АиСД у2023. Первый семестр

Домашние задания М3132-М3133

⟨Версия от 7 декабря 2023 г.⟩

Темы

Неделя 1. Введение. Асимптотические оценки алгоритмов.

Конспект от Гриши @kgrigoriy Хлытина

```java
/* Функция min_search поиска минимума в массиве arr */
int min_search(int[] arr) {
    /* Инициализируем переменную ans максимальным числом типа int */
    int ans = Integer.MAX_VALUE;
    /* Пройдем циклом for по всему массиву */
    for (int i = 0; i < arr.length; i++) {
        /* Если текущий элемент меньше текущего минимума */
        if (arr[i] < ans) {
            ans = arr[i];
        }
    }
    /* Вернем найденный минимум */
    return ans;
}
```

Listing 1: Поиск минимума в массиве

```java
/* Рекурсивная функция sum */
/* Подсчитывает сумму всех натуральных чисел от 0 до x */
/* BigInteger - тип данных для больших целых чисел (без ограничения размера) */
BigInteger sum(BigInteger x) {
    /* Если x <= 0, то вернем как ответ 0 */
    if (x <= 0) {
        /* Вернем BigInteger со значением равным нулю */
        return BigInteger("0");
    } else {
        /* Иначе рекурсивно вычислим сумму от 0 до (x - 1) */
        BigInteger ans = sum(x - 1);
        /* Вычислим ответ от 0 до x*/
        ans = ans + x;
        /* Вернем результат */
        return ans;
    }
}
```

Listing 2: Сумма всех натуральных чисел от 0 до x

Конспект лекции:

▷ Что такое алгоритм?

- Это **конечная** последовательность действий для абстрактного вычислителя;
- Алгоритм получает на вход данные *input*, и после завершения работы выдает данные *output*;

▷ Абстрактный вычислитель: Машина Тьюринга, RAM-модель. Какие отличия между RAM-моделью и реальным компьютером?

- В RAM-модели программы (набор инструкций) хранятся в отдельной памяти, а в реальном компьютере хранятся вместе с другими данными в оперативной памяти;
- В RAM-модели мы грубо считаем, что каждая инструкция выполняется за **одну** условную единицу времени, в то время как для реального компьютера это не так (см. пункт ниже);

▷ Реальный компьютер: язык ассемблера, кэш-пямять.

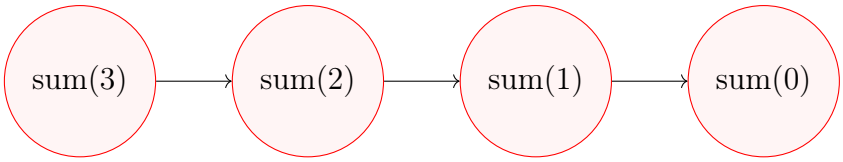
- Подробнее про язык ассемблера можно почитать здесь (не обязательно);
- Арифметические инструкции (например `ADD`) работают примерно за **один** такт процессора, в то время как инструкции работы с оперативной памятью (например `MOV`) работают за несколько сотен тактов;
- Исходя из пункта выше, для оптимизации процессор *кеширует* данные: То есть при обращении в оперативную память к ячейке с адресом i, процессор заодно выгружает из оперативной памяти несколько соседних с i-ой ячеек (это называется *кеш-линией*) и запоминает их в *кеш-память* (которая очень быстрая и находится прямо в процессоре).
  Теперь, если мы захотим обратится в оперативную память к ячейке с адресом $i-1$, процессор не пойдет в оперативную память, а возьмет данные из *кеш*-а (и это будет в несколько сотен раз быстрее, примерно за **один** такт процессора);

▷ Как оценить производительность алгоритма?

- Засекать время плохо — у разных процессоров разная производительность;
- Будем считать функцию $T(n)$ — количество абстрактных операций (арифметические операции и операции обращения к памяти) в RAM-модели, которые делает наш алгоритм, если на вход алгоритму подаются входные данные *input* и размер входных данных равен $n = |input|$ (в простейшем случае n это количество чисел подающихся на вход нашему алгоритму);
- Псевдокод алгоритма на языке `Java` для поиска минимума в массиве, который был рассмотрен на лекции, приведен на листинге 1;
- На лекции мы посчитали, что *время работы* такого алгоритма с листинга 1 примерно равно $T(n) = 2 + 3 \cdot n$ абстрактных операций (по одной абстрактной операции на строчках: `ans = Integer.MAX_VALUE`, `return ans`; а так же цикл `for` размера n, на каждой итерации которого выполняется около трех абстрактных операций);

▷ А как оценить производительность алгоритма, если он представляет из себя рекурсивную функцию?

- Построим **дерево рекурсивных вызовов**!
- Посмотрим на листинг 2, на котором изображена рекурсивная функция $sum(x)$, которая вычисляет сумму всех натуральных чисел от 0 до x;
- Как оценить её *время работы*? Как для неё будет выглядеть функция $T(x)$ количества абстрактных операций в зависимости от числа x? А как для неё будет выглядеть функция $T(n)$, где n — размер входа?
  (Обратите внимание: в данном случае, так как число x подающееся на вход алгоритму может быть очень большим по значению, размер входа будет равен $n = log(x)$, так как размер входа это количество цифр в записи числа x)
- Построим *дерево рекурсивных вызовов* например для вызова функции $sum(3)$, как делали это на лекции:



- Сколько абстрактных операций делает функция *sum* на одном слое в *дереве рекурсивных вызовов*?
  По её коду видно, что на одном слое она только складывает два числа: `ans = ans + x`. Сейчас для простоты **временно будем считать**, что **суммирование** двух больших чисел работает за **одну** абстрактную операцию (хотя на самом деле такое суммирование работает за количество цифр в записи числа — то есть за $\mathcal{O}(log\, x)$);
- На рисунке выше видно, что количество слоёв в нашем *дереве рекурсивных вызовов* равно x, причем мы сказали что на каждом слое функция *sum* делает только одну абстрактную операцию, таким образом время работы функции $sum(x)$ будет равно *количество слоев* $\times$ *количество операций на слое* и будет равно: $T(x) = x \cdot 1 = \Theta(x)$.
  Если расписать $T(x)$ рекурсивно, то будет: $T(x) = T(x-1)+1$ (так как мы сказали что на каждом слое функция *sum* делает **одну** операцию, а так же рекурсивно вызывает саму себя с аргументом ровно на один меньше);
- **Важно для понимания №1**:
  если бы наша исследуемая функция *sum* на каждом слое делала не 1 операцию, а x операций, тогда ее время работы было бы: $T(x) = T(x-1)+x$, и было бы верно, что $T(x) = \Theta(x^2)$, так как количество слоев все еще было бы равно x и на каждом слое функция делала бы порядка x операций: $T(x) = x + (x-1) + (x-2) + (x-3) + (x-4) + \ldots + 2 + 1 = \Theta(x^2)$;
- **Важно для понимания №2**:
  так как мы уже выше сказали, что суммирование двух больших чисел `ans = ans + x` на самом деле работает не за одну операцию, а за количество цифр в записи числа — то есть за $\mathcal{O}(log(x))$, получается что на одном слое функция *sum* на самом деле делает порядка $log(x)$ операций, и на самом деле время ее работы: $T(x) = T(x-1) + log(x)$. Количество слоев все еще равно x, на каждом слое функция *sum* делает порядка $log(x)$ операций, получаем: $T(x) = log(x) + log(x-1) + log(x-2) + \ldots + log(2) + log(1) = \Theta(x \cdot log(x))$;

Def. Так как нас интересует только *асимптотика* функции $T(n)$ (порядок роста функции $T(n)$ относительно размера n входных данных), мы сказали, что нам интересен только наибольший член этой функции, а так же неинтересны константы. Чтобы формально и коротко это записать, мы ввели следующие обозначения:

- $T(n) = \mathcal{O}(f(n))$
- $T(n) = \Omega(g(n))$
- $T(n) = \Theta(h(n))$

В примерах выше, функции $f(n)$, $g(n)$ — это соответственно *оценка сверху* и *оценка снизу* для нашей функции $T(n)$.

Def. Напомним используемые далее определения:

- $f(n) = \mathcal{O}(g(n)) \equiv \exists n_0, c > 0 : \forall n \geqslant n_0 : f(n) \leqslant c \cdot g(n)$
- $f(n) = \Omega(g(n)) \equiv \exists n_0, c > 0 : \forall n \geqslant n_0 : f(n) \geqslant c \cdot g(n)$
- $f(n) = \Theta(g(n)) \equiv \exists n_0, c_1, c_2 > 0 : \forall n \geqslant n_0 : c_1 \cdot g(n) \leqslant f(n) \leqslant c_2 \cdot g(n)$

Def. Когда мы записываем $T(n) = T(n-1) + f(n)$, это значит, что при построении *дерева рекурсивных вызовов* количество слоёв будет равно n и **ветвления не будет** (дерево будет выглядеть как прямая линия), на каждом слое функция будет делать: $f(n)$, $f(n-1)$, $f(n-2)$, …, $f(2)$, $f(1)$ операций соответственно, тогда для неё верно:

- $T(n) = f(n) + f(n-1) + f(n-2) + \ldots + f(2) + f(1)$
- Если $f(n) = n$, получаем $T(n) = \Theta(n^2)$
- Если $f(n) = log(n)$, получаем $T(n) = \Theta(n \cdot log(n))$
- Если $f(n) = 1$, получаем $T(n) = \Theta(n)$

Доказательство этих фактов остается читателю в качестве упражнения.

Def. Когда мы записываем $T(n) = 2 \cdot T(n-1) + f(n)$, это значит, что при построении *дерева рекурсивных вызовов* количество слоёв будет равно n и **ветвление будет** (дерево будет выглядеть как полное двоичное дерево), на каждом слое функция **в сумме по всему слою** будет делать: $f(n)$, $2 \cdot f(n-1)$, $4 \cdot f(n-2)$, …, $2^n \cdot f(1)$ операций соответственно, тогда для неё верно:

- $T(n) = f(n) + 2 \cdot f(n-1) + 4 \cdot f(n-2) + \ldots + 2^n \cdot f(1)$

Def. Когда мы записываем $T(n) = T(n/2) + f(n)$, это значит, что при построении *дерева рекурсивных вызовов* количество слоёв будет равно $log(n)$ и **ветвления не будет** (дерево будет выглядеть как прямая линия), на каждом слое функция будет делать: $f(n)$, $f(n/2)$, $f(n/4)$, …, $f(1)$ операций соответственно,

тогда для неё верно:

⋆ $T(n) = f(n) + f(n/2) + f(n/4) + \ldots + f(1)$

ДЗ

Докажите по определению — выберите константы c и n_0 и объясните соответствующее неравенство

Докажите или опровергните следующие утверждения. Для опровержения достаточно привести пример, на котором утверждение неверно. Все функции положительные.

1.1. Если $f(n) = \mathcal{O}(g(n))$, то $g(n) = \mathcal{O}(f(n))$

1.2. Если $f(n) = \mathcal{O}(g(n))$, то $2^{f(n)} = \mathcal{O}(2^{g(n)})$

1.3. $f(n) = \mathcal{O}(f(n)^2)$

1.4. $f(n) = \Theta(f(\frac{n}{2}))$

1.5. Если $f(n) = \mathcal{O}(g(n))$, то $g(n) = \Omega(f(n))$

Докажите по определению

1.6. Докажите по определению, что если $f_1(n) = \mathcal{O}(g_1(n))$ и $f_2(n) = \mathcal{O}(g_2(n))$, то $f_1(n) + f_2(n) = \mathcal{O}(g_1(n) + g_2(n))$.

1.7. Докажите по определению, что $\max(f(n), g(n)) = \Theta(f(n) + g(n))$, если $f(n)$ и $g(n)$ неотрицательны.

1.8. Докажите по определению, что $\sum\limits_{i=1}^{n+5} 2^i = \mathcal{O}(2^n)$.

1.9. Докажите по определению, что $\frac{n^3}{6} - 7n^2 = \Omega(n^3)$

1.10. Докажите по определению, что $\log_3(n^3 + 5n + 10) = \Theta(\log(n))$

1.11. Докажите по определению, что $2^n \cdot n^5 = \mathcal{O}(2.1^n)$

Пусть $p(n) = \sum\limits_{i=0}^{d} a_i n^i$, где $a_d > 0$ — полином степени d от n. Используя определения, полагая, что k — константа, докажите, что:

1.12. Если $k \geq d$, то $p(n) = \mathcal{O}(n^k)$

1.13. Если $k = d$, то $p(n) = \Theta(n^k)$

1.14. Если $k \leq d$, то $p(n) = \Omega(n^k)$

Следующий тип заданий:

1.15. Придумайте две положительные функции $f(n)$ и $g(n)$, такие что не выполнено ни одно из двух: $f(n) = \mathcal{O}(g(n))$ и $f(n) = \Omega(g(n))$.

1.16. Докажем, что $n^2 = \mathcal{O}(n)$ по индукции. Для этого докажем, что для любого k верно, что $kn = \mathcal{O}(n)$. Действительно, для $k = 1$ это верно. Пусть утверждение верно для $k-1$, тогда $kn = (k-1)n + n = \mathcal{O}(n)$. Таким образом, это верно для всех k, а значит и для $k = n$, таким образом $n \cdot n = \mathcal{O}(n)$. Где ошибка в этом доказательстве?

1.17. Докажите или опровергните, что $\log(n!) = \Theta(n \log n)$.

1.18. Докажите, что $\sum\limits_{i=1}^{n} \frac{n}{i} = \mathcal{O}(n \log n)$.

1.19. Для каждой из приведенных ниже программ найдите и аргументируйте точную $\mathcal{O}$-асимптотику времени ее работы.

(a)
```
for i = 0..n:
    for j = 0..i:
        for k = 0..j:
            print(i, j, k)
```

(b)
```
for i = 0..n:
    j = 0, k = 2 * i
    while j < k:
        j++, k--
```

(c)
```
i = 1
while i < n:
    for j = 0..i:
        print(i, j)
    i = i * 2
```

(d)
```
for i = 0..n:
    j = i
    while j > 0:
        j = j / 2
```

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

1.20. $T(n) = T(n-1) + n$

1.21. $T(n) = 3T\left(\frac{n}{2}\right) + 3$

1.22. $T(n) = T\left(\frac{n}{3}\right) + \log n$

Неделя 2. Квадратичные сортировки. Сортировка слиянием

2.23. В отсортированный массив длины n в произвольное место вставили случайное число. Придумайте алгоритм, который за $\mathcal{O}(n)$ сортирует полученный массив.

2.24. Дано k отсортированных массивов суммарной длины n. Опишите функцию `merge`, которая сливает эти массивы в один отсортированный массив за время $\mathcal{O}(nk)$.

Def. Сортировка называется *стабильной*, если она не меняет относительный порядок сортируемых элементов, имеющих одинаковые значения.

2.25. Для каждой из следующих сортировок покажите, является ли она стабильной. Если это зависит от реализации, покажите как именно реализовать стабильный вариант выбранной сортировки.

(a) сортировка выбором

(b) сортировка вставками

(c) сортировка слиянием

Def. *Перестановкой* размера n называется массив размера n, в котором каждое целое число от 1 до n встречается ровно по одному разу.

2.26. Даны два отсортированных массива a и b. Найдите такие i и j, что значение $|a_i - b_j|$ минимально. Время работы $\mathcal{O}(n)$.

2.27. Даны два отсортированных массива a и b. Найдите количество пар (i, j), таких что $a_i < b_j$. Время работы $\mathcal{O}(n)$.

2.28. Даны два отсортированных массива a и b. Найдите количество пар (i, j), таких что $a_i = b_j$. Время работы $\mathcal{O}(n)$.

2.29. Дан массив a. Назовем *инверсией* пару индексов (i, j), такую что $i < j$ и $a_i > a_j$. Пусть в массиве длины n ровно k инверсий. Докажите, что сортировка вставками работает за $\mathcal{O}(n + k)$.

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

2.31. Придумайте способ сделать сортировку слиянием «снизу вверх», без использования рекурсии.

2.32. Даны два массива a и b длины n. Найдите такую пару (i, j), что $a_i < a_j$ и $b_i < b_j$, либо определите, что такой пары не существует. Время работы $\mathcal{O}(n \log n)$.

2.33. Дан массив a длины n, состоящий из неотрицательных целых чисел. Найдите в данном массиве подотрезок $[l, r]$, такой что $a_l + a_{l+1} + \ldots + a_r = k$ либо определите, что его не существует. Время работы $\mathcal{O}(n)$.

2.34. Решите предыдущую задачу при условии, что нужно найти подотрезок максимальной длины.

2.35. Дан массив a длины n, состоящий из неотрицательных целых чисел. Найдите количество подотрезков массива, сумма которых равна k.

2.36. Дан массив a длины n. Необходимо для каждого элемента найти количество элементов, меньших его. Время работы $\mathcal{O}(n \log n)$.

2.37. Дан массив a длины n, состоящий из натуральных чисел. Требуется найти минимальное натуральное число, которого нет в массиве, за $\mathcal{O}(n \log n)$.

2.38. Дан массив a длины n. Постройте массив b длины n, состоящий из чисел от 0 до $n-1$, чтобы было выполнено следующее свойство: если $a_i < a_j$, то $b_i < b_j$. Время работы $\mathcal{O}(n \log n)$.

Неделя 3. Быстрая сортировка. K-я порядковая статистика

3.39. Приведите пример массива, на котором быстрая сортировка работает за $\Omega(n^2)$, если в качестве опорного элемента выбирается:

- ▷ Самый левый элемент отрезка
- ▷ Самый правый элемент отрезка
- ▷ Центральный элемент отрезка (то есть `a[(l + r) / 2]`)

3.40. То же самое, но в качестве разделителя выбирается медиана из этих трех элементов.

3.41. То же самое, но в качестве разделителя выбирается $(\frac{a[l]+a[r]+a[(l+r)/2]}{3})$

3.42. Предположим, что злоумышленник знает, по какому принципу выбирается опорный элемент в быстрой сортировке (например, он знает `seed` и алгоритм, генерирующий случайные числа). Как он может составить тест, на котором алгоритм будет работать за $\Omega(n^2)$?

3.43. У вас есть n болтов и n подходящих к ним гаек. Все болты (и, соответственно, гайки) имеют разный диаметр. Посмотрев на два болта (или две гайки), сложно понять, какой больше, а какой меньше, поэтому единственная операция, которую вы можете делать: взять какой-то болт и какую-то гайку и сравнить их диаметры. Найдите для каждого болта подходящую гайку за $\mathcal{O}(n \log n)$.

3.44. Дан массив a длины n, а также массив p длины m. Найдите p_i-ю порядковую статистику для всех i за $\mathcal{O}(m \log n + n)$.

3.45. Есть n ящиков, на каждом из которых написан уникальный номер. Необходимо отсортировать ящики по возрастанию номеров. У вас есть кран, который умеет выполнять одну команду: `swap(i, j)`, которая меняет местами ящики i и j. Постройте план работы крана, который позволит отсортировать все ящики за минимальное количество команд. Время работы (вашего алгоритма, а не крана) $\mathcal{O}(n \log n)$.

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

3.47. Есть массив длины n. Посчитайте, сколько есть способов упорядочить его элементы в неубывающем порядке.

3.48. На прямой живут n друзей, i-й друг живет в точке x_i. Они хотят встретиться в одной точке. Помогите им найти такую точку, чтобы суммарное расстояние, которое они пройдут, было бы минимально.

3.49. На прямой живут n друзей, i-й друг живет в точке x_i. Они хотят встретиться в одной точке. Помогите им найти такую точку, чтобы сумма квадратов расстояний, которое они пройдут, была бы минимальна.

3.50. Даны два массива a и b, состоящих из целых чисел. Проверьте, можно ли в массиве a выбрать k чисел, а в массиве b — m чисел так, что любое число, выбранное в первом массиве, строго меньше любого числа, выбранного во втором массиве.

3.51. Есть улица длины l, которая освещается n фонарями, i-й фонарь находится в точке a_i. Фонарь освещает все точки улицы, которые находятся от него на расстоянии не больше d, где d — некоторое положительное число, общее для всех фонарей. Найдите минимальное d, при котором вся улица освещена.

3.52. Есть массив, состоящий из n натуральных чисел. Можно уменьшать числа, но чтобы они оставались натуральными. Какое максимальное число различных чисел может быть в массиве после нескольких таких операций?

3.53. Дан отсортированный массив целых чисел размера n. За $\mathcal{O}(n)$ времени определите, можно ли разбить его элементы на две группы так, чтобы в каждой группе любые два элемента отличались хотя бы на k.

Неделя 4. Куча, сортировка кучей

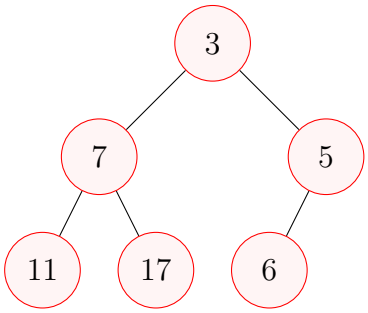
Конспект от непревзойденного Гриши Хлытина

▷ Структура данных «Двоичная куча»:

- Подвешенное двоичное дерево;
- Каждый слой, кроме последнего, заполнен полностью (количество вершин на i-ом слое равно 2^i);
- Последний слой заполнен слева-направо «без дырок»;
- Верно одно из двух:

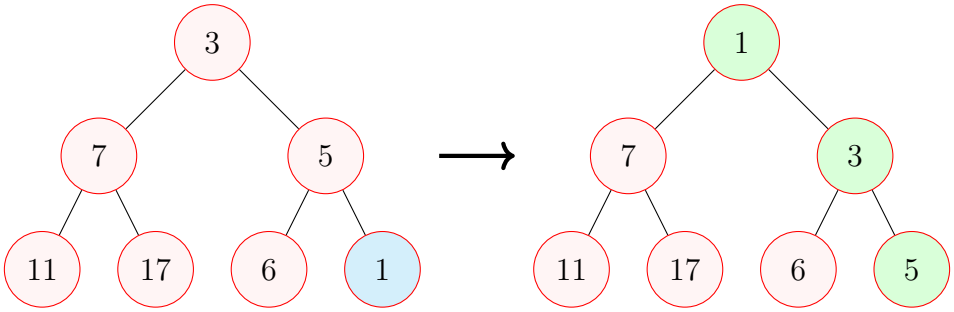
4.1. на всех ребрах написано отношение $\leqslant$ (значение в вершине нестрого меньше значений в её детях) — тогда говорят, что построена *куча по минимуму*;

4.2. на всех ребрах написано отношение $\geqslant$ (значение в вершине нестрого больше значений в её детях) — тогда говорят, что построена *куча по максимуму*;

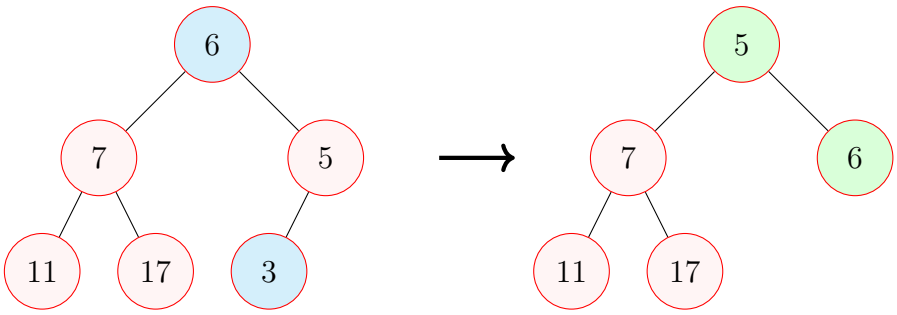


▷ Пусть дана куча «по минимуму». Какие операции в базовой версии с этой структурой данных мы умеем делать?

- Вспомогательная операция `siftUp(`x`)` — просеять элемент x вверх по дереву, работает за $\mathcal{O}(\log n)$;
- Вспомогательная операция `siftDown(`x`)` — просеять элемент x вниз по дереву, работает за $\mathcal{O}(\log n)$;
- Операция `add(`x`)` — добавить элемент x в кучу, работает за $\mathcal{O}(\log n)$: (Напоминание: добавляем x последней вершиной на последнем слое, а затем вызываем `siftUp(`x`)`)



- Операция `extractMin()` — извлечь текущий минимальный элемент из кучи, работает за $\mathcal{O}(\log n)$: (Напоминание: свапаем корень с последней вершиной на последнем слое, удаляем последнюю вершину на последнем слое, а затем вызываем `siftDown(root)`)



▷ Как хранить кучу на n элементов?

– Храним в массиве `heap`, достаточно выделить размер `heap`$[0 \dots 2n]$;
– Если вершина v хранится в i-й ячейке массива, то её непосредственные дети (если они существуют) хранятся в ячейках с индексами $2i+1$ и $2i+2$ (индексация массива с нуля);

ДЗ

4.54. Проиллюстрируйте построение кучи и работу алгоритма сортировки кучей на примере массива `[5, 10, 4, 10, 3, 8, 6, 10]`.

4.55. Пусть в двоичную кучу добавили n элементов. Сколько листьев у получившегося дерева?

4.56. Пусть в двоичную кучу в некотором порядке добавили все натуральные числа от 1 до 1 000 (каждое число было добавлено ровно один раз). Какое минимальное число может находиться на самом нижнем слое кучи?

4.57. Пусть в двоичную кучу добавили n элементов. Докажите, что в такой куче найдется не более, чем $\left\lceil \frac{n}{2^{h+1}} \right\rceil$ вершин с высотой h (*высота* — это расстояние от вершины до самого глубокого листа в поддереве этой вершины).

4.58. Придумайте способ модифицировать двоичную кучу «по минимуму», добавив новую операцию `decreaseKey(`v`, `x`)` — уменьшить значение в вершине v (которая в свою очередь хранится в i-й ячейке массива, i известно) на величину x, или покажите, что это невозможно. Время $\mathcal{O}(\log n)$.

4.59. Придумайте способ модифицировать двоичную кучу «по минимуму», добавив новую операцию `increaseKey(`v`, `x`)` — увеличить значение в вершине v (которая в свою очередь хранится в i-й ячейке массива, i известно) на величину x, или покажите, что это невозможно. Время $\mathcal{O}(\log n)$.

4.60. Придумайте способ модифицировать двоичную кучу «по минимуму», добавив новую операцию `remove(`v`)` — удалить вершину v (которая в свою очередь хранится в i-й ячейке массива, i известно), или покажите, что это невозможно. Время $\mathcal{O}(\log n)$.

4.61. Имеются k отсортированных по неубыванию массивов, суммарный размер всех массивов равен n. Придумайте алгоритм, позволяющий все массивы слить в один отсортированный массив размера n за $\mathcal{O}(n \log k)$.

4.62. На лекции был рассмотрен код, строящий двоичную кучу без использования дополнительной памяти за $\mathcal{O}(n \log n)$:

```
for (i = 0...n - 1):
    sift_up(i)
```

Теперь попробуем поменять операцию `sift_up` на операцию `sift_down`, получим следующий код:

```
for (i = 0...n - 1):
    sift_down(i, n)
```

Приведите пример массива, на котором данный алгоритм не строит корректную двоичную кучу. Напоминание:

- ▷ sift_down(i, n) — просеивание вниз, где i — индекс начального элемента, n — количество элементов в куче
- ▷ sift_up(i) — просеивание вверх

4.63. Докажите, что следующий алгоритм корректно строит кучу на заданном массиве:

```
for (i = n - 1...0):
    sift_down(i, n)
```

4.64. Докажите, что $\sum\limits_{i=0}^{\infty} \frac{1}{2^i} = 2$, а также, что $\sum\limits_{i=0}^{\infty} \frac{i}{2^i} = 2$. Данное доказательство пригодится в следующем задании.

4.65. Докажите, что код, приведенный в задании 4.63, строит двоичную кучу за $\mathcal{O}(n)$. При доказательстве можно считать факты из задания 4.64 доказанными.

4.66. Докажите или опровергните, что сортировка кучей является стабильной.

4.67. Даны две кучи с размерами n и m. Придумайте алгоритм, позволяющий слить данные кучи в одну кучу размера $n + m$ за $\mathcal{O}(n + m)$.

4.68. Дан массив длины n. Найдите в нем k минимальных элементов за $\mathcal{O}(n+k \log n)$.

4.69. Дан массив длины n. Найдите в нем k минимальных элементов за $\mathcal{O}(n+k \log k)$.

4.70. Пусть a — некоторый массив длины n, а b — массив a, отсортированный по неубыванию. Назовем *медианой* массива a элемент x, равный $b_{\frac{n}{2}}$. Придумайте структуру данных, позволяющую выполнять следующие операции за $\mathcal{O}(\log n)$:

- ▷ Добавить элемент
- ▷ Найти и удалить медиану

4.71. Придумайте структуру данных, позволяющую выполнять следующие операции за $\mathcal{O}(\log n)$:

- ▷ Добавить элемент
- ▷ Найти и удалить минимальный элемент
- ▷ Найти и удалить максимальный элемент

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

4.73. Назовем d-ичной кучей дерево, организованное следующем образом: у каждой вершины (кроме нижнего слоя) имеется ровно d детей. Какие номера имеют дети и родитель вершины с номером v? За какое время работают основные операции с кучей на таком дереве? Приведите оценку, зависящую от n и d.

4.74. Имеется d-ичная куча размера n. С ней выполняются m операций `extract_min` и k операций `decrease_key`. Выберите такое d, чтобы суммарное время работы было оптимально.

Неделя 5. Сортировка подсчетом. Цифровая сортировка.

5.75. Придумайте способ отсортировать массив строк при помощи цифровой сортировки. Оцените асимптотику полученного алгоритма. Оценка должна зависеть от суммы длин всех строк ($\sum|s_i|$) и размера алфавита $|\Sigma|$.

5.76. Рассмотрим массив из n целых чисел из диапазона $[0, n-1]$. Назовем i-м циклическим сдвигом исходного массива следующую последовательность чисел: $a_i, a_{i+1}, \ldots, a_{n-1}, a_0, a_1, \ldots, a_{i-1}$. Придумайте алгоритм, позволяющий отсортировать все циклические сдвиги массива за $\mathcal{O}(n^2 \log n)$.

5.77. Придумайте алгоритм, позволяющий отсортировать все циклические сдвиги массива за $\mathcal{O}(n^2)$.

5.78. Придумайте алгоритм, позволяющий отсортировать все циклические сдвиги массива за $\mathcal{O}(n \log n)$. Подсказка: отсортировав все циклические подмассивы длины L, можно за $\mathcal{O}(n)$ отсортировать все циклические подмассивы длины $2L$.

5.79. Есть массив, состоящий из n натуральных чисел. Найти минимальное натуральное число, которого нет в массиве, за $\mathcal{O}(n)$.

5.80. Оцените время работы сортировки слиянием, если на каждом уровне рекурсии мы разбиваем массив не на две части, а на k. Приведите оценку, зависящую от n и k. Считайте, что объединить k отсортированных массивов в один отсортированный массив можно за $\mathcal{O}(nk)$.

5.81. Оцените время работы сортировки слиянием, если на каждом уровне рекурсии мы разбиваем массив не на две части, а на k. Приведите оценку, зависящую от n и k. Считайте, что объединить k отсортированных массивов в один отсортированный массив можно за $\mathcal{O}(n \log k)$.

5.82. Дан массив длины n, состоящий из целых чисел. После этого поступают q запросов вида: «вычислить сумму $a_l + a_{l+1} + \ldots + a_r$». Научитесь отвечать на такие запросы за $\mathcal{O}(1)$. Разрешается использовать $\mathcal{O}(n)$ дополнительной памяти и выполнить некоторую предварительную обработку данных (далее мы будем называть это *препроцессингом*) за $\mathcal{O}(n)$.

5.83. Дан массив длины n, изначально заполненный нулями. После этого поступают q запросов вида: «прибавить к элементам $a_l, a_{l+1}, \ldots, a_r$ число x». Необходимо обработать все запросы и вывести итоговый массив. Время работы: $\mathcal{O}(n+q)$.

5.84. Дан массив из n целых чисел из диапазона $[0, k-1]$. Придумайте структуру данных, позволяющую за $\mathcal{O}(1)$ отвечать на запросы: сколько элементов массива лежат в диапазоне $[a, b]$ (то есть нужно найти количество таких i, что $a \le a_i \le b$). Время на препроцессинг $\mathcal{O}(n+k)$.

5.85. Дан массив длины n. Требуется найти подотрезок массива с максимальной суммой. Время работы: $\mathcal{O}(n)$.

5.86. Докажите, что не существует структуры данных, которая умеет выполнять следующие две операции асимптотически быстрее, чем за $\mathcal{O}(\log n)$:

- ▷ Добавить элемент
- ▷ Найти и удалить медианный элемент

5.87. Дан отсортированный массив длины n. Найдите в нем такие позиции x и y, что $a_x - a_y \leq k$ и $a_x - a_y$ максимально. Время работы: $\mathcal{O}(n)$.

5.88. Даны n точек на окружности, пронумерованных по часовой стрелке. Требуется найти две самые удаленные точки за $\mathcal{O}(n)$.

Неделя 6. Бинарный и тернарный поиск

6.89. Дан массив неотрицательных чисел длины n. Необходимо отвечать на запросы: какое максимальное количество чисел из начала массива можно взять, чтобы их сумма не превосходила x? Время на один запрос — $\mathcal{O}(\log n)$. Можно ли решить задачу таким же образом, если числа в массиве могут быть отрицательными?

Def. *Циклическим сдвигом* массива `arr` *вправо* на величину d называется массив, получаемый из исходного переносом последних d элементов в начало:

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

6.90. Дан массив длины n, полученный циклическим сдвигом отсортированного по возрастанию массива. Все элементы массива попарно различны. Требуется за $\mathcal{O}(\log n)$ найти в массиве заданное число x. Можно ли решить задачу таким же образом, если числа в массиве могут повторяться?

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

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

6.93. Пусть выполняется целочисленный бинарный поиск с начальными значениями L и R. Придумайте алгоритм, позволяющий за $\mathcal{O}(\log(R-L))$ определить, могла ли пара чисел (l, r) быть границами бинарного поиска в некоторый момент времени.

6.94. Дан отсортированный по возрастанию массив длины n, а также число x. Найдите такую позицию p, что $a[p] = x$ за $\mathcal{O}(\log p)$ (обратите внимание, что нужно решение за $\mathcal{O}(\log p)$, а не за $\mathcal{O}(\log n)$).

6.95. К отсортированному по **неубыванию** массиву **приписали справа** отсортированный по **невозрастанию** массив, 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:

(a) все элементы в массивах различны

(b) элементы могли повторяться

6.96. В некоторой компьютерной игре имеется n видов ресурсов. Для постройки одного здания требуется a_i единиц i-го вида ресурсов для всех i от 1 до n. У вас есть b_i единиц ресурса i, а также g единиц золота. Одну единицу золота можно обменять на c_i единиц ресурса i-го вида. Определите, какое максимальное количество зданий вы сможете построить.

6.97. В выборах участвуют n кандидатов. Вы выяснили, что за i-го кандидата планируют проголосовать a_i человек. Вы хотите для **каждого** кандидата посчитать сколько денег недо потратить, чтобы он победил в выборах (набрал строго больше голосов, чем все остальные кандидаты). За s рублей можно изменить мнение любого одного голосующего. (Решите эту задачу в ограничениях $1 \le n \le 10^5$ и $TL = 1$ секунда)

6.98. Пусть $f(x) = \sum\limits_i |a_i \cdot x + b_i|$ для некоторых (неизвестных вам) a_i и b_i. Можно ли найти минимум функции $f(x)$ при помощи тернарного поиска?

6.99. Дан отсортированный по возрастанию массив длины n. Требуется выбрать k чисел из массива таким образом, чтобы минимальная разность соседних двух выбранных чисел была как можно больше. То есть, формально, если вы выбрали числа $i_1, i_2, \ldots, i_k$, то нужно максимизировать $\min\limits_{j=1}^{k-1} a[i_{j+1}] - a[i_j]$.

6.100. Домик черепашки расположен в точке 0 числовой прямой. Однажды она узнала, что скоро вырастут n одуванчиков. Одуванчик с номером i вырастет в точке x_i в момент времени t_i, причем все x_i и t_i положительные, а так же $x_{i+1} > x_i$ и $t_{i+1} > t_i$. Черепашка выходит из дома в момент 0, она движется со скоростью не больше v. Чтобы съесть одуванчик, ей нужно остановиться около него на время d. За какое время она сможет съесть все одуванчики и вернуться домой?

6.101. Дан массив длины n, состоящий из нулей и единиц, причем первый элемент массива равен нулю, а последний элемент — единице. Найдите такую позицию i, что $a[i] = 0$ и $a[i+1] = 1$ за $\mathcal{O}(\log n)$.

6.102. Придумайте модификацию двоичной кучи, в которой операция `sift_up` делает $\mathcal{O}(\log \log n)$ сравнений элементов (очевидно, что количество обменов при этом уменьшить не получится).

6.103. Вам дан отсортированный по возрастанию массив в котором все числа кроме одного встречаются по два раза, а отсавшееся число встречается один раз. Найдите это число обращаясь к элементам массива $\mathcal{O}(\log n)$ раз.

6.104. Есть многочлен нечетной степени $P(x) = a_{2 \cdot k+1} \cdot x^{2 \cdot k+1} + a_{2 \cdot k} \cdot x^{2 \cdot k} + \ldots + a_1 \cdot x^1 + a_0$. Найдите любой корень уравнения (вы не знаете сам многочлен)

Неделя 7. Стек. Очередь. Амортизационный анализ

7.105. Добавьте в стек и очередь операцию `get_sum()`, которая будет возвращать сумму всех элементов, находящихся в структуре данных. Время работы операции $\mathcal{O}(1)$. Дополнительная память $\mathcal{O}(1)$.

7.106. Добавьте в стек операцию `get_min()`, которая будет возвращать минимальный элемент, хранимый в стеке. Время работы операции $\mathcal{O}(1)$. Дополнительная память $\mathcal{O}(n)$, где n — количество элементов в стеке.

7.107. Придумайте способ реализовать очередь, используя два стека. Подсказка: один стек должен использоваться для добавления элементов, а второй — для удаления элементов. Все операции с очередью должны иметь амортизированное время работы $\mathcal{O}(1)$. Докажите эту оценку при помощи метода бухгалтерского счета или метода потенциалов.

7.108. Добавьте в очередь операцию `get_min()`, которая будет возвращать минимальный элемент, хранимый в очереди. Время работы операции $\mathcal{O}(1)$. Дополнительная память $\mathcal{O}(n)$, где n — количество элементов в очереди.

7.109. При помощи стека вычислите значение арифметического выражения, записанного в постфиксной записи (запись, в которой оператор записывается после операндов, например, выражение `4 - ((1 + 2) * 3)` будет записано как `4 1 2 + 3 * -`).

7.110. При помощи стека вычислите значение арифметического выражения, записанного в инфиксной записи (обычные выражения). Для простоты можно считать, что в выражении каждая операция взята в скобки, например `(4 - ((1 + 2) * 3))`.

7.111. Научитесь по выражению в инфиксной записи строить выражение в постфиксной записи.

7.112. Модифицируем стек, рассмотренный на лекции, который увеличивал размер массива в два раза при нехватке памяти для сохранения добавленного элемента. Во время операции `pop` будем уменьшать размер стека в два раза, если размер массива больше, чем количество хранимых элементов, хотя бы в четыре раза. Докажите при помощи метода бухгалтерского счета или при помощи метода потенциалов, что в таком стеке операции `push` и `pop` имеют амортизированное время работы $\mathcal{O}(1)$.

7.113. Дан массив из целых чисел. Для каждого числа требуется найти ближайшее слева число, меньшее данного (то есть формально для каждого i нужно найти такое $j < i$, что $a_j < a_i$ и j максимально). Суммарное время работы $\mathcal{O}(n)$.

7.114. Проанализируйте время работы стека, если расширение при нехватке памяти во время операции `push` происходит в $k > 1$ раз. Какое значение k лучше выбрать на практике?

7.115. Реализуйте очередь на базе расширяющегося и сужающегося массива с амортизированным временем работы всех операций $\mathcal{O}(1)$.

7.116. Добавьте в стек операцию `clear`, удаляющую все элементы из стека. Амортизированное время работы операции должно быть $\mathcal{O}(1)$. Докажите данную оценку.

7.117. Добавим в стек операцию `multipop(k)`, последовательно удаляющую k элементов из стека (гарантируется, что в стеке есть хотя бы k элементов). Докажите или опровергните, что амортизированное время работы всех операций равно $\mathcal{O}(1)$.

7.118. Помимо операции `multipop(k)` добавим операцию `multipush(k, x)`, последовательно добавляющую в стек k элементов, равных x. Докажите или опровергните, что амортизированное время работы всех операций равно $\mathcal{O}(1)$.

7.119. Битовый счетчик хранит целое положительное число в виде массива двоичных цифр. Изначально все цифры равны 0. Операция `increment` увеличивает счетчик на 1. Реализуйте операцию `increment` и докажите, что ее амортизированное время работы равно $\mathcal{O}(1)$.

7.120. Как изменится время работы, если добавить операцию `add(k)`, прибавляющую к счетчику 2^k?

7.121. Как изменится время работы, если добавить операцию `decrement`, уменьшающую счетчик на 1?

7.122. Добавьте битовому счетчику операцию `clear`, присваивающую ему 0. Амортизированная стоимость операции должна быть $\mathcal{O}(1)$.

Неделя 8. Связные списки. Pointer Machine. Задачи на пройденные темы

8.123. Дан набор из n элементов, у каждого элемента есть ссылка на какой-то другой элемент. Проверьте, правда ли все элементы образуют один большой кольцевой список (менять ссылки нельзя). Время работы $\mathcal{O}(n)$, можно использовать $\mathcal{O}(1)$ дополнительной памяти.

8.124. Задан односвязный список, возможно, с циклами (для каждого элемента известен `node.next`). Предложите алгоритм, который находит длину цикла p, достижимого из заданной вершины v, а также длину пути перед циклом s, используя не более, чем $\mathcal{O}(1)$ дополнительной памяти. Время работы: $\mathcal{O}(s+p)$. Сам список менять нельзя.

8.125. Предложите способ развернуть односвязный список за $\mathcal{O}(n)$, используя $\mathcal{O}(1)$ дополнительной памяти.

8.126. Научитесь сливать два отсортированных односвязных списка в один за время $\mathcal{O}(n)$, используя $\mathcal{O}(1)$ дополнительной памяти.

8.127. Отсортируйте связный список за время $\mathcal{O}(n \log n)$, используя $\mathcal{O}(\log n)$ дополнительной памяти.

8.128. Придумайте способ хранения связного списка, который позволит разворачивать его за $\mathcal{O}(1)$, сохранив возможность выполнять прочие операции.

8.129. Есть таблица размера $n \times n$. С ней производятся m операций следующего вида: вырезать прямоугольный кусок таблицы, развернуть его на 180 градусов, и вставить обратно. Выведите состояние таблицы после выполнения всех операций. Время работы: $\mathcal{O}(nm)$.

8.130. Реализуйте двоичную кучу в Pointer Machine Model.

8.131. Как сделать в Pointer Machine Model структуру данных, позволяющую выполнять двоичный поиск в связном списке (список отсортирован и не меняется).

8.132. Счетчик — это целое число, изначально равное нулю, с которым можно производить две операции: инкремент и декремент. Реализуйте полностью персистентный счетчик с $\mathcal{O}(1)$ времени на операцию и $\mathcal{O}(1)$ дополнительной памяти на каждую версию счетчика.

8.133. Придумайте, как реализовать частично персистентную двоичную кучу с временем на одну операцию $\mathcal{O}(\log n)$ и дополнительной памятью на каждую версию $\mathcal{O}(\log n)$.

8.134. Придумайте, как реализовать полностью персистентную двоичную кучу с временем на одну операцию $\mathcal{O}(\log n)$ и дополнительной памятью на каждую версию $\mathcal{O}(\log n)$.

8.135. Покажите как сделать в Pointer Machine Model структуру, которая позволяет делать двоичный поиск в списке (список отсортирован и не меняется, требуется отвечать на запросы: найти максимальный элемент не больше x).

Неделя 9. Система непересекающихся множеств

9.136. Добавьте в СНМ операции `getMin(x)`„ вычисляющие минимум, максимум и сумму элементов множества, в котором лежит элемент с номером x. Время работы: $\mathcal{O}(\alpha(m, n))$.

9.137. Добавьте в СНМ операции `getMax(x)`„ вычисляющие минимум, максимум и сумму элементов множества, в котором лежит элемент с номером x. Время работы: $\mathcal{O}(\alpha(m, n))$.

9.138. Добавьте в СНМ операции `getSum(x)`„ вычисляющие минимум, максимум и сумму элементов множества, в котором лежит элемент с номером x. Время работы: $\mathcal{O}(\alpha(m, n))$.

9.139. В изначально пустой граф по одному добавляются q ребер. После каждого добавления нужно вычислить размер максимальной по количеству вершин компоненты связности графа. Время работы: $\mathcal{O}(q \cdot \alpha(m, n))$.

9.140. Дан пустой граф на n вершинах. Требуется выполнять запросы двух типов:

(a) Добавить ребро в граф.

(b) Найти число ребер в компоненте связности, в которой находится вершина x.

Время работы на одну операцию: $\mathcal{O}(\alpha(m, n))$.

9.141. Есть пустой граф. Делают два вида запросов:

▷ добавить ребро

▷ найти число компонент связности, являющихся деревьями

9.142. Дан массив a, состоящий из n нулей. Требуется выполнять операции:

(a) Присвоить единицу i-му элементу массива.

(b) Найти количество непрерывных отрезков, состоящих из единиц.

Время работы на одну операцию: $\mathcal{O}(\alpha(m, n))$.

9.143. Дан массив a, состоящий из n нулей. Требуется выполнять операции:

(a) Присвоить единицу i-му элементу массива.

(b) Найти ближайший ноль к i-му элементу массива.

Время работы на одну операцию: $\mathcal{O}(\alpha(m, n))$.

9.144. Дан граф, из которого последовательно q раз удаляется по одному ребру. После каждого удаленного ребра требуется вывести количество компонент связности в графе. Время работы: $\mathcal{O}(q \cdot \alpha(m, n))$.

9.145. Дан массив a из n положительных чисел. Найдите отрезок $[l, r]$ с максимальной величиной $\left(\sum\limits_{i=l}^{r} a_i\right) \cdot \min\limits_{i=l}^{r} a_i$. Время работы — $\mathcal{O}(n \log n)$. Можно решать без СНМ.

Неделя 10. Динамическое программирование — 1

10.146. Котик прыгает с кочки 1 на кочку n, длина каждого прыжка может быть от 1 до k. У каждой кочки есть стоимость. Найдите путь с минимальной стоимостью за $\mathcal{O}(n)$.

10.147. Котик прыгает с кочки 1 на кочку n, длина каждого прыжка может быть 1 или 2. У каждой кочки есть стоимость. Найдите путь с минимальной стоимостью и количество путей с минимальной стоимостью за $\mathcal{O}(n)$.

10.148. Кузнечик прыгает из клетки 1 в клетку n, длина прыжка может быть от 1 до k. У каждой клетки есть стоимость. Найти число различных путей минимальной стоимости. Время $\mathcal{O}(n)$.

10.149. Котик прыгает с кочки 1 на кочку n, длина каждого прыжка может быть 1 или 2. На каждой кочке написана буква. Найдите такой путь, чтобы строка, которую прочитает котик, посещая кочки, была лексикографически минимальной за $\mathcal{O}(n^2)$.

10.150. Черепаха ползет из клетки $(1, 1)$ в клетку (n, m), каждый раз перемещаясь в соседнюю справа или снизу клетку. В каждой клетке растет некоторое количество цветов. Посещая клетку, черепаха съедает все цветы, растущие в ней. Найдите максимальное **нечетное** количество цветов, которое она сможет съесть (пропускать цветы на пути нельзя). Время работы $\mathcal{O}(nm)$.

10.151. Черепашка ползет из клетки $(0, 0)$ в клетку (n, m) и собирает цветочки. На каждой клетке растут красные или синие цветочки. Черепашка хочет пройти таким путем, чтобы число клеток с синими и красными цветочками было одинаковым. Помогите ей найти такой путь. Время $\mathcal{O}(n \cdot m)$.

10.152. Черепаха ползет из клетки $(1, 1)$ в клетку (n, m), каждый раз перемещаясь в соседнюю справа или снизу клетку. В каждой клетке записано целое число a_{ij}. Если это число положительное, то черепаха получает a_{ij} монет, а иначе — платит a_{ij} монет. Сколько монет нужно иметь черепахе в начале, чтобы добраться до конца пути таким образом, чтобы количество монет у нее всегда было неотрицательным. Время работы: $\mathcal{O}(nm \log C)$, где C — ответ.

10.153. У Васи есть калькулятор, который умеет выполнять три операции: прибавить 1, умножить на 2 и умножить на 3. Какое наименьшее количество операций нужно сделать, чтобы получить из числа 1 число n? Время работы $\mathcal{O}(n)$.

10.154. В очереди за билетами на концерт Моргенштерна стоят n школьников. Чтобы увеличить скорость движения очереди, два подряд идущих школьника могут договориться, и тот, что идет раньше в очереди купит два билета: на себя и на следующего. Школьник i тратит a_i секунд на покупку одного билета и b_i секунд на покупку двух билетов. За какое минимальное время все смогут купить билеты?

10.155. Вася любит ходить в кино и решать домашку по АиСД. Еще Вася любит разнообразие, поэтому он никогда не делает одно и тоже два дня подряд. Для n последовательных дней известно, есть ли в этот день интересное кино и есть ли в этот день интересная домашка. Какое минимальное число дней Вася будет сидеть без дела?

10.156. Посчитайте количество битовых последовательностей длины n, в которых не встречается подстрока 111. Время работы $\mathcal{O}(n)$.

Неделя 11. Динамическое программирование — 2

11.157. Дана строка s длины n. Вычислите для каждой пары (i, j) длину наибольшего общего префикса строк $s[i \dots n]$ и $s[j \dots n]$. Время работы $\mathcal{O}(n^2)$.

11.158. Дана строка s длины n и строка t длины m. За одну операцию можно удалить любой символ из строки s, добавить любой символ в любое место стоки s, либо поменять любой символ строки s. Найдите минимальное количество операций, за которое можно прекратить строку s в строку t. Время работы $\mathcal{O}(nm)$.

11.159. Как изменится решение предыдущей задачи, если у каждой операции есть своя стоимость и нужно вычислить, за какую минимальную стоимость можно преобразовать s в t?

11.160. Дан массив длины n. Найдите наидлиннейшую подпоследовательность массива, в которой каждый следующий элемент делится на предыдущий. Время работы $\mathcal{O}(n^2)$.

11.161. Дан массив a длины n и массив b длины n. Найдите длину их наибольшей общей возрастающей подпоследовательности. Время работы $\mathcal{O}(n^3)$.

11.162. У колдуна есть n заклинаний и m единиц маны. Заклинание i тратит c_i единиц маны и наносит противнику d_i единиц урона. Убейте монстра, обладающего h единицами здоровья, потратив как можно меньше маны. Время работы: $\mathcal{O}(nm)$.

11.163. Петя и Вася выиграли на олимпиаде n призов, для каждого из которых известна его стоимость. Суммарная стоимость призов равна r. Друзья хотят разделить призы таким образом, чтобы разница в суммарной стоимости призов Пети и призов Васи была как можно меньше. Время работы: $\mathcal{O}(nr)$.

11.164. Петя и Вася выиграли на олимпиаде $n = 2k$ призов, для каждого из которых известна его стоимость. Суммарная стоимость призов равна r. Друзья хотят разделить призы таким образом, чтобы каждому из них досталось ровно k призов и чтобы разница в суммарной стоимости призов Пети и призов Васи была как можно меньше. Время работы: $\mathcal{O}(n^2 r)$.

11.165. Назовем строку *палиндромом*, если она читается одинаково слева-направо и справа-налево (то есть $s_1 = s_n, s_2 = s_{n-1}, \dots$). Дана строка s длины n. Найдите наидлиннейшую подпоследовательность s, являющуюся палиндромом.

11.166. Дан массив a длины n и массив b длины n. Найдите длину их наибольшей общей возрастающей подпоследовательности. Время работы $\mathcal{O}(n^3)$.

11.167. Имеется следующий код:

```
for (int s = mask; s > 0; s = (s - 1) & mask)
    print(s)
print(0)
```

Докажите, что данный код выводит все подмаски маски $mask$, упорядоченные по убыванию. Здесь «&» обозначает побитовое «И».

11.168. Имеется следующий код:

```
for (int mask = 0; mask < (1 << n); ++mask)
    for (int s = mask; s > 0; s = (s - 1) & mask)
        // Some work...
```

Докажите, что данные два цикла совершают $\mathcal{O}(3^n)$ итераций суммарно.

11.169. Есть n различных номиналов монет, i-й номинал равен v_i. Монет каждого номинала бесконечное число. Нужно дать сдачу P за наименьшее число монет.

11.170. У Пети есть n книг, отсортированных по названию. Книга i имеет высоту h_i. Петя хочет сложить книги в шкаф в отсортированном порядке: первые несколько книг положить на первую полку, следующие несколько книг — на вторую полку, и так далее. На каждую полку помещается не более, чем m книг. Высота полки равна максимальной высоте книги на полке. Помогите Пете распределить книги таким образом, чтобы суммарная высота полок была как можно меньше. Время работы $\mathcal{O}(nm)$.

Неделя 12. Динамическое программирование — 3

12.171. Петя и Вася выиграли на олимпиаде n призов, для каждого из которых известна его стоимость. Суммарная стоимость призов равна r. Друзья хотят разделить призы таким образом, чтобы разница в суммарной стоимости призов Пети и призов Васи была как можно меньше. Время работы: $\mathcal{O}(2^{\frac{n}{2}} \cdot n)$.

12.172. Петя и Вася выиграли на олимпиаде $n = 2k$ призов, для каждого из которых известна его стоимость. Суммарная стоимость призов равна r. Друзья хотят разделить призы таким образом, чтобы каждому из них досталось ровно k призов и чтобы разница в суммарной стоимости призов Пети и призов Васи была как можно меньше. Время работы: $\mathcal{O}(2^{\frac{n}{2}} \cdot n)$.

12.173. Дан неориентированный граф. Требуется покрасить каждую его вершину в некоторый цвет таким образом, чтобы никакие две вершины, соединенные ребром, не были окрашены в один цвет. При этом требуется минимизировать количество различных цветов. Время работы: $\mathcal{O}(3^n)$.

12.174. Дан набор из n целых чисел. Требуется разбить его на минимальное количество наборов, в каждом из которых НОД всех чисел больше единицы. Время работы: $\mathcal{O}(3^n)$.

12.175. Заданы n предметов, у каждого имеется некоторый вес. За один раз вы можете унести с собой некоторое количество предметов, вес которых не превышает S. Какое минимальное количество подходов нужно сделать, чтобы унести все предметы? Время работы: $\mathcal{O}(2^n \cdot n)$.

12.176. Задан массив длины n. Найдите в этом массиве **подпоследовательность** максимальной длины, которая является *плавной*. Последовательность называется *плавной*, если модуль разности соседних элементов ровно 1. Время работы: $\mathcal{O}(n^2)$.

12.177. Имеется n предметов с весами w_i и стоимостями c_i. Нужно выбрать некоторое множество предметов, суммарный вес которых не превосходит W, максимизируя суммарную стоимость выбранных предметов. При этом запрещается брать в множество предметы, номера которых отличаются на 1: например, если в множество взят предмет с номером i, то запрещается брать предметы с номерами $i - 1$ и $i + 1$. Время работы: $\mathcal{O}(n \cdot W)$.

12.178. Дана последовательность чисел. Найдите максимальную по длине подпоследовательность, являющуюся арифметической прогрессией. Время работы: $O(n^3)$.

12.179. Дана последовательность из n круглых, квадратных и фигурных скобок. Удалите минимальное число скобок, чтобы получилась правильная скобочная последовательность. Время $O(n^3)$.

12.180. Даны два массива $p_1, p_2, \ldots, p_n$ и $q_1, q_2, \ldots, q_m$. Массивы p и q являются перестановками (все числа попарно различны и лежат в диапазоне от 1 до длины массива). Найти НОП массиов p и q за $\mathcal{O}(N \log N)$, где $N = \max(n, m)$.

12.181. Вычислить количество чисел, не превосходящих 10^n, в которых каждые две соседние цифры имеют НОД, равный 1.

12.182. Деревня Гадюкино представляет собой n домов, расположенных вдоль прямой дороги. Координата i-го дома a_i. Компания «интернет в каждый дом» хочет поставить в деревне несколько вайфай-точек, покрывающих все дома. Точка с радиусом покрытия r стоит $A + Br^2$ рублей. Найдите минимальную стоимость необходимых точек. Время $O(n^2)$.

12.183. Вам нужно распилить бревно длины L на несколько частей. Точки распила уже отмечены, всего есть n точек, расстояние от левого конца до i-й точки распила a_i. Лесопилка за один раз может сделать один распил, при этом стоимость равна длине бревна, которое нужно распилить. В каком порядке нужно делать распилы, чтобы суммарная стоимость была минимальна? Время $O(n^3)$.