Name: Nam Huynh

ID: nsh1507

1.

a.

If we assume that the activities are sorted in descending order of finish time (i.e., f1 ≥ f2 … ≥ fn), this problem becomes the same as the original one with the finishing time being reversed, so it produces an optimal solution for the same reasons.

New Pseudocode:

count(s,f,k,m,n) =	0	if m > n
count(s,f,k,m,n) =	count(s, f, k, m+1, n)	if s[k] < f[m]
count(s,f,k,m,n) =	1+count(s, f, m, m+1, n)	otherwise

b.

Counterexample selecting the least duration:

Suppose our activity times are {(1,5),(3,6),(5,10)}. If we decide to pick the shortest first, we have to eliminate the (1,5) and (5, 10) and our list would be {(3,6)}, resulting in a count of 1, while the optimal is {(1,5), (5, 10)}.

Counterexample selecting the task with fewest overlaps:

Suppose our activity times are {(0,3),(3,5),(5,7),(7,9), (0,2), }. If we decide to pick the activity with least conflict, our list would be {}, while the optimal is {(0,3),(3,4),(4,5),(5,7)}..

Counterexample to selecting the earliest start times:

Suppose our activity times are {(1,100),(2,4),(4,6)}. If we pick the earliest start time, we will only have a single activity, (1,100), whereas the optimal solution would be to pick the two other activities.

2.

a.

Let $\psi = ((x_1 \vee x_2) \wedge x_3) \wedge ((x_1 \wedge x_2 \wedge \neg x_3) \vee x_3) \wedge (x_1 \wedge x_2 \wedge \neg x_3)$

Suppose x_1 = True and x_3 = False,

Then the statement $((x_1 \vee x_2) \wedge x_3)$ is False since the statement $(x_1 \vee x_2)$ evaluates to True and True $\wedge$ False (x_3) evaluates to False. Therefore ψ is not satisfiable.

b.

A DNF propositional formula ψ is satisfiable **iff** there is at least one clause in ψ that is satisfiable. To check if a clause is satisfiable, we only need to do one pass of the propositional argument to verify that whether or not a literal and its negation does not appear in the clause. If a literal and its negation both exist within the formula, our algorithm will return a False, otherwise True. Since the algorithm works in a single pass, the running time of the algorithm is O(n)

c.

Given two graphs G1 and G2 and a certificate as input to the verifier, the verifier algorithm could go over each vertex of G1 and compare if the neighbors of vertex in G1 correspond to the neighbors in G2 . If not, reject.

d.

Prove by contrapositive:

Assume P = NP then:

- For every language in NP, we also have that language in P, and since the languages in NP are closed under complement, then $\overline{L} \in NP$, therefore $L \in$ CoNP, since CoNP consists of the complements of all problems in NP.
- For every language in CoNP, $\overline{L} \in NP$ since CoNP consists of the complements of all problems in NP. We also have $\overline{L} \in P$ since P = NP by assumption, and since the languages in P are closed under complement, $L \in$ P, therefore $L \in$ NP.
- Since NP ⊆ CoNP and CoNP ⊆ NP, NP = CoNP

e.

Prove that the relation $\leq_p$ is reflexive:

Suppose that there exists a decision problem Q, to prove the relation $\leq_p$ is reflexive is to say that $Q \leq_p Q$, or Q is reducible to Q. Assume there exists a reduction function called C that takes a Q input and returns a Q output, such a function has a time complexity of **O(1)** since it only returns the input. Since we can make a reduction function C that runs in a polynomial time, the relation $\leq_p$ is reflexive.

f.

Prove that the relation $\leq_p$ is transitive:

Suppose that there exists the decision problems Q, R, and S. Assume that $Q \leq_p R$ *and* $R \leq_p S$, to prove the relation $\leq_p$ is transitive is to say that Q is also reducible to S, $Q \leq_p S$. Since $Q \leq_p R$, it is by definition that the reduction function to reduce R to Q is ran in polynomial time, and the same holds for $R \leq_p S$, we can combines the reduction functions of these two relations $Q \leq_p R$ and $R \leq_p S$ together to produce a new reduction function that converts the Q input to S output, Q $\leq_p S$, that runs in polynomial time. Since $Q \leq_p R$ and $R \leq_p S$ implies that $Q \leq_p S$, the relation $\leq_p$ is transitive.

3.

a.

The time complexity of the algorithm is **O(nW)**, where n is the number of items and W is the capacity of the knapsack

b.

The analysis does not prove that P = NP since there are no algorithms that can solve this problem in polynomial time.