Name: ..... **Matr.-Nr.:** .....

Multiple-Choice-Test zu Grundlagen der Algorithmik (A)
TU Berlin, 20.05.2019

(Niedermeier/Bentert, Sommersemester 2019)

Arbeitszeit: 20 Minuten, Gesamtpunktzahl: 25

Hinweis: Je Aufgabe ist **mindestens** eine Antwortmöglichkeit korrekt.
Sobald eine **falsche** Antwortmöglichkeit angekreuzt wurde, gibt es **Null** Punkte für die betroffene Aufgabe.
Viel Erfolg!

Aufgabe 1: **Stable Matching** (6 Punkte)

Gegeben seien die Mengen $\{A, B, C\}$ der Männer und $\{X, Y, Z\}$ der Frauen mit folgenden Präferenzen:

$$A : X < Y < Z, \qquad X : C < B < A,$$
$$B : X < Z < Y, \qquad Y : A < C < B,$$
$$C : X < Z < Y, \qquad Z : A < C < B.$$

Dabei bedeutet z.B. „$A : X < Y < Z$“, dass A findet, dass X besser als Y ist und Y besser als Z ist. Welche der folgenden Aussagen sind wahr?

- ☐ Die Zuordnung $(A, Y), (B, Z), (C, X)$ ist stabil.
- ☐ Die Zuordnung $(A, Y), (B, X), (C, Z)$ ist stabil.
- ☐ Der Propose&Reject-Algorithmus aus der Vorlesung berechnet $(A, Y), (B, X), (C, Z)$.
- ☐ Der Propose&Reject-Algorithmus aus der Vorlesung berechnet $(A, Y), (B, Z), (C, X)$.

Erinnerung: Eine Zuordnung heißt stabil, wenn es kein Paar (M, W) gibt, sodass sich sowohl M als auch W gegenüber dem in der gegebenen Zuordnung zugeteilten Partner bevorzugen.

Aufgabe 2: **Repräsentative Probleme** (3 Punkte)

In einer Fahrschule gibt es 32 Fahrschüler und vier Fahrlehrerinnen. Die Fahrschüler können Fahrstunden buchen, die jeweils 50, 100 oder 180 Minuten andauern. Die Fahrschüler geben hierzu den Startzeitpunkt und die gewünschte Dauer an. Ein automatisches System prüft für jede Buchung, ob es möglich ist mit den vier Fahrlehrerinnen alle Buchungswünsche zu erfüllen und akzeptiert dann entweder die neue Buchung oder teilt dem jeweiligen Fahrschüler mit, dass der Buchungswunsch leider nicht bestätigt werden kann, da nicht genügend Fahrlehrerinnen zur Verfügung stehen. Jede Fahrlehrerin kann jedem Buchungswunsch zugeordnet werden und das System muss bei der Bestätigung noch nicht entscheiden, welche Fahrlehrerin es dem Buchungswunsch zuordnet.

Welches Problem muss das automatische System für jeden Buchungswunsch lösen? (Genau eine Antwort ist korrekt!)

- ☐ Independent Set
- ☐ Interval Partitioning
- ☐ Interval Scheduling
- ☐ Competitive Facility Location

Hinweis:

Problemname	Fragestellung
Independent Set	Gibt es k „unabhängige“ (paarweise nicht benachbarte) Knoten in einem gegebenen Graph?
Interval Scheduling	Gibt es für eine gegebene Menge aus n Jobs mit Start- und Endzeiten eine Teilmenge der Größe k, in der sich keine zwei Jobs zeitlich überschneiden?
Interval Partitioning	Ist es k Maschinen möglich, eine gegebene Menge aus n Jobs mit Start- und Endzeiten abzuarbeiten, sodass jeder Job von genau einer Maschine ausgeführt wird und jede Maschine zu jedem Zeitpunkt an nur einem Job arbeitet?
Competitive Facility Location	Zwei Spielerinnen wählen abwechselnd Knoten aus einem gegebenen knotengewichteten Graphen und jeder gewählte Knoten wird samt seinen Nachbarn gelöscht. Spielerin 1 beginnt und will erreichen, dass Spielerin 2 so wenig Punkte wie möglich bekommt.

Aufgabe 3: **Laufzeitanalyse** (6 Punkte)

Welche der folgenden Angaben sind korrekte Laufzeitabschätzungen des folgenden Algorithmus? Beachten Sie, dass z.B. eine Laufzeit von $O(n^3)$ auch eine korrekte Laufzeitabschätzung für jeden Linearzeitalgorithmus ist.

Input: Eine Menge $J = \{(s_1, t_1), (s_2, t_2), \ldots, (s_n, t_n)\}$ von n Jobs mit Start- und Endzeiten (im folgenden start(j) und end(j) genannt) und eine natürliche Zahl $k > 0$.
Output: Antwort auf die Frage, ob eine Maschine mindestens k Jobs abarbeiten kann.

```
1  Sortiere Jobs aus J nach aufsteigenden Endzeiten;
2  foreach i ∈ J (entsprechend der Sortierung) do
3      answer ← answer + 1;
4      foreach j ∈ J \ {i} do
5          if start(j) ≤ end(i) then
6              lösche j aus J;
7      lösche i aus J;
8  if answer ≥ k then
9      return true
10 return false
```

- ☐ $O(n)$
- ☐ $O(n^2)$
- ☐ $O(n \log n)$
- ☐ $O(n^3)$

Aufgabe 4: **Minimaler Spannbaum** (6 Punkte)

Das aus der Vorlesung bekannte Problem Minimaler Spannbaum ist wie folgt definiert:

Eingabe: Ein zusammenhängender und ungerichteter Graph G mit beliebigen Kantengewichten.
Aufgabe: Finde einen Baum in G, der alle Knoten von G enthält und bei dem die Summe der Kantengewichte minimal ist.

Welche der folgenden Greedy-Strategien liefern immer eine optimale Lösung?

☐ Solange es noch zwei Knoten gibt, die nicht durch einen Pfad verbunden sind, nimm eine Kante mit dem kleinsten Gewicht in die Lösung.

☐ Führe den folgenden Schritt so oft wie möglich aus: Nimm von den nicht gewählten Kanten die Kante mit kleinstem Gewicht, die keinen Kreis erzeugt, in die Lösung.

☐ Wähle einen beliebigen Knoten und eine leere Kantenmenge als Startgraph T. Solange T noch nicht alle Knoten enthält, führe folgende zwei Schritte aus: (1) Wähle eine Kante e mit minimalem Gewicht aus, die einen noch nicht in T enthaltenen Knoten v mit T verbindet. (2) Füge e und v dem Graphen T hinzu.

☐ Wähle für jeden Knoten v die Kante in die Lösung, die von allen Kanten, die v als Endpunkt haben, das kleinste Gewicht hat.

Hinweis: Ein Graph $G = (V, E)$ ist ein Kreis, falls $V = \{v_1, v_2, \ldots, v_n\}$ und $E = \{\{v_1, v_2\}, \{v_2, v_3\}, \ldots, \{v_{n-1}, v_n\}, \{v_n, v_1\}\}$.

Aufgabe 5: **Independent Set auf Wäldern** (4 Punkte)

Vervollständigen Sie folgenden Greedy-Algorithmus für Independent Set auf Wäldern, indem Sie die korrekten Lückenfüller auswählen. Die Funktion $\deg(v)$ gibt die Anzahl der Nachbarn eines Knoten v an.

Input: Ein ungerichteter Wald $G = (V, E)$.
Output: Ein größtmögliches Independent Set in G.

```
1 answer ← ∅
2 while ∃v ∈ V mit deg(v) = 1 do
3     Sei u der Nachbar von v;
4     [ A ]
5     lösche u und v (und alle an u anliegenden Kanten) aus G;
6 while ∃w ∈ V mit deg(w) = 0 do
7     [ B ]
8 return answer
```

☐ A: `answer` ← `answer` ∪ {u}; | B: `answer` ← `answer` ∪ {w}; lösche w aus G;

☐ A: `answer` ← `answer` ∪ {u}; | B: lösche w aus G;

☐ A: `answer` ← `answer` ∪ {v}; | B: lösche w aus G;

☐ A: `answer` ← `answer` ∪ {v}; | B: `answer` ← `answer` ∪ {w}; lösche w aus G;