Алгоритмы и структуры данных

Лекция 1. Введение. Асимптотический анализ. Бинарный поиск.

Артур Кулапин

ВШПИ МФТИ

6 сентября 2023 г.

О чем разговор?

1. Организационная информация
   - Экзамен
   - Зачет
   - О плагиате
2. Сложность программы
   - Ресурсы
   - Асимптотический анализ
3. Бинарный поиск
   - Классический бинарный поиск
   - Бинарный поиск по ответу
   - Вещественный бинарный поиск

Организационная информация

В рамках курса ставится две оценки.

Организационная информация

В рамках курса ставится две оценки.

- Зачет с оценкой по практикуму.
- Экзамен.

Экзамен

- Проходит в зимнюю сессию. Точные дата и место TBD.

Экзамен

- Проходит в зимнюю сессию. Точные дата и место TBD.
- Работа в семестре не влияет на оценку.

Экзамен

- Проходит в зимнюю сессию. Точные дата и место TBD.
- Работа в семестре не влияет на оценку.
- Формат проведения и правила TBD.

Зачет

- Формат оценивания — дифференцированный зачет (с оценкой).
- Оценка складывается из работы в семестре.

Из чего складывается оценка?

Зачет

- Формат оценивания — дифференцированный зачет (с оценкой).
- Оценка складывается из работы в семестре.

Из чего складывается оценка?

- Контесты. За них можно набрать до 5 баллов. Балл за них вычисляется по формуле: $C = 5 \cdot \frac{\text{your points}}{\text{total points}}$.

Зачет

- Формат оценивания — дифференцированный зачет (с оценкой).
- Оценка складывается из работы в семестре.

Из чего складывается оценка?

- Контесты. За них можно набрать до 5 баллов. Балл за них вычисляется по формуле: $C = 5 \cdot \frac{\text{your points}}{\text{total points}}$.
- Лабораторные работы. За них можно набрать до 3 баллов. Балл за них вычисляется по формуле: $L = 3 \cdot \frac{\text{your points}}{\text{total points}}$.

Зачет

- Формат оценивания — дифференцированный зачет (с оценкой).
- Оценка складывается из работы в семестре.

Из чего складывается оценка?

- Контесты. За них можно набрать до 5 баллов. Балл за них вычисляется по формуле: $C = 5 \cdot \frac{\text{your points}}{\text{total points}}$.
- Лабораторные работы. За них можно набрать до 3 баллов. Балл за них вычисляется по формуле: $L = 3 \cdot \frac{\text{your points}}{\text{total points}}$.
- Теоретические задания. За них можно набрать до 3 баллов. Балл за них вычисляется по формуле: $T = 3 \cdot \frac{\text{your points}}{\text{total points}}$.

Зачет

- Формат оценивания — дифференцированный зачет (с оценкой).
- Оценка складывается из работы в семестре.

Из чего складывается оценка?

- Контесты. За них можно набрать до 5 баллов. Балл за них вычисляется по формуле: $C = 5 \cdot \frac{\text{your points}}{\text{total points}}$.
- Лабораторные работы. За них можно набрать до 3 баллов. Балл за них вычисляется по формуле: $L = 3 \cdot \frac{\text{your points}}{\text{total points}}$.
- Теоретические задания. За них можно набрать до 3 баллов. Балл за них вычисляется по формуле: $T = 3 \cdot \frac{\text{your points}}{\text{total points}}$.

Итоговая оценка: $\min(10, [C + L + T - F])$

О плагиате

- Проверка на антиплагиат выполняется автоматически, а далее вручную перепроверяется преподавательским составом.

О плагиате

- Проверка на антиплагиат выполняется автоматически, а далее вручную перепроверяется преподавательским составом.
- Санкции накладываются одинаково на обоих участников кейса.

О плагиате

- Проверка на антиплагиат выполняется автоматически, а далее вручную перепроверяется преподавательским составом.
- Санкции накладываются одинаково на обоих участников кейса.
- Буква `F` с прошлого слайда соответствует числу задач, в которых был обнаружен плагиат.

О плагиате

- Проверка на антиплагиат выполняется автоматически, а далее вручную перепроверяется преподавательским составом.
- Санкции накладываются одинаково на обоих участников кейса.
- Буква `F` с прошлого слайда соответствует числу задач, в которых был обнаружен плагиат.
- Бонус! Если $F > 2$, то запускается протокол с дисциплинаркой, установленный в МФТИ.

Хорошая ли программа?

- Как понять, что одна программа лучше другой?

Хорошая ли программа?

- Как понять, что одна программа лучше другой?
- В чем нужно измерять «хорошесть» программы?

Хорошая ли программа?

- Как понять, что одна программа лучше другой?
- В чем нужно измерять «хорошесть» программы?
- Что делать с тем, что исполнители имеют разную производительность?

О ресурсах

Нас будут интересовать только два ресурса.

- Время исполнения или временная сложность.
- Потребляемая память или пространственная сложность.

О ресурсах

Нас будут интересовать только два ресурса.

- Время исполнения или временная сложность.
- Потребляемая память или пространственная сложность.

Модель памяти.

- Память вычислителя безгранична.
- Доступ к памяти на чтение/запись требует пренебрежимо малого времени.

$\mathcal{O}$-нотация

Def. Пусть имеются две функции $f(n)$ и $g(n)$, при этом $f, g : \mathbb{N} \to \mathbb{N}$, тогда считается, что $f(n) = \mathcal{O}(g(n))$, если

$\exists C > 0 \; \exists N_0 : \; \forall n > N_0 \; f(n) \leq C \cdot g(n)$.

$\mathcal{O}$-нотация

Def. Пусть имеются две функции $f(n)$ и $g(n)$, при этом $f, g : \mathbb{N} \to \mathbb{N}$, тогда считается, что $f(n) = \mathcal{O}(g(n))$, если
$\exists C > 0 \; \exists N_0 : \; \forall n > N_0 \; f(n) \leq C \cdot g(n)$.

Примеры

- Пусть $f(n) = n$, а $g(n) = n^2$, тогда очевидно, что $f(n) = \mathcal{O}(g(n))$.
  Например, пусть $C = 1$, а $N_0 = 2$, тогда
  $n = f(n) < C \cdot g(n) = g(n) = n^2$, что верно для $n \geq 2$.

$\mathcal{O}$-нотация

Def. Пусть имеются две функции $f(n)$ и $g(n)$, при этом $f, g : \mathbb{N} \to \mathbb{N}$, тогда считается, что $f(n) = \mathcal{O}(g(n))$, если
$\exists C > 0 \; \exists N_0 : \; \forall n > N_0 \; f(n) \leq C \cdot g(n)$.

Примеры

- Пусть $f(n) = n$, а $g(n) = n^2$, тогда очевидно, что $f(n) = \mathcal{O}(g(n))$. Например, пусть $C = 1$, а $N_0 = 2$, тогда $n = f(n) < C \cdot g(n) = g(n) = n^2$, что верно для $n \geq 2$.
- Пусть $f(n) = P_k$, а $g(n) = P_{k+\alpha}$, где P_r — многочлен степени r, $k \in \mathbb{N}$, $\alpha \in \mathbb{N}$. Тогда также нетрудно показать, что $f(n) = \mathcal{O}(g(n))$.

Больше букв

Def. Пусть имеются две функции $f(n)$ и $g(n)$, при этом $f, g : \mathbb{N} \to \mathbb{N}$, тогда считается, что $f(n) = \Omega(g(n))$, если
$\exists C > 0 \; \exists N_0 : \; \forall n > N_0 \; f(n) \geq C \cdot g(n)$.

Больше букв

Def. Пусть имеются две функции $f(n)$ и $g(n)$, при этом $f, g : \mathbb{N} \to \mathbb{N}$, тогда считается, что $f(n) = \Omega(g(n))$, если
$\exists C > 0 \; \exists N_0 : \; \forall n > N_0 \; f(n) \geq C \cdot g(n)$.

Def. Пусть имеются две функции $f(n)$ и $g(n)$, при этом $f, g : \mathbb{N} \to \mathbb{N}$, тогда считается, что $f(n) = \Theta(g(n))$, если
$\exists C_1, C_2 > 0 \; \exists N_0 : \; \forall n > N_0 \; C_1 \cdot g(n) \leq f(n) \leq C_2 \cdot g(n)$.

Бинарный поиск

Дан отсортированный массив $A[1 : n]$. Надо проверить, есть ли в нем элемент X.

Бинарный поиск

Дан отсортированный массив $A[1 : n]$. Надо проверить, есть ли в нем элемент X.

Очевидное решение — пройтись в цикле по массиву и проверить, есть ли X. Но зачем целый слайд про это?

Бинарный поиск

Дан отсортированный массив $A[1 : n]$. Надо проверить, есть ли в нем элемент X.

Очевидное решение — пройтись в цикле по массиву и проверить, есть ли X. Но зачем целый слайд про это?

Обозначим за $l = 1$ и $r = n$ левую и правую границы рассматриваемого подмассива.

1. Если диапазон валиден, то проверим, $A\left[\lceil \frac{l+r}{2} \rceil\right] \geq X$. Иначе вернем границу.

Бинарный поиск

Дан отсортированный массив $A[1 : n]$. Надо проверить, есть ли в нем элемент X.

Очевидное решение — пройтись в цикле по массиву и проверить, есть ли X. Но зачем целый слайд про это?

Обозначим за $l = 1$ и $r = n$ левую и правую границы рассматриваемого подмассива.

1. Если диапазон валиден, то проверим, $A\left[\left\lceil \frac{l+r}{2} \right\rceil\right] \geq X$. Иначе вернем границу.
2. Если полученный результат истинен, то запускаем рекурсивно поиск в левой половине массива.
3. Иначе — в правой половине массива.

Сложность

Что делает алгоритм?

1. Сравнивает средний элемент с X за $\mathcal{O}(1)$.
2. Решает, продолжать в левой или правой половине за $\mathcal{O}(1)$.
3. Запускается рекурсивно, уменьшая область поиска в два раза.

Сложность

Что делает алгоритм?

1. Сравнивает средний элемент с X за $\mathcal{O}(1)$.
2. Решает, продолжать в левой или правой половине за $\mathcal{O}(1)$.
3. Запускается рекурсивно, уменьшая область поиска в два раза.

Результат будет получен, когда длина диапазона (далее n) станет равной 1.

Сложность

Что делает алгоритм?

1. Сравнивает средний элемент с X за $\mathcal{O}(1)$.
2. Решает, продолжать в левой или правой половине за $\mathcal{O}(1)$.
3. Запускается рекурсивно, уменьшая область поиска в два раза.

Результат будет получен, когда длина диапазона (далее n) станет равной 1. А значит всего шагов: $\log_2 n$

Сложность

Что делает алгоритм?

1. Сравнивает средний элемент с X за $\mathcal{O}(1)$.
2. Решает, продолжать в левой или правой половине за $\mathcal{O}(1)$.
3. Запускается рекурсивно, уменьшая область поиска в два раза.

Результат будет получен, когда длина диапазона (далее n) станет равной 1. А значит всего шагов: $\log_2 n$, а значит итоговое время:

$$\log_2 n \cdot \mathcal{O}(1) = \mathcal{O}(\log_2 n) = \mathcal{O}(\log n)$$

Бинарный поиск по ответу

Пусть имеется монотонный предикат $P(n)$, то есть

$$\exists N_0 : \begin{cases} P(n) = 0, & n < N_0 \\ P(n) = 1, & n \geq N_0 \end{cases}$$

И перед нами стоит задача отыскать это N_0. Например, задача проверки наличия элемента X в отсортированном массиве. В данном случае предикат будет звучать как $P(i) = 1 \iff a[i] \geq X$.

Алгоритм

Пусть N_0 заведомо лежит в каком-то диапазоне $[l, r]$. Тогда алгоритм следующий.

1. Если диапазон валиден, то вычислим $P\left(\lceil \frac{l+r}{2} \rceil\right)$. Иначе вернем границу.

Алгоритм

Пусть N_0 заведомо лежит в каком-то диапазоне $[l, r]$. Тогда алгоритм следующий.

1. Если диапазон валиден, то вычислим $P\left(\lceil \frac{l+r}{2} \rceil\right)$. Иначе вернем границу.
2. Если полученный результат истинен, то $N_0 \in \left[l, \lceil \frac{l+r}{2} \rceil\right]$. Запускаем рекурсивно поиск в левой половине массива.

Алгоритм

Пусть N_0 заведомо лежит в каком-то диапазоне $[l, r]$. Тогда алгоритм следующий.

1. Если диапазон валиден, то вычислим $P\left(\lceil \frac{l+r}{2} \rceil\right)$. Иначе вернем границу.
2. Если полученный результат истинен, то $N_0 \in \left[l, \lceil \frac{l+r}{2} \rceil\right]$. Запускаем рекурсивно поиск в левой половине массива.
3. Иначе $N_0 \in \left[\lceil \frac{l+r}{2} \rceil, r\right]$. Запускаем рекурсивно поиск в правой половине массива.

Сложность алгоритма:

Алгоритм

Пусть N_0 заведомо лежит в каком-то диапазоне $[l, r]$. Тогда алгоритм следующий.

1. Если диапазон валиден, то вычислим $P\left(\lceil \frac{l+r}{2} \rceil\right)$. Иначе вернем границу.
2. Если полученный результат истинен, то $N_0 \in \left[l, \lceil \frac{l+r}{2} \rceil\right]$. Запускаем рекурсивно поиск в левой половине массива.
3. Иначе $N_0 \in \left[\lceil \frac{l+r}{2} \rceil, r\right]$. Запускаем рекурсивно поиск в правой половине массива.

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

Вещественный бинарный поиск

Допустим, нужно решить уравнение $x + \tan x = 3$, $x \in \left(0, \frac{\pi}{2}\right)$.

Вещественный бинарный поиск

Допустим, нужно решить уравнение $x + \tan x = 3$, $x \in \left(0, \frac{\pi}{2}\right)$.

Рассмотрим функцию $f(x) = x + \tan x - 3$ и предикат $P(x) = I(f(x) \geq 0)$.

Вещественный бинарный поиск

Допустим, нужно решить уравнение $x + \tan x = 3$, $x \in \left(0, \frac{\pi}{2}\right)$.

Рассмотрим функцию $f(x) = x + \tan x - 3$ и предикат $P(x) = I(f(x) \geq 0)$.

Теперь применим бинарный поиск из алгоритма выше ($l = 0$, $r = \frac{\pi}{2}$), только условие выхода: $|r - l| < \varepsilon$.

Сложность алгоритма:

Вещественный бинарный поиск

Допустим, нужно решить уравнение $x + \tan x = 3$, $x \in \left(0, \frac{\pi}{2}\right)$.

Рассмотрим функцию $f(x) = x + \tan x - 3$ и предикат $P(x) = I(f(x) \geq 0)$.

Теперь применим бинарный поиск из алгоритма выше ($l = 0$, $r = \frac{\pi}{2}$), только условие выхода: $|r - l| < \varepsilon$.

Сложность алгоритма: $\mathcal{O}\left(\log \frac{r-l}{\varepsilon}\right)$.