Datenstrukturen und Algorithmen

Exercise 7

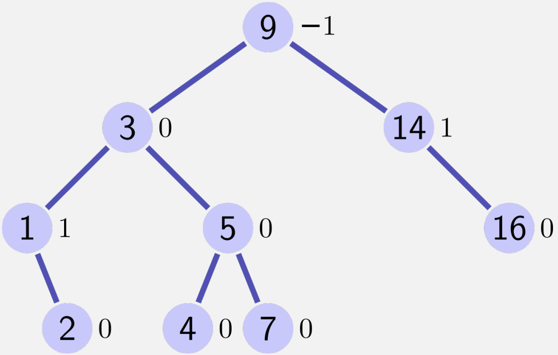
FS 2019

Program of today

1. Feedback of last exercise(s)
2. Repetition theory

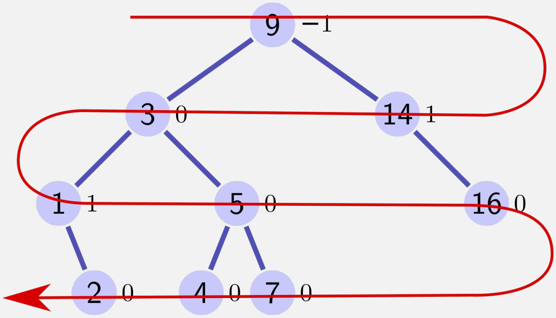
AVL insertion

- Given an AVL tree, is there an order that produces the same tree and does not cause any rotations



AVL insertion

- Given an AVL tree, is there an order that produces the same tree and does not cause any rotations



AVL insertion - sketch of proof

- Any sequence that keeps the height order intact is fine
- Proof?
- By induction over the height of the tree.

AVL insertion - sketch of proof

- Any sequence that keeps the height order intact is fine
- Proof?
- By induction over the height of the tree.
- Hypothesis: Keys at height h and lower are placed in the same place and do not cause insertion.

AVL insertion - sketch of proof

- Any sequence that keeps the height order intact is fine
- Proof?
- By induction over the height of the tree.
- Hypothesis: Keys at height h and lower are placed in the same place and do not cause insertion.
- Step: Show that the traversal is the same as in the original tree, yields same position. Use AVL property of T to show that there will not be a height difference bigger than 1, and therefore no rotation.

2. Repetition theory

Dynamic programming

A complete description of a dynamic program **always** consists of the following aspects:

Dynamic programming

A complete description of a dynamic program **always** consists of the following aspects:

- **Definition of the DP table**:

Dynamic programming

A complete description of a dynamic program **always** consists of the following aspects:

- **Definition of the DP table**: What are the dimensions of the table? What is the meaning of each entry?

Dynamic programming

A complete description of a dynamic program **always** consists of the following aspects:

- **Definition of the DP table**: What are the dimensions of the table? What is the meaning of each entry?
- **Computation of an entry**:

Dynamic programming

A complete description of a dynamic program **always** consists of the following aspects:

- **Definition of the DP table**: What are the dimensions of the table? What is the meaning of each entry?
- **Computation of an entry**: How can an entry be computed from the values of other entries? Which entries do not depend on others?

Dynamic programming

A complete description of a dynamic program **always** consists of the following aspects:

- **Definition of the DP table**: What are the dimensions of the table? What is the meaning of each entry?
- **Computation of an entry**: How can an entry be computed from the values of other entries? Which entries do not depend on others?
- **Calculation order**:

Dynamic programming

A complete description of a dynamic program **always** consists of the following aspects:

- **Definition of the DP table**: What are the dimensions of the table? What is the meaning of each entry?
- **Computation of an entry**: How can an entry be computed from the values of other entries? Which entries do not depend on others?
- **Calculation order**: In which order can entries be computed so that values needed for each entry have been determined in previous steps?

Dynamic programming

A complete description of a dynamic program **always** consists of the following aspects:

- **Definition of the DP table**: What are the dimensions of the table? What is the meaning of each entry?
- **Computation of an entry**: How can an entry be computed from the values of other entries? Which entries do not depend on others?
- **Calculation order**: In which order can entries be computed so that values needed for each entry have been determined in previous steps?
- **Extracting the solution**:

Dynamic programming

A complete description of a dynamic program **always** consists of the following aspects:

- **Definition of the DP table**: What are the dimensions of the table? What is the meaning of each entry?
- **Computation of an entry**: How can an entry be computed from the values of other entries? Which entries do not depend on others?
- **Calculation order**: In which order can entries be computed so that values needed for each entry have been determined in previous steps?
- **Extracting the solution**: How can the final solution be extracted once the table has been filled?

Dynamic programming

A complete description of a dynamic program **always** consists of the following aspects:

- **Definition of the DP table**: What are the dimensions of the table? What is the meaning of each entry?
- **Computation of an entry**: How can an entry be computed from the values of other entries? Which entries do not depend on others?
- **Calculation order**: In which order can entries be computed so that values needed for each entry have been determined in previous steps?
- **Extracting the solution**: How can the final solution be extracted once the table has been filled?

Longest ascending Sequence in matrix

Given $n \times m$ matrix A:

9	27	42	41	48
35	39	8	3	5
12	49	2	38	4
15	47	29	28	6
19	1	25	33	10

Longest ascending Sequence in matrix

Given $n \times m$ matrix A:

9	27	42	41	48
35	39	8	3	5
12	49	2	38	4
15	47	29	28	6
19	1	25	33	10

Wanted longest ascending sequence:

$$4, 6, 28, 29, 47, 49$$

Definition of the DP table

- What are the dimensions of the table?

Definition of the DP table

- What are the dimensions of the table?
  - $n \times m$

Definition of the DP table

- What are the dimensions of the table?
  - $n \times m(\times 2)$

Definition of the DP table

- What are the dimensions of the table?
  - $n \times m(\times 2)$
- What is the meaning of each entry?

Definition of the DP table

- What are the dimensions of the table?
  - $n \times m (\times 2)$
- What is the meaning of each entry?
  - In $T[x][y]$ is the length of the longest ascending sequence that ends in $A[x][y]$
  - In $S[x][y]$ are the coordinates of the predecessor in ascending sequence (if exists)

Computation of an entry

- How can an entry be computed from the values of other entries? Which entries do not depend on others?

Computation of an entry

- How can an entry be computed from the values of other entries? Which entries do not depend on others?
  - Consider neighbors with smaller entry in A

Computation of an entry

- How can an entry be computed from the values of other entries? Which entries do not depend on others?
  - Consider neighbors with smaller entry in A
  - From the smaller entries choose entry with the largest entry in T

Computation of an entry

- How can an entry be computed from the values of other entries? Which entries do not depend on others?
  - Consider neighbors with smaller entry in A
  - From the smaller entries choose entry with the largest entry in T
  - Update T and S. (S gets coordinate from selected neighbor, T gets value from selected neighbor increased by one)

Computation of an entry

- How can an entry be computed from the values of other entries? Which entries do not depend on others?
  - Consider neighbors with smaller entry in A
  - From the smaller entries choose entry with the largest entry in T
  - Update T and S. (S gets coordinate from selected neighbor, T gets value from selected neighbor increased by one)

Calculation order

- In which order can entries be computed so that values needed for each entry have been determined in previous steps?

Calculation order

- In which order can entries be computed so that values needed for each entry have been determined in previous steps?
- Start with smallest element in A and so on. (Means that one has to sort A)

Calculation order

- In which order can entries be computed so that values needed for each entry have been determined in previous steps?

- Start with smallest element in A and so on. (Means that one has to sort A)

- Arbitrary order, if entry is already computed skip it otherwise compute for smaller neighbor recursively.

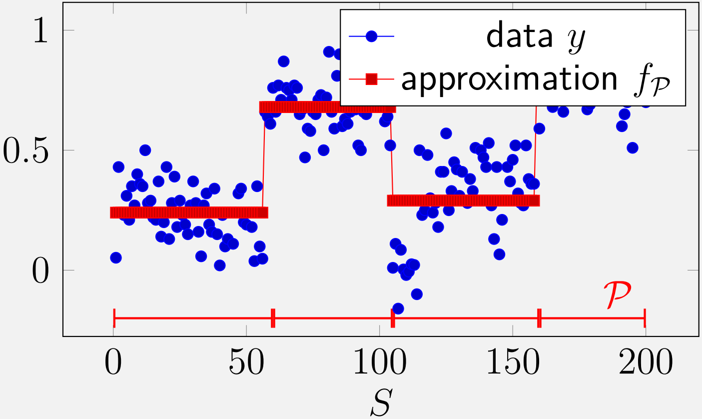
Extracting the solution

- How can the final solution be extracted once the table has been filled?

Extracting the solution

- How can the final solution be extracted once the table has been filled?
  - Consider all entries to find one with a longest sequence. From there, we can reconstruct the solution by following the corresponding predecessors.

Piecewise Constant Approximation



Piecewise Constant Approximation

$$H_{\gamma,y} : \mathcal{P} \mapsto \gamma|\mathcal{P}| + \sum_{I \in \mathcal{P}} \sum_{i \in I} (y_i - \mu_I)^2$$

Piecewise Constant Approximation

$$H_{\gamma,y} : \mathcal{P} \mapsto \gamma|\mathcal{P}| + \sum_{I \in \mathcal{P}} \sum_{i \in I} (y_i - \mu_I)^2$$

- $\mathcal{P}$: (set of intervals I_i, such that $\cup_i I_i = S$).
- **Goal:** find the partition $\hat{\mathcal{P}}$ such that $H_{\gamma,y}(\hat{\mathcal{P}})$ is minimal
- Utilize: efficient computation of the mean using prefix sums (exercise 1): $\mu_I = \frac{1}{|I|} \sum_{i \in I} y_i$

Piecewise Constant Approximation

$$H_{\gamma,y} : \mathcal{P} \mapsto \gamma|\mathcal{P}| + \sum_{I \in \mathcal{P}} \sum_{i \in I} (y_i - \mu_I)^2$$

- $\mathcal{P}$: (set of intervals I_i, such that $\cup_i I_i = S$).
- **Goal:** find the partition $\hat{\mathcal{P}}$ such that $H_{\gamma,y}(\hat{\mathcal{P}})$ is minimal
- Utilize: efficient computation of the mean using prefix sums (exercise 1): $\mu_I = \frac{1}{|I|} \sum_{i \in I} y_i$
- Utilize: Efficient computation of $e_{[l,r)} = \sum_{i=l}^{r-1} (y_i - \mu_{[l,r)})^2$

Piecewise Constant Approximation

$$H_{\gamma,y} : \mathcal{P} \mapsto \gamma|\mathcal{P}| + \sum_{I \in \mathcal{P}} \sum_{i \in I} (y_i - \mu_I)^2$$

- $\mathcal{P}$: (set of intervals I_i, such that $\cup_i I_i = S$).
- **Goal:** find the partition $\hat{\mathcal{P}}$ such that $H_{\gamma,y}(\hat{\mathcal{P}})$ is minimal
- Utilize: efficient computation of the mean using prefix sums (exercise 1): $\mu_I = \frac{1}{|I|} \sum_{i \in I} y_i$
- Utilize: Efficient computation of $e_{[l,r)} = \sum_{i=l}^{r-1} (y_i - \mu_{[l,r)})^2$

Piecewise Constant Approximation

$$H_{\gamma,y} : \mathcal{P} \mapsto \gamma|\mathcal{P}| + \sum_{I\in\mathcal{P}} \sum_{i\in I} (y_i - \mu_I)^2$$

- **Goal:** find the partition $\hat{\mathcal{P}}$ such that $H_{\gamma,y}(\hat{\mathcal{P}})$ is minimal
- **Dynamic programming**: definition of the table, computation of an entry, calculation order, extracting solution
- Utilize*: $H_{\gamma,y}(\mathcal{P} \cup \{[l,r)\}) = H_{\gamma,y}(\mathcal{P}) + \gamma + e_{[l,r)}$

*Assumption: $\mathcal{P} \cup \{[l,r)\}$ is a partition

Questions?