Universität Stuttgart Fakultät Informatik

Name:________________________________

7.6.1999

Test 1
Entwurf und Analyse effizienter Algorithmen, SS 1999

Sie haben für die 22 Fragen 25 Minuten Zeit. Kennzeichnen Sie Ihren Bearbeitungszettel in eindeutiger Weise. Wenn Sie einen Schein erwerben wollen, müssen Sie Ihren Namen angeben, da die Tests in die Übungswertung eingehen.

Jede Frage kann keine, eine oder mehrere richtige Antworten haben. Eine Frage bringt genau dann einen Punkt, wenn sie genau richtig beantwortet wurde. 11 Punkte sind zum „Bestehen" des Tests notwendig.

Was gilt für $A := \mathcal{O}(2^n)$ und $B := \mathcal{O}(n2^n)$?	⊗ $A \subseteq B$	○ $A \supseteq B$	⊗ $A \cap B \neq \emptyset$
Was gilt für $A := \mathcal{O}(4^{log(n)})$ und $B := \mathcal{O}(n^2)$?	⊗ $A \subseteq B$	⊗ $A \supseteq B$	⊗ $A \cap B \neq \emptyset$
Was gilt für $A := \mathcal{O}(\|\sin(\frac{\pi n}{2})\|)$ und $B := \mathcal{O}(\|\cos(\frac{\pi n}{2})\|)$?	○ $A \subseteq B$	○ $A \supseteq B$	⊗ $A \cap B \neq \emptyset$
Was gilt für $A := \Omega(n)$ und $B := \Theta(n^2)$?	○ $A \subseteq B$	⊗ $A \supseteq B$	⊗ $A \cap B \neq \emptyset$
Was gilt für $A := \Theta(n^2)$ und $B := o(n^2)$?	○ $A \subseteq B$	○ $A \supseteq B$	○ $A \cap B \neq \emptyset$
Was gilt für $A := \omega(n!)$ und $B := o(2^{n \log(n)})$?	○ $A \subseteq B$	○ $A \supseteq B$	⊗ $A \cap B \neq \emptyset$
Welche Laufzeit hat der Quicksort-Algorithmus?	○ $\mathcal{O}(n\,log(n))$	○ $\Theta(n^2)$	⊗ $\mathcal{O}(n^2)$
Welche Laufzeit hat der Heapsort-Algorithmus?	⊗ $\mathcal{O}(n\log(n))$	⊗ $\Theta(n\log(n))$	⊗ $\mathcal{O}(\log(n!))$
Welche Laufzeit hat der Mergesort-Algorithmus?	⊗ $\mathcal{O}(n\log(n))$	⊗ $\Theta(n\log(n))$	○ $\Theta(n^2)$
Mit welcher Laufzeit läßt sich der Dijkstra-Algorithmus implementieren($n := \|\text{Knoten}\|, e := \|\text{Kanten}\|$)?	⊗ $\mathcal{O}((n+e)\log(n))$	⊗ $\mathcal{O}(n^2)$	○ $\mathcal{O}(n\log(n))$
Welche Laufzeit hat der Warshall-Algorithmus?	○ $\mathcal{O}(n^2)$	⊗ $\mathcal{O}(n^3)$	⊗ $\mathcal{O}(n^4)$
Welcher Algorithmus berechnet einen minimalen aufspannenden Baum?	○ Floyd	⊗ Prim	○ Warshall
Welcher Algorithmus berechnet kürzeste Wege?	○ Warshall	⊗ Dijkstra	○ Prim
Welcher Algorithmus arbeitet korrekt mit negativen Kantengewichten?	⊗ Prim	○ Floyd	○ Dijkstra
Für welche Graphen ist die Verwendung eines Heaps vorteilhaft beim Dijkstra-Algorithmus?	⊗ Dünne	○ Dichte	○ Alle
Mit welcher Entwurfsstrategie kann die größte Teilfolgensumme in $\mathcal{O}(n)$ berechnet werden?	⊗ Greedy	○ Dynamisches Programmieren	⊗ Divide and Conquer
Zwischen welchen Mengen existieren keine Kanten beim Dijkstra-Algorithmus (B :=abgearbeitete Knoten , R :=Randknoten, U :=noch nicht betrachtete Knoten)?	⊗ B und U	○ R und R	○ R und U
Was gilt für die Catalanschen Zahlen $T(n)$?	○ $T(n) \in \mathcal{O}(2^n)$	⊗ $T(n) \in 2^{\mathcal{O}(n)}$	⊗ $T(n) \in \mathcal{O}(n!)$

Dijkstra-Algorithmus: Sei $G = (V, E, \delta)$; es wird das $v' \in R$ in B aufgenommen, für das gilt:

⊗ $\exists v \in B\ \forall v'' \in B\ \forall v''' \in R$ ist $d(v) + \delta(v, v') \leq d(v'') + \delta(v'', v''')$

○ $\forall v \in B\ \exists v'' \in B\ \forall v''' \in R$ ist $d(v) + \delta(v, v') \leq d(v'') + \delta(v'', v''')$

○ $\exists v \in B\ \forall v'' \in B\ \forall v''' \in R$ ist $\delta(v, v') \leq \delta(v'', v''')$

Prim-Algorithmus: Sei $G = (V, E, \delta)$; es wird das $v' \in R$ in B aufgenommen, für das gilt:

○ $\forall v \in B\ \exists v'' \in B\ \forall v''' \in R$ ist $d(v) + \delta(v, v') \leq d(v'') + \delta(v'', v''')$

○ $\forall v \in B\ \exists v'' \in B\ \forall v''' \in R$ ist $\delta(v, v') \leq \delta(v'', v''')$

⊗ $\exists v \in B\ \forall v'' \in B\ \forall v''' \in R$ ist $\delta(v, v') \leq \delta(v'', v''')$

Sortieralgorithmen: Zum Sortieren eines Feldes benötigt

○ Quicksort stets weniger Vergleiche als Mergesort

⊗ Mergesort stets mehr Platz als Quicksort

⊗ Direktes binäres Einfügen $\Theta(n\log(n))$ Vergleiche

Suchbäume: Für optimale Suchbäume gilt:

○ Es gilt, dass Knoten mit höherer Zugriffswahrscheinlichkeit stets weiter oben im Suchbaum liegen, d.h. $\forall i, j : p(a_i) \leq p(a_j) \Rightarrow \lambda(\gamma^{-1}(a_i)) \leq \lambda(\gamma^{-1}(a_j))$

○ Der optimale Suchbaum ist eindeutig.

⊗ Der optimale Suchbaum ist bipartit.