Analysis of Algorithms

Homework 6

Thomas Schollenberger (tss2344)

December 2, 2022

Question: 1

CLRS 15.1 – 2
Suppose that instead of always selecting the first activity to finish, you instead select the last activity to start that is compatible with all previous selected activities. Describe how this approach is a greedy algorithm, and prove that it yields an optimal solution.

Answer to 1: This problem is the same as the activity selection problem, but with the difference that we select the last activity. This means that the greedy algorithm that we implement, which just finds the longest activity start time from the previous activity's end time, is functionally equivalent to the original activity selection problem, just preformed in reverse. This means that we implicitly know that it yields an optimal solution. It is greedy because we find the best case each step. □

Question: 2

CLRS 15.1 – 3
Not just any greedy approach to the activity-selection problem produces a maximal 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.

Answer to 2: For the first case:
If we have three durations, two which are long and one which is short, and we select the shortest one, we will have only one solution, when the correct choice is two.
An example of this is $\{(1,5),(6,7),(8,15)\}$.

For the second case:
This greedy algorithm breaks when you have multiple activities with the same start and end times.
An example of this is $\{(1,5),(1,5),(1,5),(2,3),(2,3),(3,5),(3,5)(4,6),(5,9),(5,9)\}$.
The algorithm would pick $\{(4,6)\}$ as the answer, but the correct answer is $\{(1,5),(5,9)\}$.

For the third case:
This greedy algorithm breaks when you have the earliest start time also be extremely long.
An example of this is $\{(1,7),(2,3),(4,7)\}$.
The algorithm would pick $\{(1,7)\}$ as the answer, but the correct answer is $\{(2,3),(4,7)\}$.

□

Question: 3

(GRAPHISOMORPHISM). Given $\langle G1, G2\rangle$, where $G1$ and $G2$ are graphs, are $G1$ and $G2$ 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.

Proof of 3: We can verify that $G1$ and $G2$ are isomorphic by checking that they have the same number of vertices and edges, and that they have the same degree sequence. This algorithm runs in polynomial time because it is a simple linear search through each graph. □

Question: 4

The class coNP is defined as the class of problems such that Q $\in$ coNP if and only if Q $\in$ NP. Prove that if NP $\neq$ coNP then P $\neq$ NP. (HINT: Try proving the contrapositive.)

Proof of 4: Contrapositive: If P = NP then NP = coNP If we were to assume that P is equal to NP, then an algorithm in P would have it's complement in coNP. This then contradicts the statement that NP = coNP, because the nature of the complement means that it is outside of the set P, which is equal to NP. Thus, the contrapositive is proven to be false, proving that NP $\neq$ coNP then P $\neq$ NP. □

Question: 5

Let $\Psi = ((x_1 \vee x_2) \wedge x_3) \wedge ((x_1 \wedge x_2 \wedge \overline{x_3}) \vee x_3) \wedge (x_1 \wedge x_2 \wedge \overline{x_3})$.
Verify that Ψ is not satisfiable

Verification of 5: Verification table showing that for all values, Ψ is false:

x_1	x_2	x_3	$((x_1 \vee x_2) \wedge x_3) \wedge ((x_1 \wedge x_2 \wedge \overline{x_3}) \vee x_3) \wedge (x_1 \wedge x_2 \wedge \overline{x_3})$
T	T	T	F
T	T	F	F
T	F	T	F
T	F	F	F
F	T	T	F
F	T	F	F
F	F	T	F
F	F	F	F

□

Question: 6

Consider the definition of *disjunctive normal form.*
Definition: *A propositional normal formula ψ is in disjunctive normal form if ψ has the form $\phi = \varphi_1 \vee \cdots \vee \varphi_n$, where $\varphi_i = \ell_1^i \wedge \cdots \wedge \ell_{m_i}^i$ and ℓ_i^j , a literal, is either a variable or the negation of a variable.*
Show that the problem of determining the satisfiability of propositional formulas in disjunctive normal form is polynomial time solvable.

Answer to 6: Fundamentally, each literally is just a boolean value, either a 1 or a 0. This means that each φ_i can be calculated in polynomial time. This means that the entire formula can be calculated in polynomial time, making it polynomial time solvable. In most cases, this would actually be $O(1)$. □

Question: 7

Recall the 0-1 knapsack problem.
a. What is the time complexity of the dynamic programming based algorithm? b. The knapsack decision problem is NP-complete. Does your analysis above prove that P = NP? Explain.

Answer to 7a: The time complexity of the dynamic programming based algorithm is $O(nW)$, where n is the number of items and W is the maximum weight. □

Answer to 7b: My above analysis, showing that the time complexity of the problem is in polynomial time, would only prove P = NP if and only if all NP problems could be reduced to the 0-1 knapsack problem. □