AlgoDat Klausur Juli 2017

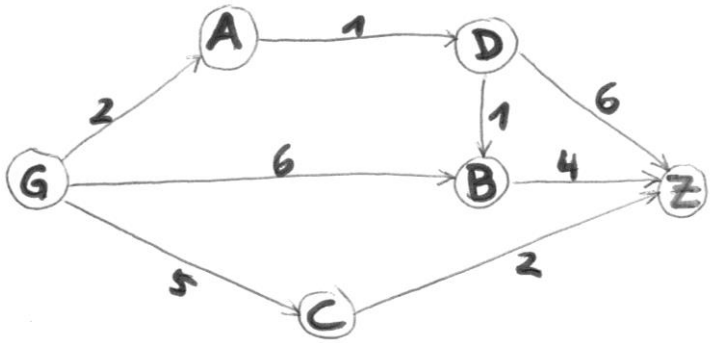
1. Theoriefragen

1) Welchen Methoden und Klassen haben Zugriff auf "protected"?
2) Beispielcode mit abstract class -> muss eine Klasse, die von der abstract Klasse erbt eine Methode implementieren, die nicht abstract war?
3) Wofür kann Tiefensuche genutzt werden?
   a) Um Pfade für Ford-Fulkerson zu finden
   b) Für topological Sorting
   c) Um negative Zyklen zu finden
   d) Um kürzeste Wege in Graphen zu finden in ungerichteten Graphen ohne Kantengewichte

…… (ca. 15-20 Fragen, genau die richtigen Antworten mussten getroffen werden)

2. Dijkstra

Wende den Dijkstra Algorithmus auf folgenden Graph an.



Current Node	Distance in Nodes						Nodes in PrioQueue
	S	A	B	C	D	Z	
	0	∞	∞	∞	∞	∞	
S	0	2	6	5	∞	∞	A, C, B
A							
…							
…							

3. Hashing

Hashfunktion: h(x,y) = (5*x + 7*y) mod 9

1) Fülle die Hashtabelle mit den Elementen (in der Reihenfolge):
{(2, 3), (4, 4), (1, 1), (1, 3), (2, 1)}.

Bei Konflikten nutze lineares Sondieren mi Inkrement 1.

Index	0	1	2	3	4	5	6	7	8
Wert									
Hashwert									

2) Lösche, wenn möglich, die Elemente (4, 4) und (1, 3).

Index	0	1	2	3	4	5	6	7	8
Wert									

3) Fülle die Hashtabelle (Werte siehe oben) und nutze bei Konflikten eine verkettete Liste.

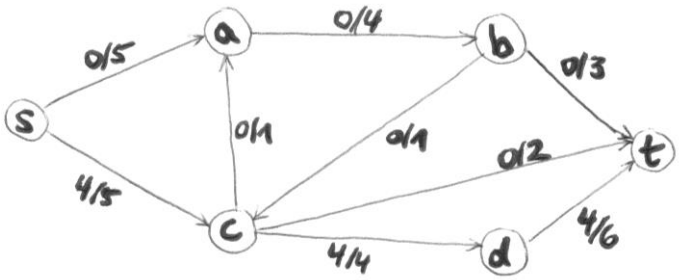
Index	0	1	2	3	4	5	6	7	8
Wert 1									
Wert 2									
Wert 3									

4. FizzBuzz

Das FizzBuzzProblem ist ein beliebtes Problem bei Bewerbungsgesprächen. Übergeben wird eine Zahl n. Statt System.out.println darf print verwendet werden.
Implementiere den Pseudocode.

```
Fuer element in 1 - n :
    Wenn element durch 15 teilbar (ohne Rest) gib FizzBuzz aus
    Sonst wenn element durch 3 teilbar gib Fizz aus
    Sonst wenn element durch 5 teilbar gib Buzz aus
    Sonst gib element aus
```

5. Restflussgraph



1) Zeichne den Restflussgraphen.

2) Welchen Pfad wählt der Algorithmus von Edmonds-Karp als Nächstes?

3) Zeichne den neuen Graph mit dem Pfad der von EK gewählt wurde.

6. Fraction

```
public class Fraction{
      public int numerator;
      public int denumerator;

      public Fraction(int a, int b){
            this.numerator  =  a;
            this.denumerator  =  b;
      }

      public double getValue(){


      }

      public void multiplyWih(Fracttion other){


      }
}
```

1) Schreibe getValue() die den Bruch als Dezimalzahl zurückgibt und multiplyWith() die zwei Brüche miteinander multipliziert.

2) Schreibe eine main, die die Brüche x = 1/3 und y = 2/5 erstellt. Dann multipliziere x mit y und gib das Ergebnis als Dezimalzahl aus.

7. Assertion

func(x) ist eine beliebige Funktion. Es soll gelten:

asserTrue(func(x) >= 0)
assertTrue(func(x) == x || func(x) == -x)

Schreibe func(x).

8. Fehler suchen in Java-Code

Laufzeitfehler oder Compilerfehler?

9. Komplexität analysieren

……