TECHNISCHE UNIVERSITÄT BERLIN
Fakultät IV – Elektrotechnik und Informatik
Fachgebiet Neurotechnologie (MAR 4-3)
Prof. Dr. Benjamin Blankertz
Röhr / Stahl

Algorithmen und Datenstrukturen, SoSe 18
Klausur am 04.10.2018

Bitte füllen Sie alle folgenden Felder aus:

Klausur-ID:	**666B**
tubIT-login:	
Vorname:	
Nachname:	
Matrikelnummer:	
Studiengang:	
Hochschule:	

Durch meine Unterschrift bestätige ich die Korrektheit obiger Angaben.

Ort, Datum

Unterschrift

Beachten Sie die folgenden Hinweise!

- Die Klausuren sind nummeriert und werden anonymisiert korrigiert. Bitte schreiben Sie Ihren Namen **nur** auf das Deckblatt.
- Insgesamt können in der Klausur **100 Punkte** erreicht werden.
- Diese Klausur besteht mit diesem Deckblatt aus den (nummerierten) Seiten **1-14**.
- Schreiben Sie **nicht** mit roter Farbe, grüner Farbe oder Bleistift. Diese Lösungen werden nicht bewertet!
- Notieren Sie Ihre Antworten nur auf dem Blatt (oder dessen Rückseite), auf dem auch die zugehörige Aufgabe steht, da die Aufgaben getrennt korrigiert werden.
- Geben Sie nur eine Lösung pro Aufgabe ab, streichen Sie alle alternativen Lösungsansätze auf Schmier-/Notitzblättern durch.
- Bitte den Barcode am Ende der Seiten nicht beschädigen oder überschreiben.

Zusatzblätter No.:	

$\log N$

28

15

1

1

G E A B C F D

0 1 2 3 4 5 6 7

F E A B C G D

3

1 1 2 3 2 4 6

E BCFADG

Aufgabe 3: Laufzeit $(5 + 2 + 5 = 12$ Punkte)

Sei $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$ ein gerichteter Graph, der wie in der Vorlesung mit Adjanzenlisten implementiert ist. Der Graph habe keine reflexiven Kanten (d.h. keine Kanten $v \to v$ von einem Knoten zu sich selbst).

(a) Geben Sie eine möglichst niedrige (bzw. genaue) Wachstumsordnung für folgenden Pseudocode an. Begründen Sie Ihre Herleitung der Laufzeit mit Bezug auf Zeilennummern.

```
1  // gegeben ein gerichteter Graph G = (V, E)
2  bool ← false
3  für alle u ∈ V                                  O(V)
4    für alle v ∈ V                                O(V)
5      für alle w ∈ V mit w ≠ u                    O(V)
6        falls (u,v) ∈ E und (v,w) ∈ E und (w,u) ∈ E     O(E)
7          bool ← true;
8        end
9      end
10   end
11 end
```

$O(V^3 \cdot E)$

(b) Beschreiben Sie in einem Satz, was der Code in a) macht.

Check ob in G einem Circle mit Länge 3 existiert

(c) Formulieren Sie einen schnelleren Algorithmus in Pseudocode und geben Sie eine möglichst genaue Abschätzung der Laufzeit an.

```
boolean[] marked;
int[] parent;
boolen hasCircle;

for alle v in V:
    if !marked[v]:
        dfs(G, v)

dfs(G, v)
    marked[v] = true
    for w mit e = v->w
        if !marked[w]
            parent[w] = v
            dfs(G, w)
        else if (parent[v] != w)
            hasCycle = true
```

$O(V + E)$

Aufgabe 4: Objektorientierte Programmmierung (10+3 = 13 Punkte)

(a) Es sind drei Klassen gegeben, wobei `Repeat` und `Multiple` von der Klasse `Echo` erben. In diesem Code sind vier Fehler eingebaut. Geben Sie zu jedem Fehler jeweils die Methode, die Zeile, die Art des Fehlers und eine Berichtigung an.

```
1  abstract class Echo {
2     protected int volume;
3     public Echo(int volume) {
4        this.volume = volume;
5     }
6     abstract void print(String word);
7
8     public void echo(String [] said) {
9        for (int i = 0; i <= word.length; i++) {
10          print(said[i]);        Said
11       }
12    }
13 }
```

```
1  public class Repeat extends Echo {
2     public Repeat() {
3        this(4);
4     }  super
5     public void print(String word) {
6        System.out.println(word + " (" + this.volume + ") ");
7     }
8  }
```

```
1  public class Multiple extends Echo {
2     public Multiple() {
3        super(3);
4     }
5     public void print(String word) {
6        int i = 0;
7        int vol = this.volume;
8     while (i < vol)
9        System.out.print(word + " (" + vol + ") ");
10       vol = vol - 1;
11    }
12 }
```

(b) Betrachten Sie nun die folgende Methode:

```
public class Shout {
   public static void main(String[] args) {
      Echo[] Shouted = new Echo[2];
      Shouted[0] = new Repeat();
      Shouted[1] = new Multiple();
      for (Echo v : Shouted){
         v.echo(args);
      }
   }
}
```

Was wird nach einer vollständigen Korrektur des obigen Codes bei der Ausführung auf der Konsole ausgegeben, wenn Sie „Viel Erfolg!“? eingeben?

1 2 3 8 9 10

2 3 1 9 10 8

< 10

= 9

-	A
A	BCD
B	CDEG
C	DEGF
D	EGF
E	GFH
G	FHI

A B C D E G F H I

H - E - F - B - A

G F B A

A B F E

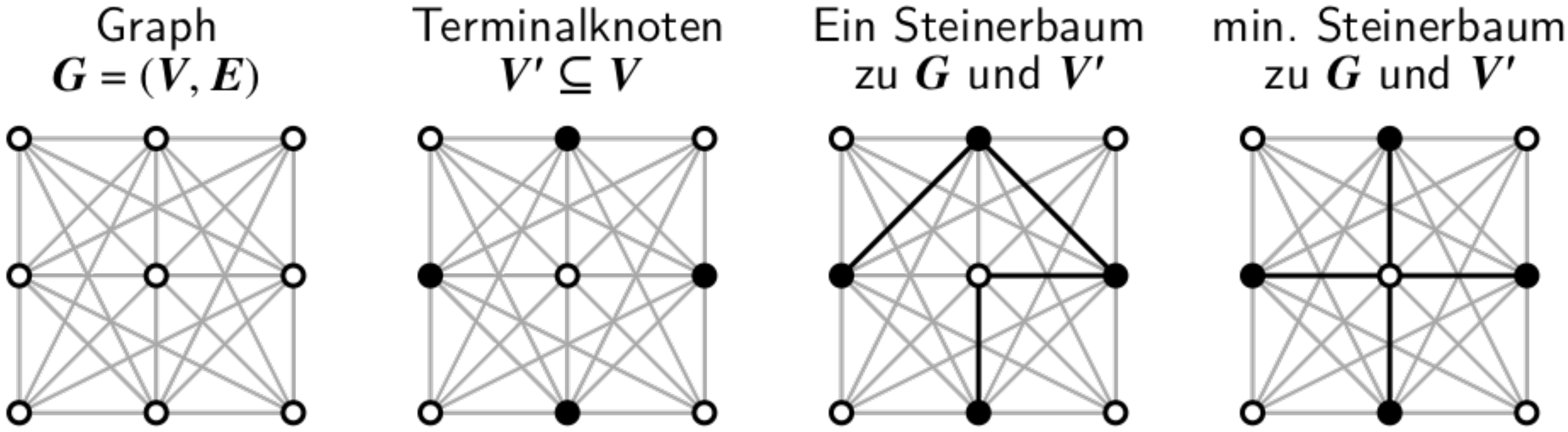
$x = 8$

$y > 8$

C

Aufgabe 9: Approximative Algorithmen ($2+7=9$ Punkte)

Es sei ein vollständiger Graph $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$ mit Kostenfunktion c gegeben. Ein *Steinerbaum* zu $\boldsymbol{G}$ und einer Menge von Terminalknoten $\boldsymbol{V}' \subseteq \boldsymbol{V}$ ist ein Teilgraph $\boldsymbol{T}$ von $\boldsymbol{G}$, der ein Baum ist und alle Terminalknoten $\boldsymbol{V}'$ enthält, siehe Abbildung. Ein *minimaler Steinerbaum* zu $\boldsymbol{G}$ und $\boldsymbol{V}'$ ist ein Steinerbaum, dessen Kanten die geringste Summe von Kantengewichten unter allen Steinerbäumen besitzt. Bei der Definition ist zu beachten, dass der Steinerbaum auch nicht-terminale Knoten (d.h. Knoten aus $\boldsymbol{V} - \boldsymbol{V}'$) beinhalten kann.



Wir betrachten die Aufgabe, einen minimalen Steinerbaum zu $\boldsymbol{G}$ und $\boldsymbol{V}'$ in dem metrischen Fall zu bestimmen ('metrisch' bedeutet, dass die Kosten, bzw. Kantengewichte, die Dreiecksungleichung erfüllen: $c(u,w) \leq c(u,v) + c(v,w)$). Diese Aufgabe ist NP-vollständig.

Hinweis: Ein Graph $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$ heißt *vollständig*, wenn er alle (nicht-reflexiven) Kanten enthält, also $\boldsymbol{E} = \{(v,w) \mid v,w \in \boldsymbol{V} \text{ mit } v \neq w\}$ gilt.

(a) Beschreiben Sie in einem Satz, was ein ρ-Approximationsalgorithmus zu einem Minimierungsproblem ist.

(b) Geben Sie einen effizienten Algorithmus (d.h. mit polynomieller Laufzeit) an, dessen Lösung maximal die doppelten Kosten des minimalen Steinerbaums besitzt. Belegen Sie die Approximationsgüte.

Tipp: Denken Sie an die Approximation des metrischen TSP in der Vorlesung.