Algotheorie-Database

May 2021

1 Algorithmen-Übersicht

Algorithm List					
Kategorie	**Problem**	**Algorithmus/Strategie**	**Related/Example Problems**	**Laufzeit**	**Facts**
Stable Matching	Stable Matching	Stable Matching/Propose and Reject	Ordne Medizinstudierende Praktikumsplätzen in Krankenhäusern zu, wobei die gegenseitigen Präferenzen beachtet werden	$\mathcal{O}(n^2)$	• Frauenoptimal (jede Frau bekommt bestmöglichen Mann zugeordnet) • Jeder Mann bekommt schlechtmöglichste Frau zugeordnet
Graphtraversierung und Co.	BFS	Breitensuche mit FIFO		$\mathcal{O}(n+m)$	n = Anzahl Knoten und m = Anzahl Kanten im Graph
	TFS	Tiefensuche mit Stack		$\mathcal{O}(n+m)$	n = Anzahl Knoten und m = Anzahl Kanten im Graph
	Stark Zusammenhängender Graph	BFS von beliebigem Knoten aus		$\mathcal{O}(n+m)$	Ein gerichteter Graph heißt stark zusammenhängend falls jedes Knotenpaar wechselseitig durch jeweils mindestens einen gerichteten Pfad verbunden ist
	Topologische Sortierung	Rekursiv	DAG - Directed Acyclic Graph	$\mathcal{O}(n+m)$	G ist DAG $\Leftrightarrow$ G hat topologische Sortierung

Algorithm List					
Kategorie	**Problem**	**Algorithmus/Strategie**	**Related/Example Problems**	**Laufzeit**	**Facts**
Greedy	Intervall Scheduling	Wähle Prozess mit frühestmöglicher Endzeit		$\mathcal{O}(nlogn)$	
	Intervall Partioning	Greedy - IP	nichts	$\mathcal{O}(nlogn)$	• Eingabe: Intervalle (Jobs) mit Startzeit und Endzeiten $f_i, 1 \leq i \leq n$. • Aufgabe: finde kleinstmögliche Menge von Zeitstrahlen, sodass alle Jobs, auf diese verteilt, nicht überlappen. • Greedy - IP liefert immer optimale Lösung.
	Verspätungsmin.	Führe Jobs gemäß ansteigender Frist d_j aus		$\mathcal{O}(nlogn)$	• Eingabe: Jobs j, $i \leq jn$, mit Zeitdauer t_j und Frist d_j. • Aufgabe: : Finde Ausfuhrungsreihenfolge der Jobs, so dass maximale Versp¨atung minimiert wird • liefert immer optimale Lösung.

Algorithm List					
Kategorie	**Problem**	**Algorithmus/Strategie**	**Related/Example Problems**	**Laufzeit**	**Facts**
Greedy	Kürzeste Wege in Graphen	Dijikstra		$\mathcal{O}(\|E\| + \|V\|log\|V\|)$	• Eingabe: Gerichteter Graph G = (V,E) mit Längenangabe $I_e \geq 0$ für jede Kante $e \in E$, Startknoten s und Endknoten t. • Aufgabe: Finde kürzesten Pfad (Summe der Kantenlängen) von s nach t • liefert immer optimale Lösung.
	Minimale Spannbäume (MST)	Prim (Kruskal auch möglicher, aber schlechter) Erzeuge Baum ausgehend von einem Startknoten durch fortwährende Erweiterung um billigste Kante.	Kurzeste Wege (genau zwei Blätter)	$\mathcal{O}(m + nlogn)$	• Eingabe: Zusammenhängender ungerichteter Graph G mit beliebigen Kantengewichten. • Aufgabe: Finde einen Baum in G, der alle Knoten von G enthält und bei dem die Summe der Kantengewichte minimal ist.

Algorithm List					
Kategorie	**Problem**	**Algorithmus/Strategie**	**Related/Example Problems**	**Laufzeit**	**Facts**
Greedy	Kodierung (Huffman Codes)	Huffman Algorithmus - Bottom Up		$\mathcal{O}(nlogn)$	• Eingabe: Zeichenkette T über endlichem Alphabet $\Sigma = \{c_1, .., c_n\}$ für jeden Buchstaben c_i eine "relative Häufigkeit" $f(c_i) \geq 0$, wobei die Summe der Häufigkeiten 1 ist. • Aufgabe: Kodiere T mit Binäralphabet {0,1}, sodass der entstehende Code minimale Länge hat und präfixfrei ist. • Algorithmus liefert optimale präfixfreie Binärkodierung.

Algorithm List					
Kategorie	**Problem**	**Algorithmus/Strategie**	**Related/Example Problems**	**Laufzeit**	**Facts**
Divide and Conquer	Rekursionsgl.	Master Theorem			WICHTIG DAS ZU VERSTEHEN (VL 4)
	Inversionen Zählen	Divide and Conquer halt :D		$\mathcal{O}(nlogn)$	• Eingabe: Eine feste Ordnung $a_1, a_2, ..., a_n$ der Zahlen von 1 bis n. • Aufgabe: Bestimme die Anzahl von Inversionen im Vergleich zur ORdnung 1,2,...,n wobei eine Inversion ein Paar (i,j), $1 \leq i < j \leq n$ mit $a_i > a_j$ ist. • Algorithmus liefert optimale präfixfreie Binärkodierung.
	Closest Pair	Divide and Conquer halt :D		$\mathcal{O}(nlogn)$	• Eingabe: n Punkte in der Euklidischen Ebene • Aufgabe: Finde Punktepaar mit geringstem Abstand. • Algorithmus liefert optimale präfixfreie Binärkodierung.