Junior-Prof. Dr. Robert Elsässer, Marco Muñiz, Phillip Heidegger WS 2009/2010

Algorithms Theory, Solution for Assignment 6

http://lak.informatik.uni-freiburg.de/lak_teaching/ws09_10/algo0910.php

Exercise 6.1 - Greedy Algorithms

- Use the *IntervalScheduling* algorithm from the lecture, one call for each lecture hall until all the activities are assigned.
- The algorithm is greedy in the sense that it fills the maximum number of activities per hall and thus, uses the minimum number of halls.
- Let S_i be a set of activities, let H be a set of halls (e.g. we denote hall one by h_1), and let A_i be $IntervalScheduling(S_i)$. Then the optimal substructure is:

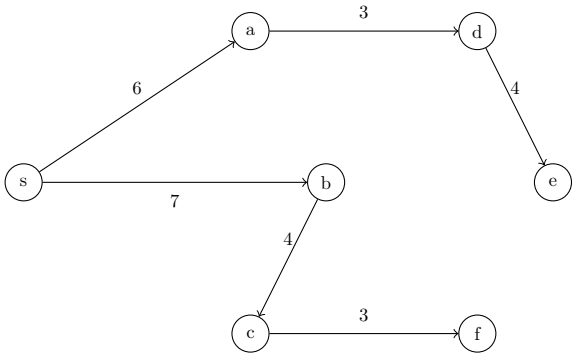
$$HallAss(S_i, H) = HallAss(IntervalScheduling(S_i \setminus A_i), H \setminus \{h_i\}) \cup \{h_i \mapsto A_i\}$$

  Since *IntervalScheduling* has the greedy property, we do not need to examine all the activities in S_i.

Exercise 6.2 - Shortest path

	S	U	u	DIST$[s]$	DIST$[a]$	DIST$[b]$	DIST$[c]$	DIST$[d]$	DIST$[e]$	DIST$[f]$
1	{}	V		0	∞	∞	∞	∞	∞	∞
2	{ s }	$\{a, b, c, d, e, f\}$	s	0	6	7	12	10	∞	∞
3	{s,a}	$\{b, c, d, e, f\}$	a	0	6	7	12	9	∞	∞
4	{s,a,b}	$\{c, d, e, f\}$	b	0	6	7	11	9	15	16
5	{s,a,b,d}	$\{c, e, f\}$	d	0	6	7	11	9	13	16
6	{s,a,b,d,c}	$\{e, f\}$	c	0	6	7	11	9	13	14
6	{s,a,b,d,c,e}	$\{f\}$	e	0	6	7	11	9	13	14
6	{s,a,b,d,c,e,f}	{}	f	0	6	7	11	9	13	14

Rooted tree, root=s:



Exercise 6.3 - Dijkstra's algorithm

The running time of Dijkstra's algorithm, as given in the lecture (slide 17), is

$$\mathcal{O}(n(T_{insert} + T_{empty} + T_{deleteMin}) + m \cdot T_{decreaseKey} + m + n)$$

By using the running times given in the lecture Fibonacci Heaps(slide 4) we have the following:

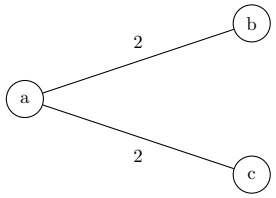
- $\mathcal{O}(n(\mathcal{O}(1) + \mathcal{O}(1) + \mathcal{O}(n)) + m \cdot \mathcal{O}(1) + n + m) = \mathcal{O}(n^2)$
- $\mathcal{O}(n(\mathcal{O}(\log n) + \mathcal{O}(1) + \mathcal{O}(\log n)) + m \cdot \mathcal{O}(\log n) + n + m) = \mathcal{O}((n + m) \log n)$
- $\mathcal{O}(n(\mathcal{O}(\log n) + \mathcal{O}(1) + \mathcal{O}(\log n)) + m \cdot \mathcal{O}(\log n) + n + m) = \mathcal{O}((n + m) \log n)$

Proposed algorithm:

- The algorithm still works. Let u be the remaining element not extracted from the min-priority queue U. If u is not reachable from s, then $\mathsf{DIST}(u) = \mathsf{dist}(s, u) = \infty$. If u is reachable from s, then there is a shortest path $p = s, \ldots, x, u$. We now that when the node x was extracted from the queue U the edge (x, u) was relaxed. Therefore, $\mathsf{DIST}(u) = \mathsf{dist}(s, u)$.

Exercise 6.4 - Minimum spanning trees

Suppose that for every cut of G, there is a unique light edge crossing the cut. Let us consider two minimum spanning trees, T and T', of G. We will show that every edge of T is also in T', which means that T and T', are the same tree and hence there is a unique minimum spanning tree.
Consider any edge $(u, v) \in T$. If we remove (u, v) from T, then T becomes disconnected, resulting in a cut $(S, V \setminus S)$. The edge (u, v) is a light edge crossing the cut $(S, V \setminus S)$. Now consider the edge $(x, y) \in T'$ that crosses $(S, V \setminus S)$. It is also a light edge crossing this cut. Since the light edge crossing $(S, V \setminus S)$ is unique, the edges (u, v) and (x, y) are the same. Thus, $(u, v) \in T'$. Since we chose (u, v) arbitrarily, every edge in T is also in T'.
Below, a counterexample for the converse:



The graph is its own minimum spanning tree, and so the minimum spanning tree is unique. Consider the cut $(\{a\}, \{b, c\})$. Both of the edges (a, b) and (a, c) are light edges crossing the cut.