Präsenzblatt 12

Hinweis: Dieses Aufgabenblatt wurde von Tutor:innen erstellt. Die Aufgaben sind für die Klausur weder relevant noch irrelevant.

Aufgabe 12.1: Schneechaos

Stellen Sie sich vor, es ist Winter und es hat in Saarbrücken so viel geschneit, dass das Räumen einer Strecke sehr aufwendig ist. Man hat es inzwischen aufgegeben, jede Straße räumen zu wollen, man soll aber immer noch jeden Punkt in der Stadt erreichen können. Wie kann man das mit möglichst kleinem hinbekommen? (Sie dürfen annehmen, dass für die Räumfahrzeuge die Fahrtkosten auf geräumten Strecken vernachlässigbar sind.)

Aufgabe 12.2: Feuer!

Sie wollen die Ausbreitung eines Feuers simulieren. Sie möchten wissen, zu welchem Zeitpunkt das Feuer an einem bestimmten Wegpunkt ankommt. Nehmen Sie zunächst an, dass alle Wege gleichlang sind. Welchen Algorithmus würden Sie wählen? Wie sieht es aus, wenn die Wege eine beliebige (nicht-negative) Länge haben?

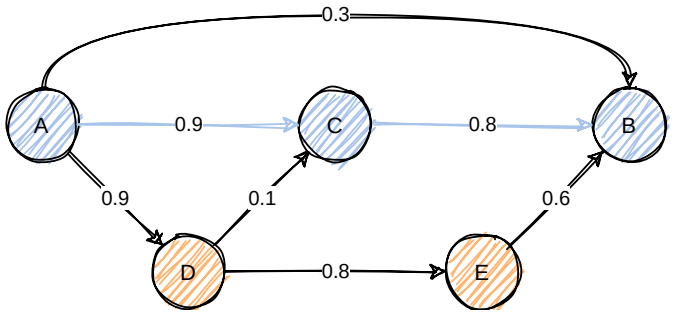
Aufgabe 12.3: Gerichtete Graphen

Sei $G = (V, E)$ ein DAG. Geben Sie einen Algorithmus an, der prüft, ob es einen Pfad gibt, der jeden Knoten genau einmal besucht. Ihre Algorithmus sollte Laufzeit $\mathcal{O}(|V| + |E|)$ haben.

Aufgabe 12.4: Sicherste Pfade

Man könnte das Internet als einen gewichteten Graphen $G = (V, E)$ mit $|V| = n$ und $|E| = m$ modellieren. Die Knoten entsprechen dabei den Rechnern und die Kanten repräsentieren die Verbindungen zwischen ihnen. Weiterhin ist jede Kante e mit der Wahrscheinlichkeit $w(e)$ dafür gewichtet, dass die Datenübertragung problemlos vonstatten geht, also dass sie nicht fehlschlägt.

Wir nehmen weiterhin an, dass die Wahrscheinlichkeiten unabhängig voneinander sind. Damit ergibt sich die Wahrscheinlichkeit, dass Pfad mit den Kanten $e_1, \ldots, e_n$ sicher ist, als $\prod_{i=1}^{n} w(e_i)$. Nehmen Sie weiter an, dass keine Leitung tot oder völlig sicher ist, also dass $0 < w < 1$ gilt.



Um im obigen Beispiel sicher eine E-Mail von A nach B zu verschicken, sollte man den Pfad $\langle A, C, B\rangle$ wählen, da die Erfolgswahrscheinlichkeit $P = 0,9 \cdot 0,9 = 0,72$ hier maximal ist.

1. Beschreiben Sie einen Algorithmus mit Laufzeit $\mathcal{O}(n \log n + m)$, der den sichersten Pfad von $s \in V$ nach $t \in V$ berechnet.
2. Argumentieren Sie, warum Ihr Algorithmus korrekt ist. Es ist kein vollständiger Beweis notwendig.
3. Erklären Sie, warum Ihr Algorithmus die geforderte Laufzeit hat.

Hinweis: Übertragen Sie das Problem auf einen Ihnen bekannten Graph-Algorithmus. Es gilt:

$$w(e_1) \cdot w_(e_2) = c^{\log_c w(e_1)} \cdot c^{\log_c w(e_2)} = c^{\log_c w(e_1) + \log_c w(e_2)} \quad \text{für } c \in (0,1)$$

Aufgabe 12.5: Floyd-Warshall

Sie wollen für jedes beliebige Knotenpaar den kürzesten Weg ermitteln. Entwickeln Sie mithilfe dynamischer Programmierung einen Algorithmus, der dies in $\mathcal{O}(|V|^3)$ schafft.

Hinweis: Lassen Sie für den Weg schrittweise mehr Knoten zu. Am Anfang kann jeder Knoten also nur sich selbst erreichen, als nächstes kann der Weg auch über Knoten 1 führen, dann über die Knoten 1 und 2 usw.

Aufgabe 12.6: Graph-Comparison

Füllen Sie die Lücken in folgender Tabelle aus. Mit „Sparse“ ist die Laufzeit auf einem Graphen ohne Kanten gemeint, mit „Dense“ die Laufzeit auf einem Graphen mit maximal vielen Kanten. Eingabegröße meint hier, welche Größenordnung n, m haben dürfen, damit die Berechnung noch praktikabel ist. Es handelt sich natürlich nicht um harte Schranken. Mit „APSP“ ist die Laufzeit gemeint, die man erhält, wenn man für alle Knotenpaare den kürzesten Weg ermitteln möchte.

	BFS	Dijkstra	Bellman-Ford	Floyd-Warshall
Laufzeit				
⟶ Sparse				
⟶ Dense				
Ungewichtet (✓/✗)				
Gewichtet (✓/✗)				
Negative Kanten (✓/✗)				
Eingabegröße	$n, m \leq 10^7$	$n, m \leq 10^6$	$n \cdot m \leq 10^7$	$n \leq 500$
APSP				

Aufgabe 12.7: AVL

(a) Beantworten Sie die folgenden Fragen. Entscheiden Sie bei Aussagen, ob diese wahr oder falsch sind und geben Sie eine Begründung an.

1. Es gibt eine Sequenz von 4 Zahlen, sodass beim Einfügen der Zahlen in einen anfangs leeren AVL-Baum bei der vierten Zahl eine Rotation erfolgt.
2. In welcher Reihenfolge sollte man die Schlüssel $1, 2, 3, 4, 5, 6, 7$ in einen AVL-Baum einfügen, sodass möglichst wenig Rotationen entstehen?
3. Man kann in $\mathcal{O}(n)$ entscheiden, ob die Schnittmenge (der Schlüsselmengen) von zwei AVL-Bäumen mit je n Elementen leer ist.
4. Ein AVL-Baum mit Höhe 6 hat mindestens 30 Elemente.
5. Dieter Schlau nennt eine (endliche) Menge $M \subseteq \mathbb{N}$ von Zahlen hübsch, wenn $\max_{x,y \in M} |x - y| > k$ für ein festes $k \in \mathbb{N}$. Die Menge kann sich ändern, das heißt es können neue Elemente hinzu kommen und Elemente gelöscht werden. Dieter hat sich überlegt, die Elemente in einem AVL-Baum zu speichern. Ist diese Wahl der Datenstruktur geeignet, um dynamisch zu überprüfen, ob eine Menge hübsch ist? Welche Laufzeiten erreicht Dieter? Wären HashSets eine bessere Wahl?

(b) Fügen Sie in einen anfangs leeren AVL-Baum die Zahlen $8, 5, 4, 7, 9, 6, 10$ (in dieser Reihenfolge) ein. Markieren Sie jeweils den Knoten, an dem die AVL-Bedingung zuerst verletzt wird.

Aufgabe 12.8: Heaps

Gegeben seien n Zahlen in einem Feld A, d. h. $A[1], \ldots, A[n] \in \mathbb{Z}$. Durch sequenzielles Einfügen in einen leeren Heap können wir einen binären Min-Heap in Laufzeit von $\mathcal{O}(n \log n)$ produzieren, welcher die Zahlen aus A enthält. Geben Sie einen Algorithmus an, der das in Zeit $\mathcal{O}(n)$ erreicht.

Aufgabe 12.9: Deque

Zur Erinnerung: Ein Stack it ein Datentyp der auf einer Folge $A = \langle a_1, a_2, \ldots, a_z \rangle$ diese drei Operationen realisiert: $A.\textit{size}()$ gibt die Länge t von A zurück, $A.\textit{pop}()$ gibt a_t zurück und ändert A zu $\langle a_1, a_2, \ldots, a_{t-1} \rangle$; schließlich ändert $A.\textit{push}(b)$ die Folge A zur Folge $\langle a_1, \ldots a_t, b \rangle$. Nehmen Sie an, dass jede dieser drei Operationen konstante Zeit im schlechtesten Fall braucht.

Eine Dequeue ist ein Datentyp, der Push und Pop an beide Enden einer Folge zulässt. Zusätzlich zu den Operationen des Stacks gibt es noch: $A.\textit{leftpop}()$ ändert A zu $\langle a_2, \ldots a_n \rangle$ und gibt a_1 zurück sowie $A.\textit{leftpush}(b)$, welche A zu $\langle b, a_1, \ldots, a_t \rangle$ ändert.

In den Übungen haben Sie diese Struktur bereits über ein Feld implementiert, man kann aber auch Stacks verwenden. Die Folge $A = \langle a_1, \ldots, a_t \rangle$ wird an der Stelle s geteilt betrachtet, also $\langle a_1, \ldots, a_s \rangle \langle a_{s+1}, \ldots, a_t \rangle$ und es wird ein Stack für $A_L = \langle a_s, \ldots, a_s \rangle$ und $A_R = \langle a_{s+1}, \ldots, a_t \rangle$ verwendet. *leftpush* und *leftpop* werden über *push* und *pop* in A_L realisiert.

Ein Problem liegt vor, wenn bei leftpop A_L leer ist oder bei *pop* A_R leer ist. Wir betrachten o. B. d. A. den ersten Fall. Sei $A_R = \langle a_1, \ldots a_p \rangle$ mit $p > 1$. Eine Vorgehensweise ist, die Hälfte der Elemente von A_R nach A_L zu verschieben. Dann entsteht $A_L = \langle a_m, \ldots a_1 \rangle$ und $A_R = \langle a_{m+1}, \ldots a_p \rangle$ mit $m = \lfloor p/2 \rfloor$. Das Verschieben geschieht in Zeit $\mathcal{O}(p)$.

Zeigen Sie, dass bei dieser Implementierung jede Operation armortisierte konstante Laufzeit hat.