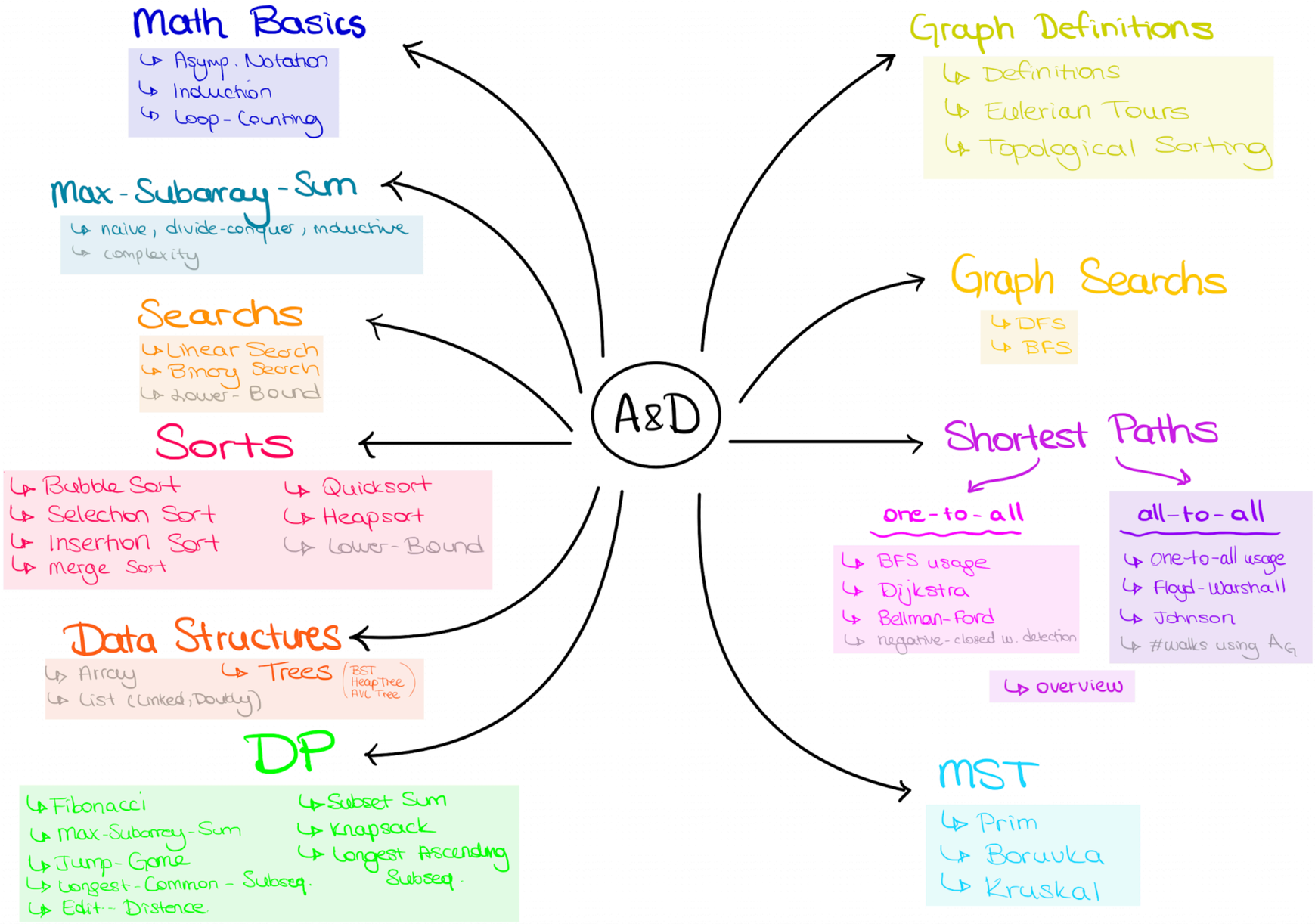
A&D

Exercise Session 7

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, HeapTree, 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

- Quiz
- Exercise Sheets

- DP I - Edit Distance
- DP II

- DP Mock Exam

- Next week

Quiz

Exercise Sheet 5

Bonus Feedback

- 5.1
  - “Attention mistakes” won’t be tolerated in exam
- 5.3
  - Don’t forget to refer to the pseudocode !
  - Is it O or Θ !!
- 5.4
  - Tree Proofs structure

- Feel free to correct me !

Tree Proofs

General Structure

- Base Case
  - Usually leaves !
- I.H.
  - Assume the property for some n
- I.S.
  - Show what happens to n in one iteration (assuming I.H.) , describe briefly
  - How does the recursion/iteration end ?
    - arriving to the leaf , root
    - fulfilling an if condition

Exercise Sheets

- Exercise Sheet 4 left for next time, again :(

- Exercise Sheet 6 peergrading
  - 6.1 this week
  - Emails are sent

- New groups for Exercise Sheet 7 !

DP I

Edit Distance

Problem : Given two strings A and B, find the minimum number of edits (operations) to convert A into B

Operations : Replace : Replace a character at any index of A with some other character

Insert : Insert any character into A

Remove : Remove a character of A

Examples :	Inputs :	Output :	Operations :
	“cat" and “cut”	1	replace a with u
	“sunday” and “saturday”	3	convert un to atur : replace n by r insert a, insert t

Edit Distance

Operations: Replace : Replace a character at any index of A with some other character

Insert : Insert any character after or before any index of A

Remove : Remove a character of A

Idea : For every element of A , 3 things can happen

- will be deleted
- B[j] gets inserted after
- will be replaced to match B[j]

Definition of the DP table : DP[i][j] = ED of A[0..i] and B[0..j] DP[0...n][0...m]

the minimum number of edits to convert A[0..i] into B[0..j]

Computation of an entry :

Initialization : DP[i][0] = i DP[0][j] = j

Recursion :

$$DP[i][j] = \begin{cases} DP[\,i-1\,][\,j-1\,] & \text{if } A[i] == B[j] \\ 1 + \min\{\, DP[\,i-1\,][\,j\,],\ DP[\,i\,][\,j-1\,],\ DP[\,i-1\,][\,j-1\,] \,\} & \text{else} \end{cases}$$

add B[j] to the end

delete A[i]

replace A[i] with B[j]

Extracting the solution : The solution is at DP[n][m]

Subset Sum

Problem : Given an array A , check if there's a subset of A s.t. it's sum is equal to a given number b.

Return true if we find I else false $I \subseteq \{1 \dots n\}$ s.t. $\sum_{i \in I} A[i] = b$

Examples :	Inputs :	Output :	what's used
	[1,2,3,4,5] , 1000	False	
	[1,2,3,4,5] , 10	True	2,3,5 or 1,2,3,4
	[] , 0	True	

Subset Sum

Idea : Two things can happen to each element

- It gets used in I
- It doesn't get used in I

DP[0...n][0...S]

Definition of the DP table : DP[i][s] = "Can I find a subset sum from A[0...i] that's equal to s "

Computation of an entry :

Initialization : DP[0][0] = True DP[i][0] = True DP[0][s] = False

Recursion :

we don't use i in I we use i in I

DP[i][s] = DP[i-1][s] || DP[i-1][s-A[i]]

Extracting the solution : The solution is at DP[n][S]

Knapsack

Problem :

Given : $\mathcal{W}$: weight limit

w_i : weight of each item

p_i : profit of each item

Searched : Maximum profit that one can have

Examples :	Inputs :	Output :	Explanation :
$\mathcal{W} = 0$ $p = [1, 2, 3]$ $w = [5, 5, 5]$		0	all items are above weight limit
$\mathcal{W} = 5$ $p = [1, 2, 3]$ $w = [5, 5, 5]$		3	we can only pick one item, and we pick the most profitable
$\mathcal{W} = 10$ $p = [1, 2, 3]$ $w = [5, 5, 5]$		5	we can pick two items, and we pick 2+3 = 5

Knapsack

Idea : Two things can happen to each element

- We use it and get profit
- We don’t use it

DP[0...n][0...W]

Definition of the DP table : DP[i][w] = “Maximum profit from A[0..i] with weight limit w“

Computation of an entry :

Initialization : DP[i][0] = 0

Recursion :

we don’t use i in I we use i in I

DP[i][w] = DP[i-1][w] || p[i] + DP[i-1][w- w[i]]

Extracting the solution : The solution is at DP[n][W]

Let’s take a break

Longest Increasing Subsequence

Problem : Given array A find the length of the Longest Increasing Subsequence (LIS)

LIS

The longest possible subsequence in which the elements of the subsequence are sorted in increasing order.

Subsequence

A subsequence is a sequence generated from the original array by deleting 0 or more elements without changing the relative order of the remaining elements.

A [1, 3, 5, 7]

Is it a subsequence ?

[1, 5, 7]

[3, 7]

[1, 7, 5]

[ ]

Longest Increasing Subsequence

Problem : Given array A find the length of the Longest Increasing Subsequence (LIS)

LIS

The longest possible subsequence in which the elements of the subsequence are sorted in increasing order.

Subsequence

A subsequence is a sequence generated from the original array by deleting 0 or more elements without changing the relative order of the remaining elements.

Examples :	Input :	Output :	LIS:
	[10, 9, 2, 5, 3, 7, 101, 18]	4	[2, 3, 7, 18]
	[3, 2, 1]	1	[3] , [2] or [1]

Longest Increasing Subsequence

Idea : We need to mark the smallest ending !

DP[0...n-1][1...n]

Definition of the DP table :

DP[i][l] = "smallest ending of an increasing subsequence of length l in A[0...i]"

∞ if no such incresing subsequence exists

Computation of an entry :

Initialization : DP[0][1] = A[0] DP[0][l] = ∞ for l > 1

Recursion :

A[i] fits the element coming before by being bigger than it (it should be increasing)

A[i] improves the current smallest ending of length i by being smaller

$$DP[i][l] = \begin{cases} A[i] & \text{if } DP[i-1][l-1] < A[i] \text{ and } A[i] < DP[i-1][l] \\ DP[i-1][l] & \text{else} \end{cases}$$

Extracting the solution : The solution is found by backtracking

DP

Exam Question

Theory Task T3. / 9 P

You are given an array of n natural numbers $a_1, \ldots, a_n \in \mathbb{N}$, and two natural numbers $A, B \in \mathbb{N}$. You want to determine whether there is a subset $I \subseteq \{1, \ldots, n\}$ satisfying

$$\sum_{i \in I} a_i = A \quad \text{and} \quad \sum_{i \in I} a_i^2 = B.$$

For example,

- The answer for the input $(a_i)_{i \leq n} = [2, 4, 8, 1, 4, 5, 3]$, $A = 8$ and $B = 30$ is *yes* because the set of indices $I = \{1, 4, 6\}$, which corresponds to $(a_i)_{i \leq I} = [2, 1, 5]$, yields the *sum* $2 + 1 + 5 = 8$ and the *sum-of-squares* $2^2 + 1^2 + 5^2 = 30$.
- The answer for the input $(a_i)_{i \leq n} = [2, 4, 8, 1]$, $A = 6$ and $B = 15$ is *no*.

Provide a *dynamic programming* algorithm that determines whether such a subset I exists. In order to get full points, your algorithm should have an $O(n \cdot A \cdot B)$ runtime. Address the following aspects in your solution:

DP

How to learn

- Theory, written tasks :
  - Exam questions T3 !!
  - Exercise sheets
  - geeksforgeeks

Always a combination of the ideas dicussed in lecture !

- Coding :
  - CodeEx exercises , my videos
  - Old Exam exercises
  - Leetcode https://leetcode.com/studyplan/dynamic-programming/

Table ? 🤦‍♀️

DP

Exam Tipps

- Get a hint from the running time
  - Doesn’t always work!
- Have an order for yourself

Always a combination of the ideas dicussed in lecture !

- The definition of an entry should be very clear to you, at all times !

- Initialization : What should the entry be in base cases (ex : A = [ ])
- Recursion : How can you use the previous entries to get the current entry
  - This is the only question that you’re answering !!

Done with DP !

DP Mini Exam (lol)

Next Week

Questions

Feedbacks , Recommendations

Nil Ozer