Algorithmen & Datenstrukturen

Woche 9

Marius Tomek, Nicolas Wehrli, Tim Rieder

21. November 2022

ETH Zürich

Inhalt

Kurze Kommentare zur letzten Serie

Serie 07

Häufige Fehler:

- DP nicht vollständig beschrieben
  Tipp: 6 Punkte einprägen und verwenden
- 7.3: Aufgabenbeschreibung falsch verstanden
- Korrektheit der Lösung begründen.
  Die Rekursion soll begründet sein, sonst ist die Lösung oft nicht nachvollziehbar

Graphen

Graphen - Terminologie

- Kante (Edge)
- Knoten (Vertex)
- Weg (Walk)
- Pfad (Path)
- Eulerweg (Eulerian Walk)
- Eulerkreis (Eulerian Circuit)
- Hamiltonkreis (Hamilton Cycle)

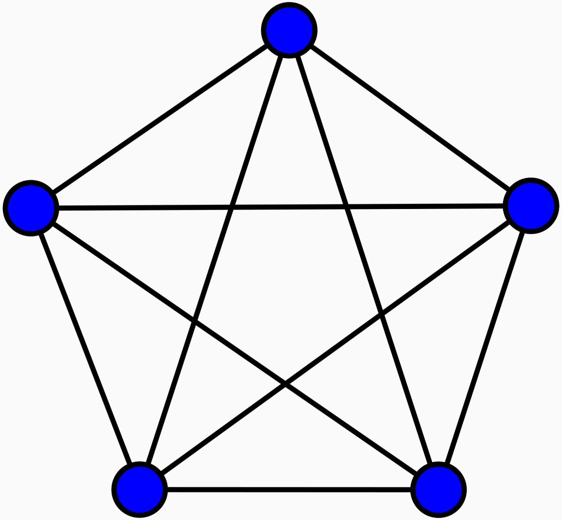
Graphen - Terminologie 2

- Weg (Walk)
  Eine Reihe von verbundenen Knoten
- Pfad (Path)
  Eine Reihe paarweise unterschiedlicher verbundener Knoten
  Es gelte $\forall i, j (i \neq j) \implies (v_i \neq v_j)$
- Eulerweg (Eulerian Walk)
  Ein Weg welcher alle Kanten im Graph genau ein mal durchläuft
- Eulerkreis (Eulerian Circuit)
  Ein Zyklus welcher alle Kanten im Graph genau ein mal durchläuft
  Es gelte $v_0 = v_{end}$
- Hamiltonkreis (Hamilton Cycle)
  Ein Kreis welcher alle Knoten im Graph genau ein mal durchläuft
  Es gelte ebenfalls $v_0 = v_{end}$

Graphen - Visualisiert

- Hat der folgende Graph einen Eulerpfad?
- Hat der folgende Graph einen Eulerkreis?
- Hat der folgende Graph einen Hamiltonkreis?

Graphen - Visualisiert 2



Eulerweg und Eulerkreis

- Eulerweg
  $\#$ Knoten v mit $deg(v) \leq 2$
  Für alle Knoten ausser Start- und Endknoten gelte:
  $deg(v) \equiv_2 0$
- Eulerkreis
  $\forall v \; deg(v) \equiv_2 0$

Homework 08

Exercise 8.2 *AVL trees* **(1 point)**.

(a) Draw the tree obtained by inserting the keys 1, 6, 8, 0, 3, 2, 9 in this order into an initially empty AVL tree. Give also the intermediate states before and after each rotation that is performed during the process.

(b) Delete 0, 2, and 1 in this tree, and afterwards delete key 6 in the resulting tree. Give also the intermediate states before and after each rotation is performed during the process.

Exercise 8.5

Exercise 8.5 *Computing with a stack* **(2 points)**.

In many programming languages, e.g., in Python, stacks are commonly used for evaluating arithmetic expressions. Evaluating expressions usually happens in two steps. First, values are loaded into the stack. Then, operations are applied stepwise on the top elements in order to obtain the desired value.

Exercise 8.5

In this exercise, we focus on the second phase, and consider the following three basic operations used to compute with stacks:

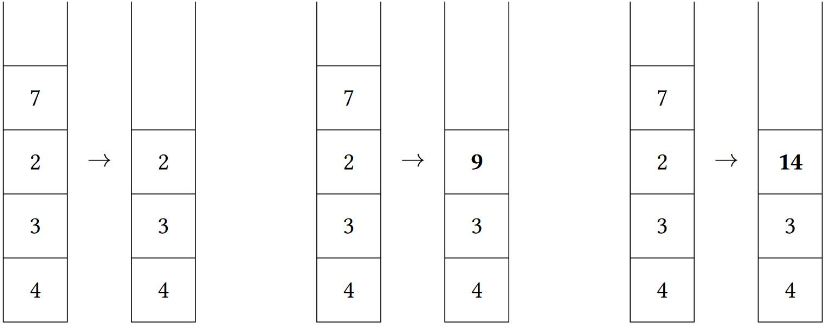
POP: If there is at least one element in the stack, remove the top element of the stack. Otherwise, do nothing.

ADD: If there are at least two elements in the stack, remove the top two elements, compute their sum, and push this sum back into the stack. If there is less than two elements in the stack, do nothing.

MUL: If there are at least two elements in the stack, remove the top two elements, compute their product, and push this product back into the stack. If there is less than two elements in the stack, do nothing.

Exercise 8.5

Below are examples of applications of POP, ADD, and MUL.



(a) POP (remove 7)

(b) ADD $(7 + 2 = 9)$

(c) MUL $(7 \cdot 2 = 14)$

Kahoot

Kahoot

Kahoot

Peergrading 8.5