Theoretische Informatik II

Übung 4

Übungsgruppe: Sommersemester 2024

Aufgabe 3

(a) **Eingabe**: Ein ungerichteter Baum, da die Xa'Leh nur minimal viele Sprungtore haben, die trotzdem alle Systeme miteinander verbinden

(b) **Ausgabe**: Boolean Array A an Knoten, das angibt, an welchen Knoten Wartungsstationen gebaut werden sollen.

(c) **Korrektheitsbedingung**: Eine Lösung ist valide, wenn für jede Kante gilt: mindestens ein inzidenter Knoten ist markiert.
Eine Lösung ist korrekt, wenn sie valide ist und es keine valide Lösung gibt, die weniger markierte Knoten hat.

(d) **Algorithmus**: Wir initialisieren A überall mit FALSE. Wir starten an einem beliebigen Knoten und hängen unseren Baum an diesem auf. Nun führen wir eine Tiefensuche durch und markieren einen Knoten (d.h. in A für den Knoten TRUE eintragen) immer dann, wenn wir alle seine Kinder schon betrachtet haben und mindestens eins von ihnen noch nicht markiert ist (also immer dann, wenn wir einen Knoten endgültig verlassen und der Teilbaum fertig bearbeitet ist). Wenn die Tiefensuche terminiert, geben wir A aus.

(e) **Korrektheit**: Wir stellen sicher, dass der direkte Vorgänger von jedem unmarkierten Knoten markiert ist. Daher haben wir keinen Kante, die keinen markierten inzidenten Knoten hat und unsere Lösung ist valide.
Unsere Lösung ist minimal wegen folgenden Arguments: Wenn wir die Markierung eines Knoten entfernen und damit die Anzahl der markierten Knoten verkleinern wollen würden, dann müssen wir jedes (mindestens eins) seiner unmarkierten Kinder markieren markieren und die Anzahl der markierten Knoten bleibt entweder gleich oder wird größer.

(f) **Laufzeit**: Die normale Tiefensuche, hat eine Laufzeit von $O(n + m)$. Da es in einem Baum $m = n - 1$ kanten gibt hat die Tiefensuche eine Laufzeit von $O(n)$. Zusätzlich wird für maximal alle Knoten ein Wert in A geschrieben, also haben wir einen Aufwand in $O(n)$. Beim Rückweg der Tiefensuche überprüft jeder Knoten zusätzlich seine Kindknoten, also wird noch einmal für alle Knoten A in $O(1)$ abgefragt, also insgesamt in $O(n)$. Die Ausgabe des Arrays erfolgt in $O(1)$.

Damit hat der Algorithmus insgesamt eine Laufzeit von $O(n)$