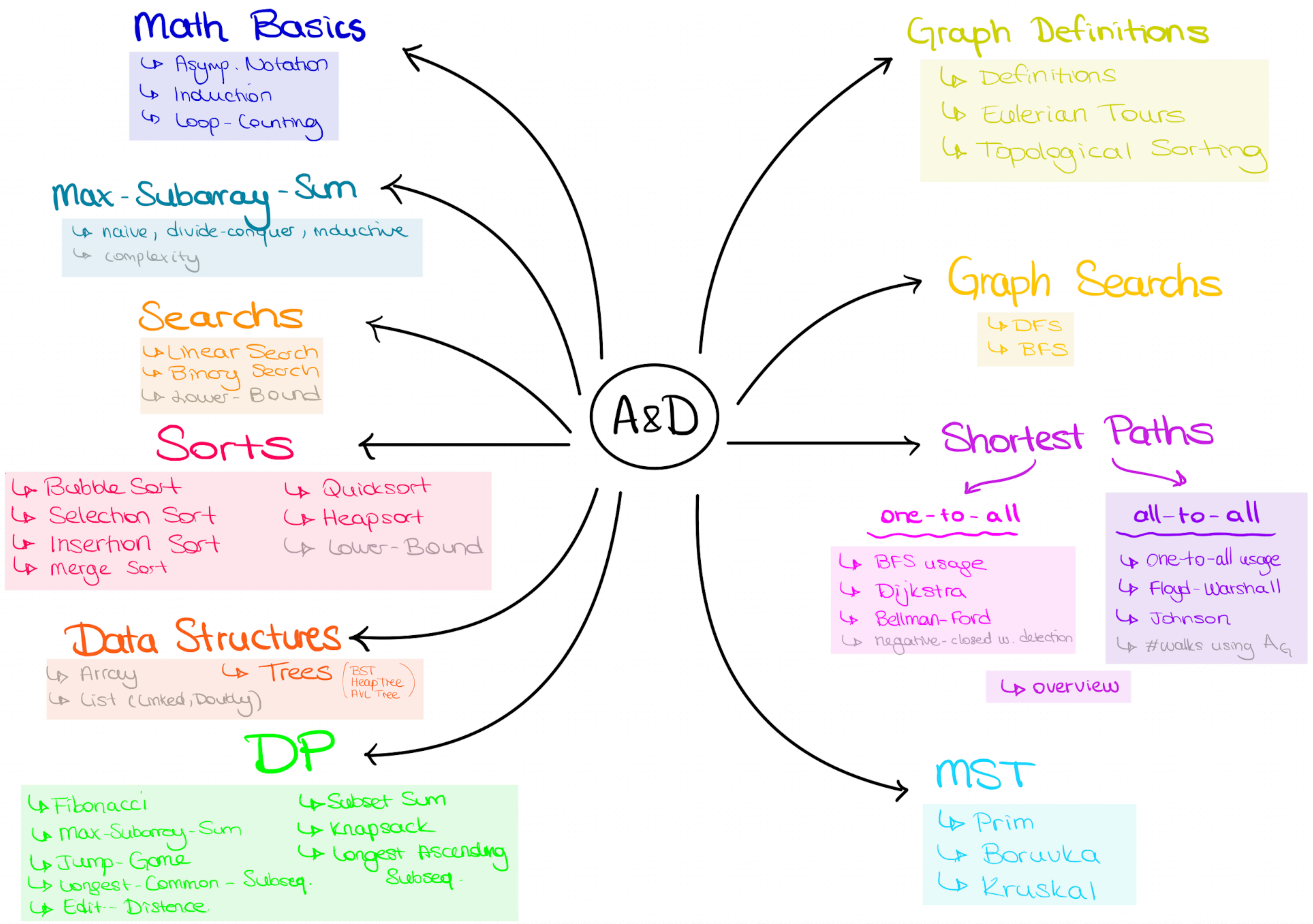
A&D

Exercise Session 3

Nil Ozer

A&D Overview



A&D
Math Basics
Asymp. Notation
Induction
Loop-Counting
Max-Subarray-Sum
naive, divide-conquer, inductive
complexity
Searchs
Linear Search
Binary Search
Lower-Bound
Sorts
Bubble Sort
Selection Sort
Insertion Sort
Merge Sort
Quicksort
Heapsort
Lower-Bound
Data Structures
Array
List (Linked, Doubly)
Trees
BST
Heap Tree
AVL Tree
DP
Fibonacci
Max-Subarray-Sum
Jump-Game
Longest-Common-Subseq.
Edit-Distance
Subset Sum
Knapsack
Longest Ascending Subseq.
Graph Definitions
Definitions
Eulerian Tours
Topological Sorting
Graph Searchs
DFS
BFS
Shortest Paths
one-to-all
BFS usage
Dijkstra
Bellman-Ford
negative-closed w. detection
all-to-all
One-to-all usage
Floyd-Warshall
Johnson
#walks using A_G
overview
MST
Prim
Boruvka
Kruskal

Outline

- Quick recap
- Quiz

- Exercise Sheet 1 Bonus Feedback
- Asymptotic Notation Kahoot

- Loop Counting

- Exercise Sheet 2 - non Bonus
- Mini-exam discussion
- Code Expert Introduction

Quick Recap

Mini cheat-sheet

$\lim_{n \to \infty}$: $1 < \log(\log(n)) < \log(n) < \sqrt{n} < n < n \cdot \log(n) < n \cdot \sqrt{n} < n^2 < 2^n < n! < n^n$

$n^x < x^n$ (x being fixed)

Sums

$$\sum_{i=1}^{n} i = 1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2}$$

$$\sum_{i=1}^{n} i^2 = 1^2 + 2^2 + 3^2 + \cdots n^2 = \frac{n(n+1)(2n+1)}{6}$$

Geometric series : $$\sum_{k=0}^{n} q^k = \frac{q^{n+1} - 1}{q - 1}$$

$$\sum_{k=0}^{3} 3^k = 3^0 + 3^1 + 3^2 + 3^3 = \frac{3^4 - 1}{3 - 1} = 40$$

Factorial

$$\frac{n}{2}^{\frac{n}{2}} \leq n! \leq n^n$$

From Exercise Sheet 1 :

$\sum_{i=1}^{n} i^k \leq n^{k+1}$

$\sum_{i=1}^{n} i^k \geq \frac{1}{2^{k+1}} \cdot n^{k+1}$

$$\sum_{i=1}^{n} i^k = \Theta(n^{k+1})$$

Quick Recap
Definitions

Definition 1 (O-Notation). For $f : N \to \mathbb{R}^+$,

$$O(f) := \{g : N \to \mathbb{R}^+ \mid \exists C > 0\, \forall n \in N\; g(n) \le C \cdot f(n)\}.$$

Definition 1 (Ω-Notation). For $f : N \to \mathbb{R}^+$,

$$\Omega(f) := \{g : N \to \mathbb{R}^+ \mid f \le O(g)\}.$$

We write $g \ge \Omega(f)$ instead of $g \in \Omega(f)$.

Definition 2 (Θ-Notation). For $f : N \to \mathbb{R}^+$,

$$\Theta(f) := \{g : N \to \mathbb{R}^+ \mid g \le O(f) \text{ and } f \le O(g)\}.$$

We write $g = \Theta(f)$ instead of $g \in \Theta(f)$.

In other words, for two functions $f, g : N \to \mathbb{R}^+$ we have

$$g \ge \Omega(f) \Leftrightarrow f \le O(g)$$

and

$$g = \Theta(f) \Leftrightarrow g \le O(f) \text{ and } f \le O(g).$$

Theorem 1 (Theorem 1.1 from the script). *Let N be an infinite subset of $\mathbb{N}$ and $f : N \to \mathbb{R}^+$ and $g : N \to \mathbb{R}^+$.*

- *If $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$, then $f \le O(g)$, but $f \ne \Theta(g)$.*
- *If $\lim_{n\to\infty} \frac{f(n)}{g(n)} = C \in \mathbb{R}^+$, then $f = \Theta(g)$.*
- *If $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$, then $f \ge \Omega(g)$, but $f \ne \Theta(g)$.*

Quiz

select one or more vs. select one

Log notation clarified

- From now on we use :
  - log for $\log_2(x)$
  - ln for $\log_e(x)$

- Will be in the notation sheet

Exercise Sheet 1

Bonus Feedback

Keep up the good work !!

- Watch out for the base case (n>=0 -> n=0)
- Try to have all of the intermediate steps

- Don't rush to the solution, until you've gained the intuition by solving the task slowly!
- Log notation !

$$\log(m^3) = \log(m \cdot m \cdot m)$$

$$\log(m)^3 = \log m \cdot \log m \cdot \log m = \log^3(m)$$

* work with log computation: $\frac{\log(m^3)}{\log(m)^3} = \frac{3 \log m}{\log(m)^3} = \frac{3}{\log(m)^2} \xrightarrow{n \to \infty} 0$

Asymptotic Notation Kahoot

Loop Counting

Loop Counting

Task Description

a) *Counting iterations:* For the following code snippets, derive an asymptotic bound for the number of times f is called. Simplify the expression as much as possible and state it in Θ-notation as concisely as possible.

i) Snippet 1:

Algorithm 1

```
for j = 1, ..., n^2 do
    for k = 1, ..., j do
        f()
    f()
```

ii) Snippet 2:

Algorithm 2

```
for j = 1, ..., n do
    k ← 1
    while k ≤ j^2 − 1 do
        ℓ ← 1
        while ℓ ≤ n do
            f()
            ℓ ← 2ℓ
        k ← k + 1
```

Your solution consist of :

1. Exact number of times f is called
2. Maximal simplification of the expression in θ-notation

Loop Counting

Learn with an example !

Exercise 3.3 *Counting function calls in loops* **(1 point).**

For each of the following code snippets, compute the number of calls to f as a function of $n \in \mathbb{N}$. Provide **both** the exact number of calls and a maximally simplified asymptotic bound in Θ notation.

(a)

Algorithm 1

$i \leftarrow 0$
while $i \leq n$ **do**
  $f()$
  $f()$
  $i \leftarrow i + 1$
$j \leftarrow 0$
while $j \leq 2n$ **do**
  $f()$
  $j \leftarrow j + 1$

(b)

Algorithm 2

$i \leftarrow 1$
while $i \leq n$ **do**
  $j \leftarrow 1$
  while $j \leq i^3$ **do**
    $f()$
    $j \leftarrow j + 1$
  $i \leftarrow i + 1$

Loop Counting

Theorem

Master theorem. The following theorem is very useful for running-time analysis of divide-and-conquer algorithms.

Theorem 1 (master theorem). *Let $a, C > 0$ and $b \geq 0$ be constants and $T : \mathbb{N} \to \mathbb{R}^+$ a function such that for all even $n \in \mathbb{N}$,*

$$T(n) \leq aT(n/2) + Cn^b. \quad (1)$$

Then for all $n = 2^k$, $k \in \mathbb{N}$,

- *If $b > \log_2 a$, $T(n) \leq O(n^b)$.*
- *If $b = \log_2 a$, $T(n) \leq O(n^{\log_2 a} \cdot \log n)$.*[1]
- *If $b < \log_2 a$, $T(n) \leq O(n^{\log_2 a})$.*

If the function T is increasing, then the condition $n = 2^k$ can be dropped. If (1) holds with "=", then we may replace O with Θ in the conclusion.

This generalizes some results that you have already seen in this course. For example, the (worst-case) running time of Karatsuba's algorithm satisfies $T(n) \leq 3T(n/2) + 100n$, so we have $a = 3$ and $b = 1 < \log_2 3$, hence $T(n) \leq O(n^{\log_2 3})$. Another example is binary search: its running time satisfies $T(n) \leq T(n/2) + 100$, so $a = 1$ and $b = 0 = \log_2 1$, hence $T(n) \leq O(\log n)$.

Either won't be used, or will be written in the task description

Let’s take a break

Loop Counting

Exam question

FS23

Theory Task T2. / 15 P

In this part, you should **justify your answers briefly**.

/ 4 P

a) *Counting iterations:* For the following code snippets, derive an asymptotic bound for the number of times f is called. Simplify the expression as much as possible and state it in Θ-notation as concisely as possible.

i) Snippet 1:

Algorithm 1

```
for i = 1, ..., n do
    for j = 1, ..., i^2 do
        f()
    f()
```

ii) Snippet 2:

Algorithm 2

```
for i = 1, ..., n do
    k ← 1
    while k ≤ i^2 do
        f()
        k ← 2k
    f()
```

Loop Counting

Exam Tipps

- T2 task
- 2 snippets
  - First one easier second one harder
- If it's needed, the master theorem will be there

- Show your work !
- Solve it, whenever it appears !

- It's relatively easy once you've practiced it !

Exercise Sheet 2

(Non Bonus)

Gruppen	11
Abgegeben	11

Mini Exam Discussion

Induction + Asymptotic Notation

Code Expert Introduction

Questions

Feedbacks , Recommendations

Nil Ozer