KIT
Institut für Theoretische Informatik

Jun.-Prof. Henning Meyerhenke, Jun.-Prof. Dennis Hofheinz
Christian Staudt, Christoph Striecks

12. Übungsblatt zu Algorithmen I im SS 2015

https://crypto.iti.kit.edu/algo-sose15
{staudt,striecks}@kit.edu

Die Aufgaben auf diesem Blatt sind optional, werden nicht bewertet und auch nicht korrigiert. Eine Abgabe ist daher **nicht** nötig.

Aufgabe 1 (*Programmieraufgabe: Kruskals Algorithmus, Union-Find-Datenstruktur*)

Die aus der Vorlesung bekannte Union-Find-Datenstruktur lässt sich durch verschiedene Optimierungen ergänzen. Sie ist die zentrale Komponente im Algorithmus von Kruskal, der einen minimalen Spannbaum in einem ungerichteten, gewichteten Graphen berechnet. Das Grundgerüst dafür finden Sie als Java-Code in folgendem Mercurial-Repository:

`https://algohub.iti.kit.edu/parco/Teaching/ProgrammieraufgabeAlgoI`

Implementieren Sie das Interface `UnionFind` mehrfach mit den verschiedenen Varianten der Datenstruktur und geben Sie dem Algorithmus `MinimumSpanningTreeAlgorithm` Instanzen der Implementierung als Parameter mit. Messen Sie die Laufzeit des Algorithmus für die einzelnen Varianten und stellen Sie fest, wieviel Beschleunigung die Optimierungen in der Praxis bringen.
Einen kleinen Testgraphen finden Sie im Repository, eine größere Instanz für Laufzeittests liegt unter `https://cloud.iti.kit.edu/index.php/s/2tgCe63twE9HrzV`

Aufgabe 2 (*Neues Rucksackproblem*)

Betrachten Sie folgendes eindimensionales Rucksackproblem: Sie haben eine Liste von n Gegenständen mit Volumina $c_1, \ldots, c_n \in \mathbb{N}$ und Nutzwert $a_1, \ldots, a_n \in \mathbb{N}$. Ihr Rucksack hat eine Kapazität $C \in \mathbb{N}$ und soll so gepackt werden, dass die Summe der Nutzwerte der mitgenommenen Gegenstände maximal ist. Formal ist eine Menge $I \subseteq \{1, \ldots, n\}$ gesucht, die

$$\left\{ \sum_{i \in I} a_i \;\middle|\; \sum_{i \in I} c_i \leq C \right\} \quad \text{maximiert.}$$

a) Wiederholen Sie das in der Vorlesung vorgestellte dynamische Programm, dass in $\mathcal{O}(nC)$ eine solche optimale Menge I berechnet.

b) Sei nun *Opt* der maximale Nutzen eines zulässigen Rucksacks. Entwickeln Sie ein dynamisches Programm, das in $\mathcal{O}(n \cdot Opt)$ eine optimale Lösungsmenge I berechnet.

c) Ein Algorithmus heißt ein *fully polynomial time approximation scheme* (FPTAS), wenn er in Zeit $p(n, \frac{1}{\varepsilon})$, wobei n die Eingabegröße und p ein Polynom in zwei Variablen ist, eine zulässige (approximierende) Lösung I_1 berechnet, die gegenüber einer optimalen Lösung I^* einen nur geringfügig kleineren Nutzwert garantiert, genauer:

$$\sum_{i \in I_1} a_i \geq (1 - \varepsilon) \sum_{i \in I^*} a_i \,.$$

Entwickeln Sie ein FPTAS für das Rucksackproblem.

Hinweis: Setzen Sie $A := \max_{i=1,\ldots,n} a_i$, skalieren Sie die Nutzwerte mit $\frac{n}{A\varepsilon}$, runden Sie ab und verwenden Sie b).

Verwenden Sie Pseudocode oder eine andere hinreichend präzise Formulierung und begründen Sie die Laufzeit Ihrer Lösungen.