момент времени (даже если заказ не доделан до конца, незаконченный заказ можно будет возобновить с того же места). Свойства заказа (r_i, t_i и его проект во втором пункте) не известны до момента его поступления. Время $\mathcal{O}(n \log n)$, при условии, что всего поступит n заказов.