Programmierkurs
Wintersemester 2020/21
Übungsblatt 11 (01.02.2021)

M.Sc. Laslo Hunhold
Dept. Math./Inf., Abt. Informatik
Universität zu Köln

Abgabe für Rückmeldungen bis zum 08.02.2021 (08:00 Uhr), Besprechung ab dem 08.02.2021.

Aufgabe 11-0 (Interval Scheduling - Vorlesungsbeispiele)
Prüfen und vollziehen Sie nach, daß die Beispiele 0 und 1 aus Vorlesung 10 jeweils bei den Anfragenauswahlmethoden nach minimaler Überlappung (Folie 10-11) und frühestem Endzeitpunkt (Folie 10-12) mit dem Greedy-Interval-Scheduling-Algorithmus optimale Lösungen liefern.

Aufgabe 11-1 (Interval Scheduling - Vertiefung)
Sei $A := \{a_1, a_2, a_3, a_4\} := \{(1, 11), (2, 4), (5, 7), (8, 10)\}$ eine Menge von Anfragen (wobei die Zahlen jeweils für Uhrzeiten stehen). Beantworten Sie folgende Fragen zum Interval-Scheduling-Problem auf A.

0. Ist $\{a_1, a_2\}$ kompatibel?
1. Ist $\{a_2, a_4\}$ kompatibel?
2. Ist $\{a_1\}$ eine kardinalitätsmaximale Lösung?
3. Ist $\{a_1\}$ eine inklusionsmaximale Lösung?
4. Ist $\{a_2, a_3, a_4\}$ eine kardinalitätsmaximale Lösung?
5. Ist $\{a_2, a_3, a_4\}$ eine inklusionsmaximale Lösung?

Aufgabe 11-2 (Druckerei)
Stellen Sie sich folgendes Szenario vor: Sie arbeiten alleine in einer Druckerei, in der alle Drucker bis auf einen defekt sind. Sie arbeiten auftragsweise und können nur einen Auftrag gleichzeitig bearbeiten. Sie erhalten für die Erledigung der Aufträge unabhängig von der Auftragsgröße eine Pauschale. Ihr Ziel ist also, so viele Aufträge wie möglich zu erledigen.
Für den heutigen Werktag sind bis zum Annahmeschluß Anfragen für Druckaufträge eingetroffen, und Sie müssen nun entscheiden, welche Anfragen Sie annehmen oder ablehnen.
Die Druckdaten werden nicht direkt mit der Anfrage selbst versendet, sondern der Kunde gibt einen Zeitpunkt an, bis zu dem er diese einreicht haben wird (um noch die Gelegenheit zu haben, letzte Änderungen vorzunehmen). Nach Einreichung erwartet der Kunde aber eine schnellstmögliche Bearbeitung, weshalb Sie, wenn Sie einen Auftrag annehmen, sicher sein müssen, daß sie ab diesem Zeitpunkt bis zur Erledigung keinen anderen Auftrag haben.
Aus diesem Umstand ergibt sich für jede Anfrage ein Startzeitpunkt, zu dem der Kunde die Daten abliefert haben wird, und ein Endzeitpunkt, der sich aus dem Umfang des Auftrags ergibt. Falls Sie die Anfrage annehmen, ist folglich Ihr Drucker in der Zwischenzeit komplett für diesen Auftrag reserviert.
Gegenstand dieser Aufgabe ist die automatische Bestimmung einer optimalen Teilmenge an kompatiblen Anfragen mit Hilfe des Greedy-Interval-Scheduling-Algorithmus. In folgendem lückenhaften Listing haben Sie bereits die Struktur und die Anfragen selbst in den Klassen `Anfrage` und `Main` implementiert (wobei die Zahlen bei den Anfragen jeweils für Uhrzeiten stehen). Implementieren Sie die Methoden `ist_kompatibel_mit(Anfrage a)`, die prüft, ob die aktuelle Instanz kompatibel mit der Instanz `a` ist, und `schedule(Anfrage[] A)`, welche den Greedy-Interval-Scheduling-Algorithmus auf ein gegebenes Array `A` von Anfragen anwendet.

```java
public class Anfrage {
	public int s; /* Startzeitpunkt */
	public int f; /* Endzeitpunkt */

	public Anfrage(int s, int f) {
		this.s = s;
		this.f = f;
	}
	public boolean ist_kompatibel_mit(Anfrage a) {...}
	public static Anfrage[] schedule(Anfrage[] A) { ... }
}
```

```java
public class Main {
	public static void print(Anfrage[] A, String name) {
		System.out.println(name);
		for (int i = 0; i < A.length; i++) {
			System.out.printf("(%2d,%2d)", A[i].s, A[i].f);
			if (i + 1 < A.length) {
				System.out.printf(", ");
			}
			/* maximal 5 Anfragen pro Zeile */
			if ((i + 1) % 5 == 0) {
				System.out.printf("\n");
			}
		}
		System.out.println("\n");
	}

	public static void main(String[] args) {
		Anfrage[] A = {
			new Anfrage( 1,  5), new Anfrage( 2,  3), new Anfrage( 3,  4),
			new Anfrage( 4,  6), new Anfrage( 6,  9), new Anfrage( 8, 10),
			new Anfrage( 9, 11), new Anfrage(15, 19), new Anfrage(18, 22),
			new Anfrage(17, 23), new Anfrage(11, 19), new Anfrage(11, 15),
			new Anfrage(17, 19), new Anfrage( 5, 12), new Anfrage( 7, 15),
			new Anfrage(20, 22), new Anfrage( 3,  5), new Anfrage(14, 16),
			new Anfrage(12, 18), new Anfrage(17, 23), new Anfrage( 8, 14),
			new Anfrage( 4,  9), new Anfrage(16, 18), new Anfrage( 7, 14),
		}, S;

		print(A, "Eingegangene Anfragen (A):");
		S = Anfrage.schedule(A);
		print(S, "Angenommene Anfragen (S):");
	}
}
```

Hinweis: Sie können eine Kopie von `A` in `schedule()` erzeugen und Anfragen „entfernen“, indem Sie den dazugehörigen Eintrag im Array auf `null` setzen. Die „Menge“ `A` ist genau dann leer, wenn alle Einträge `null` sind. Sie können auch Ihre Implementierung einer einfach verketteten Liste (nach Anpassung des Datentyps in `Knoten`) aus Aufgabe 10-1 verwenden, die sich auch gut für die „Menge“ `S` anbietet, da sie leicht um ein Element erweiterbar ist.

Präsenzaufgabe 11-3 (Problemerkennung)

Entscheiden Sie, ob es sich bei den folgenden Beispielen jeweils um ein Scheduling-, Partitioning- oder Lateness-Problem handelt.

- Um für das kommende Sommersemester 2020 den Kurt-Alder-Hörsaal mit möglichst vielen Vorlesungen zu belegen und dabei die Möglichkeiten der Dozenten zu berücksichtigen, hat jeder Dozent ein Wunschzeitfenster für seine Vorlesung genannt.
- Als Tierpfleger im Kölner Zoo müssen Sie alle Tiere füttern. Allerdings gab es Probleme bei der Geburt eines Giraffenkalbes, sodaß Sie dort aufgehalten werden. Bei den anderen Tieren gibt es vorgeschriebene Zeiten, wann diese spätestens gefüttert werden sollten. Ihnen ist bereits bewußt, daß Sie nicht alle Zeiten werden einhalten können.
- Als Cheftrainer eines Badminton-Vereins sind Sie für die Planung des Trainings verantwortlich. Dabei bilden die Spieler je nach Alter und Spielstärke unterschiedliche Trainingsgruppen, die aufgrund der zeitlichen Vorgaben der Spieler zum Teil zur selben Zeit stattfinden. Zu Beginn der neuen Saison in wenigen Wochen soll die Anzahl der Trainer optimiert werden. Dabei kann jeder Trainer nur eine Gruppe gleichzeitig trainieren.

Präsenzaufgabe 11-4 (Interval Partitioning)

Sei $A := \{a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9\} := \{(1,4), (2,5), (3,8), (6,15), (7,10), (9,12), (11,18), (13,14), (16,17)\}$ eine Menge von Anfragen (wobei die Zahlen jeweils für Uhrzeiten stehen). Wenden Sie darauf den Greedy-Interval-Partitioning-Algorithmus an. Falls Sie einer Anfrage mehrere Labels zuordnen können, nehmen Sie das Kleinstmögliche.

Präsenzaufgabe 11-5 (Stable Matching)

Sei $M := \{m_1, m_2, m_3\}$ eine Menge von Männern und $W := \{w_1, w_2, w_3\}$ eine Menge von Frauen. Die Männer haben die Präferenzlisten $L_W(m_1) := (w_1, w_2, w_3)$, $L_W(m_2) := (w_1, w_3, w_2)$ und $L_W(m_3) := (w_2, w_3, w_1)$. Die Frauen haben die Präferenzlisten $L_M(w_1) := (m_2, m_1, m_3)$, $L_M(w_2) := (m_1, m_2, m_3)$ und $L_M(w_3) := (m_1, m_2, m_3)$.

0. Erläutern Sie, warum $\{(m_1, w_1), (m_2, w_2), (m_3, w_3)\} \subset M \times W$ ein Matching ist, und prüfen Sie, ob es sich um ein stabiles Matching handelt.
1. Bestimmen Sie ein stabiles Matching mit dem Gale-Shapley-Algorithmus. Jeder Schritt muß eine der drei folgenden Formen für $m \in M$ und $w \in W$ haben:
   - m macht einen Antrag an w
   - m verlobt sich mit w
   - w löst die Verlobung zu m

   Falls in einem Schritt mehrere Männer zur Verfügung stehen, wählen Sie den mit dem kleinsten Index.