Amber Harding

Analysis of Algorithms
Homework 6
Arthur Nunes-Harwitt

1. CLRS 16.1-2

Suppose that instead of always selecting the first activity to finish, we instead select the last activity to start that is compatible with all previously selected activities. Describe how this approach is a greedy algorithm, and prove that it yields an optimal solution.

This approach is a greedy algorithm because at each individual step the optimal choice is selected. In this case the optimal choice is the activity that has the latest start time. In the original problem activities are selected in ascending order of start time. Here they are selected in descending order of start time. Therefore this approach is just a reversal of the original and will also yield an optimal solution.

2. CLRS 16.1-3

Not just any greedy approach to the activity-selection problem produces a maximum-size set of mutually compatible activities. Give an example to show that the approach of selecting the activity of least duration from among those that are compatible with previously selected activities does not work.
Do the same for the approaches of always selecting the compatible activity that overlaps the fewest other remaining activities and always selecting the compatible remaining activity with the earliest start time.

Selecting the least duration:

Activity	Starting Time	Finishing Time	Duration

1	3	6	3
2	1	5	4
3	5	9	4

In this example the optimal solution should be {2, 3, 1}, but by selecting Activity 1 because it has the least duration would result in {1} which is not optimal.

Selecting the compatible activity that overlaps the fewest other remaining activities:

Activity	Starting Time	Finishing Time	# of overlaps
1	-1	1	1
2	2	5	2
3	0	3	2
4	4	6	2
5	5	8	2
6	7	9	2
7	8	10	1
8	9	10	1
9	10	11	0

By selecting activity 1 first the solution is{1, 2, 5, 7} which is not optimal. The solution {3, 4, 6, 8, 9}is optimal because the most activities are able to be performed here.

Selecting earliest start time:

Activity	Starting Time	Finishing Time

1	0	6
2	1	5
3	5	9

Selecting 1 first gives the solution {1} which is not optimal because 2 and 3 are unable to be started. The optimal solution here is {2, 3}.

3. Consider the following problem.

Problem 1 (GRAPHISOMORPHISM). Given hG_1, G_2i, where G_1 and G_2 are graphs, are G_1 and G_2 isomorphic?
Prove that GRAPHISOMORPHISM $\in$ NP by showing that it can be verified in polynomial time. To do this you need to exhibit the verification algorithm.

Let input = G1 and G2. Let the certificate Y be the indices [(i1, i2, …, in)]. An algorithm A(x, y) verifies Graphisomorphism by checking whether or not Y is a permutation, if no return false. If Graph G1 and G2's indices map identically while preserving elements relationships.

4. Prove that if NP != coNP then P != NP.

Prove the contrapositive in order to prove the original statement true.

Assume: P = NP

Let L $\in$ $CO - NP$
We know $\overline{L} \in$ NP, but because NP = P. We conclude that $\overline{L} \in$ P
We know L = $\overline{L} \in$ P Thus CO - NP $\subseteq$ P and we know that P $\subseteq$ CO - NP thus P = CO - NP, and since P = NP we conclude that NP = CO - NP.

5. Let $\psi = ((x_1 \vee x_2) \wedge x_3) \wedge ((x_1 \wedge x_2 \wedge \bar{x}_3) \vee x_3) \wedge (x_1 \wedge x_2 \wedge \bar{x}_3)$.
Verify that ψ is not satisfiable.

$\psi = ((x_1 \vee x_2) \wedge x_3) \wedge ((x_1 \wedge x_2 \wedge \bar{x}_3) \vee x_3) \wedge (x_1 \wedge x_2 \wedge \bar{x}_3)$.

x1	x2	x3	=
T	T	T	F
T	T	F	F
T	F	T	F
F	T	T	F
T	F	F	F
F	T	F	F
F	F	T	F
F	F	F	F

All combinations of truth values result in a false solution. Therefore Ψ is not satisfiable.

6. Show that the problem of determining the satisfiability of propositional formulas in disjunctive normal form is polynomial time solvable.

CNF	DNF
$(x_{1 \text{ V } x2 \text{ V } x3})$ 1 clause	(x_1) V (x2) V (x3) 3 clauses
(x1 V x2 V x3) $\wedge$ (x4 V x5 V x6) 2 clauses	(x1 $\wedge$ x4) V (x2 $\wedge$ x4) V (x3 $\wedge$ x4) V (x1 $\wedge$ x5)V(x2 $\wedge$ x5) V(x3 $\wedge$ x5) V(x1 $\wedge$ x6) V (x2 $\wedge$ x6) V (x3 $\wedge$ x6)

	9 clauses
(x1 V x2 V x3) ∧ (x4 V x5 V x6) ∧ (x7 V x8 V x9) 3 clauses	(x1 ∧ x4 ∧ x7) V (x1 ∧ x4 ∧ x8) V .....V(x3 ∧ x6 ∧ x9) 27 clauses

7. Recall the 0-1 knapsack problem in CLRS chapter 16.

(a) What is the time complexity of the dynamic programming based algorithm?

Time complexity = O(nW)
Where n = number of items, and W = capacity of knapsack.

(b) The knapsack decision problem is NP-complete. Does your analysis above prove that P= NP? Explain. 1

Solving the Knapsack problem is the same as solving the Subset-Sum Problem which is proven to be NP complete. The Subset-Sum Problem is to find a subset of elements that are selected from a given set whose sum adds up to a given number K. We are considering the set containing non-negative values. It is assumed that the input set is unique (no duplicates are presented). The knapsack problem is solved by verifying that the collective sum of the weight of the items in the knapsack do not exceed the capacity. When inputs are binary the complexity becomes exponential. Therefore this problem is NP complete.