CSE 417

Algorithms & Computational Complexity

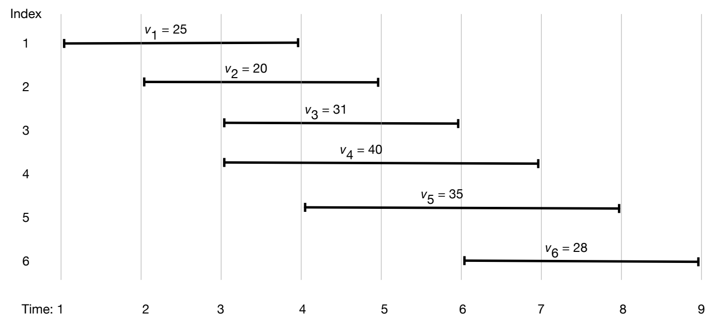
Assignment #6 (rev. a)

Due: Wednesday, 2/27/19

Turnin: Gradescope again; @uw email and gradescope password as before. On-time turn-in deadline is 11PM.

1. [10 points] Consider the instance of the Weighted Interval Scheduling Problem (KT 6.1, p 252) given in the following table and sketched in the figure.

Index	Start	Finish	Value
j	s_j	f_j	v_j
1	1	4	25
2	2	5	20
3	3	6	31
4	3	7	40
5	4	8	35
6	6	9	28



Build a table like that shown on slide 19 of my "DP-Scheduling" slides to illustrate the execution of the dynamic programming algorithm for solving this problem ("Iterative-Compute-Opt", KT 6.2, p259 or my slide 17; *not* the "recursive" or "memoized" versions given earlier in the book/slides). Your table should should repeat j, s_j, f_j, and v_j from above and show, for each $1 \leq j \leq 6$

   (a) The value of $p(j)$, (cf. p 253, slide 10) for this problem instance, as well as

   (b) The "max" expression and resulting $Opt[j]$ values (slide 19).

   In addition:

   (c) *Find* the optimal solution (not just its value) by running the "traceback" algorithm (p 257–8 or slide 20)

   (d) Decorate your table with arrows to illustrate the traceback as in my slides 22–3 (new 2/21; re-download if you have saved older versions).

   (e) Assume we change the problem so that $v_6 = 30$. Explain how, if at all, the table would change, how the Opt value would change, and how the solution and traceback would change.

   (f) **Extra Credit:** Consider the greedy algorithm "process jobs in order of increasing finish time; keep each if it is consistent with previous choices." Generalize this example to show that this greedy algorithm not only may get a suboptimal value, it also may choose many jobs that are not part of any optimal solution.

2. [10 points] Show the OPT table (Figures 6.11, 6.12) that would be produced by the Subset Sum algorithm (KT, Section 6.4, pp 266-271) when run on the sequence of weights $w_1 = 5$, $w_2 = 2$, $w_3 = 4$, $w_4 = 3$, and $w_5 = 6$, with bound $W = 16$. What is the optimum solution found (both its total value and the selected weights)? Summarize the traceback (e.g., in part by overlaying a few arrows on the table) that establishes this optimum solution.

3. [20 points] Here's a variant of the knapsack problem: You are given a knapsack of capacity W, and an *unlimited* supply of each of n kinds of item, where the i-th kind of item has integer weight $w_i > 0$ and value $v_i > 0$, $1 \leq i \leq n$. Give an $O(nW)$ time algorithm to find how many of each item to carry so as to maximize value without exceeding capacity. I.e., find non-negative integers m_i, $1 \leq i \leq n$, maximizing $\sum_{1 \leq i \leq n} m_i v_i$ subject to $\sum_{1 \leq i \leq n} m_i w_i \leq W$. (Note that the best solution might not completely fill the knapsack.)

4. [20 points] KT Chapter 6, problem 1, page 312. (.jpg image) Note that for this problem, the question is to find the optimal *solution*, not just its cost (i.e., include "trace-back"). Also, as always, "give an algorithm" means algorithm, correctness argument and run time analysis.

5. [20 points] KT Chapter 6, problem 2, page 313. (.jpg image)
6. **Extra Credit:** KT Chapter 6, problem 6, page 317. (.jpg image) Assume no word is longer than L. [Side note: this algorithm, invented by Knuth, is a core feature of $\TeX$/$\LaTeX$.]

Revision History:

Rev a:Added Q1, removed “draft” label. — 2/21/19.