Wintersemester 2021/22
Prof. Dr. S. Held
Dr. U. Brenner

Forschungsinstitut für Diskrete Mathematik
Universität Bonn

Einführung in die Diskrete Mathematik
4. Übung

1. Wir wollen eine Menge von Objekten verwalten. Jedem Objekt ist ein Schlüssel zugeordnet, und keine zwei Objekte haben denselben Schlüssel. Außerdem gibt es eine lineare Ordnung auf allen Schlüsseln. Insbesondere wollen wir effizient ein Element mit einem gegebenen Schlüssel suchen. Mit einem sortierten Array geht das durch binäre Suche offenbar in logarithmischer Zeit. Zusätzlich wollen wir aber auch effizient neue Elemente einfügen, was bei einem sortierten Array im allgemeinen lineare Laufzeit erfordern würde. Betrachten Sie statt dessen die folgende Datenstruktur:

   Wir speichern die Objekte in einer Menge von sortierten Arrays, die alle unterschiedliche Länge haben. Die Länge eines jeden Arrays soll eine Zweierpotenz einer ganzen Zahl sein. Jedes Element der Menge soll in genau einem Array vorkommen.

   (a) Zeigen Sie, dass es stets eine solche Zerlegung in Arrays gibt und dass die Längen der Arrays eindeutig sind.

   (b) Beschreiben Sie, wie man ein Element mit gegebenem Schlüssel in Zeit $O(\log^2 n)$ finden kann, wenn n die Zahl der eingefügten Elemente ist.

   (c) Zeigen Sie, dass man, wenn man mit einer leeren Menge beginnt, für eine Folge von n Einfügeoperationen Zeit $O(n \log n)$ benötigt. (1+1+3 Punkte)

2. a) Zeigen Sie, dass es Folgen von Heap-Operationen gibt, so dass in einem Fibonacci-Heap die maximale Pfadlänge in einer Arboreszenz $\Theta(n)$ ist, wenn n die Zahl der Elemente ist.

   b) Zeigen Sie, dass zwei Fibonacci-Heaps mit n_1 und n_2 Elementen in $O(\log(n_1+n_2))$ Zeit verschmolzen werden können. Das Ergebnis soll also ein Fibonacci-Heap sein, der alle $n_1 + n_2$ Elemente enthält. (3+2 Punkte)

3. Betrachten Sie folgenden Algorithmus, um in einem gegebenen gerichteten Graphen G mit Gewichten $l : E(G) \to \mathbb{R}_+$ zu einem Knoten $r \in V(G)$, von dem aus jeder Knoten in G erreichbar ist, eine aufspannende Arboreszenz T mit Wurzel r und mit minimalem Gewicht $\sum_{e \in E(T)} l(e)$ zu bestimmen: Es sei $G_0 := (V(G), \{e \in E(G) \mid l(e) = 0\})$. Wenn G_0 eine aufspannende Arboreszenz mit Wurzel r enthält, gibt man eine solche zurück. Andernfalls wählt man eine starke Zusammenhangskomponente K von G_0 mit $r \notin V(K)$ und $l(e) > 0$ für alle $e \in \delta_G^-(V(K))$. Überlegen Sie sich, warum eine solche existiert. Es sei $\alpha := \min\{l(e) \mid e \in \delta_G^-(V(K))\}$. Setze nun $l'(e) := l(e) - \alpha$ für $e \in \delta_G^-(V(K))$ und $l'(e) := l(e)$ für $e \in E(G) \setminus \delta_G^-(V(K))$. Dann berechnet man rekursiv eine kostenminimale aufspannende Arboreszenz T mit Wurzel R bezüglich l'. Zeigen Sie, dass T so gewählt werden kann, dass $|\delta_T^-(V(K))| = 1$ gilt. Eine solche Arboreszenz T gibt man dann zurück.

   Beweisen Sie, dass dieser Algorithmus korrekt funktioniert. (5 Punkte)

4. Zeigen Sie, dass Edmonds' Branching-Algorithmus zur Berechnung eines maximal gewichteten Branchings in einem gerichteten Graphen G mit Kantengewichten $c : E(G) \to \mathbb{R}_+$ so implementiert werden kann, dass er Laufzeit $O(m + n \log n)$ hat. Dabei sei $n = |V(G)|$ und $m = |E(G)|$
   Hinweis: Verwenden Sie Fibonacci-Heaps. (5 Punkte)

Abgabe: Dienstag, der 9.11.2021, **vor** der Vorlesung im Hörsaal