Informatik II

Übung 12

FS 2020

Program Today

1. Repetition theory

2. In-Class Exercise

1. Repetition theory

Dynamic Programming: Idea

- Divide a complex problem into a reasonable number of sub-problems
- The solution of the sub-problems will be used to solve the more complex problem
- Identical problems will be computed only once

Dynamic Programming = Divide-And-Conquer ?

- In both cases the original problem can be solved (more easily) by utilizing the solutions of sub-problems. The problem provides *optimal substructure*.
- Divide-And-Conquer algorithms (such as Mergesort): sub-problems are independent; their solutions are required only once in the algorithm.
- DP: sub-problems are dependent. The problem is said to have *overlapping sub-problems* that are required multiple-times in the algorithm.
- In order to avoid redundant computations, results are tabulated. For *sub-problems there must not be any circular dependencies*.

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?

2. In-Class Exercise

Longest Ascending Sequence in a Matrix

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?

- Bottom-Up: 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?

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

- Recursively: 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.

Task

Implement a DP solution in the prepared CodeExpert program.

Questions / Suggestions?