ДАА - практикум

Александър Станев

- Имейл: **a.g.stanev98@gmail.com**

Михаил Михайлов

- Имейл: **momodei.misho@gmail.com**

Информация за курса

- Материал
- На какво ще се набляга - Ще решаваме задачи, свързани с материала по ДАА, в някакви ограничения (време, памет, ...)
- Език за програмиране - C/C++
- Изисквания за кода - Няма да се държи за стил на кода в този курс

Оценяване и бонуси

- 3 Контролни - общо 250 точки
- Няколко домашни - общо 50 точки
- Бонуси за решилите най-много задачи в арената
- Бонус домашни

Скала за оценяване

- 90т - 104т - 3.00
- 105т - 119т - 3.50
- 120т - 134т - 4.00
- 135т - 149т - 4.50
- 150т - 164т - 5.00
- 165т - 179т - 5.50
- 180т+ - 6.00

Системи за автоматизирано тестване на решения (judges)

- За какво се използват
- Основно ще ползваме арената на ФМИ - https://judge.openfmi.net/
- Различни видове грешки при оценяване

Код	Грешка
OK	Успешно минал тест
WA	Wrong Answer
PE	Presentation Error (whitespace errors, на места се срещат като WA)
TL	Time Limit (Exceeded)
ML	Memory Limit (Exceeded)
RE	Runtime Error (целочислено деление на нула, ровене в чужда памет и т.н.)
CE	Compilation Error (единствения, които идва с пояснение къде ви е грешката)

Регистрации в аренатa на ФМИ

- Нека всички се регистрират в аренатa, като в истинското си име включи и факултетния си номер (за наше улеснение)

- https://judge.openfmi.net/

📈 Демо на аренатa

- Ще започнем с няколко прости задачи, за да се запознаете със системата.
- scanf, printf
- https://judge.openfmi.net/practice/open_contest?contest_id=31

Платформи за състезателно програмиране

- https://codeforces.com/
- https://www.topcoder.com/
- https://www.hackerrank.com/
- http://codeit.bg/
- https://arena.maycamp.com/
- http://action.informatika.bg/

Допълнителни ресурси за подготовка

- Програмиране =++Алгоритми - Преслав Наков, Панайот Добриков
- Introduction to Algorithms (Third Edition) - Cormen, Leiserson, Rivest
- https://cp-algorithms.com/
- https://www.geeksforgeeks.org/
- http://www.informatika.bg/

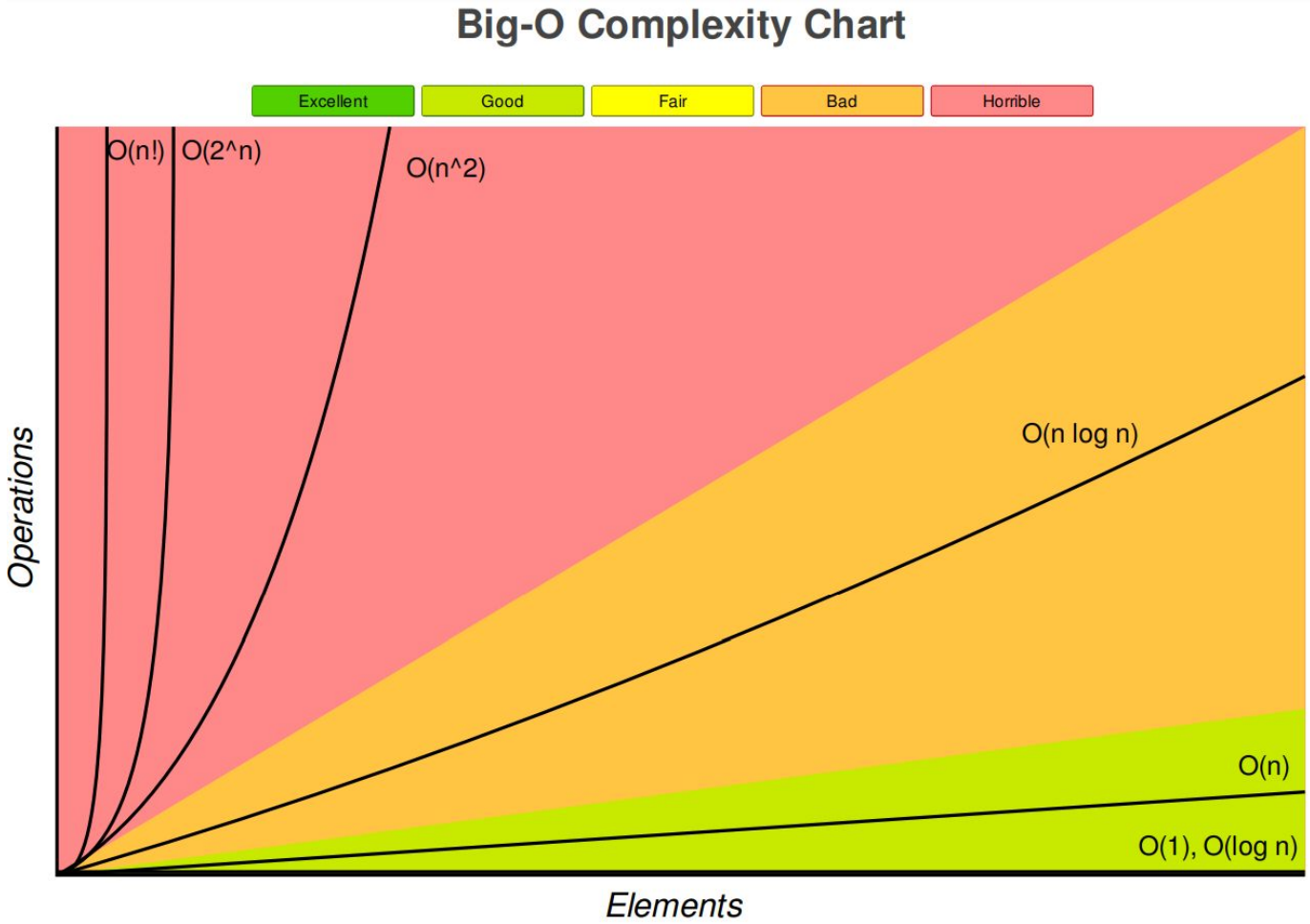
СЛОЖНОСТ НА АЛГОРИТМИ

Big-O нотация

f(n) = O(g(n)) ако $\exists\ n_0 \in \mathbb{N}$ и положителна константа **c**, такава че $f(n) \leq c.g(n)\ \forall\ n \geq n_0$.

Начин за определяне на сложността спрямо Big-O нотацията:

1. Определяне на размера на входа **N**;
2. Изразяване на максималния брой операции които се извършват спрямо **N**;
3. Премахване на всички членове освен този с най-голям степенен показател;
4. Премахване на всички константи и коефициенти;


Big-O Complexity Chart
Excellent
Good
Fair
Bad
Horrible
O(n!)
O(2^n)
O(n^2)
O(n log n)
O(n)
O(1), O(log n)
Operations
Elements

Различни суми

Нека имаме **N** различни цели числа A_1, A_2, ..., A_N ($1 \le A_i <= 10^8$) сортирани в нарстващ ред.

Колко двойки има сред тях, чиято сума е равна на **X**?

Ще тестваме решенията си с 6 различни различни по големина теста:

1) $N = 10^3$
2) $N = 10^4$
3) $N = 10^5$
4) $N = 10^6$
5) $N = 10^7$
6) $N = 10^8$

Решение 1:

Обхождаме числата с два вложени цикъла и проверяваме дали сумата на всяка възможна двойка числа (A_i, A_j) е равна на **X**.

Каква би била сложността на това решение?

```
90  int pairSumNaive() {
91      int res = 0;
92
93      for (int i = 0; i < N; i++) {
94          for (int j = i + 1; j < N; j++) {
95              if (A[i] + A[j] == X) {
96                  res++;
97              }
98          }
99      }
100
101     return res;
102 }
```

Решение 2:

Понеже числата са сортирани можем да се възползваме от това като заместим вложеният цикъл с двоично търсене.

Каква би била сложността на новото решение?

```
4  int pairSumBinSearch() {
5      int res = 0;
6
7      for (int i = 0; i < N; i++) {
8          if (binSearch(i + 1, X - A[i])) {
9              res++;
10         }
11     }
12
13     return res;
14 }
```

А сложността на функцията за двоично търсене?

```
4  bool binSearch(int from, int x) {
5      int left = from;
6      int right = N - 1;
7
8      while (left <= right) {
9          int mid = (left + right) >> 1;
10
11         if (A[mid] == x) {
12             return true;
13         }
14
15         if (A[mid] < x) {
16             left = mid + 1;
17         }
18         else {
19             right = mid - 1;
20         }
21     }
22
23     return false;
24 }
```

Решение 3:

Можем да подобрим дори това решение с помощта на някои наблюдения. Можем да забележим, че за всяка двойка индекси (**i**, **j**), **i** < **j** са верни следните неравенства:

$$A_i + A_j < A_{i+1} + A_j$$
$$A_i + A_j > A_i + A_{j-1}$$

Следователно можем да започнем с **i** = 0 и **j** = **N** - 1.

Ако $A_i + A_j < X$ увеличаваме **i**, в противен случай увеличаваме **j**. Процедурата продължава докато **i** < **j**.

Каква би била сложността на това решение?

```
90  int pairSumFast() {
91      int res = 0;
92
93      int i = 0;
94      int j = N - 1;
95
96      while (i < j) {
97          while (i < j && A[i] + A[j] > X) {
98              j--;
99          }
100
101         if (i < j && A[i] + A[j] == X) {
102             res++;
103         }
104
105         i++;
106     }
107
108     return res;
109 }
```

Резултати от 3те решения

Размер на Входа	10^3	10^4	10^5	10^6	10^7	10^8
Naive	**0.000s**	**0.024s**	**2.376s**	**237.700s**	**6.6h**	**27d**
BinSearch	**0.000s**	**0.001s**	**0.002s**	**0.026s**	**0.278s**	**2.998s**
Fast	**0.000s**	**0.000s**	**0.001s**	**0.004s**	**0.035s**	**0.082s**


THE
END