Homework 6

Foundations of Algorithms

Divesh Badod

1. This approach is essentially same as the previous approach with just reversed order of the activities and this solution is again a greedy solution because we make best looking choice for each step.

2. Consider the example of set {(1, 4),(5, 7),(6, 11)}
   Here if we go with the approach of selecting the shortest activity then we will only get to select one activity out of the three and the rest two are not considered but with the traditional greedy approach we get two activities done.
   Consider the example of set {(−1, 1),(2, 4),(3, 6),(0, 3),(0, 3),(5, 7),(9, 12)}
   Here if we go with the approach of selecting an activity of minimum conflicts then we would only get the activity (3, 6) done because it has only two conflicts instead of getting the optimal solution which has (-1, 1),(2, 4),(5, 7),(9, 12) and all of these activities won't be done.
   Consider the example of set {(1, 10),(2, 3),(4, 5)}
   Here if we go with the approach of the selecting the activities with the earliest start time then only first activity will be get done whereas the better solution gives us the other two as the answer.

3. Suppose the certificate y is used to verify the language is in the form of lists v_i for G_1 and u_i in G_2 where (i = 1,2…,n) and v_i corresponds to u_i. By using DFS we explore the edges of G_1. If there is an edge between (v_i, v_j) then we check in G_2 if there is an edge between (u_i, u_j). if there is an edge then we move on otherwise the graphs aren't isomorphic and we return FALSE. After doing so with G_1 we do it again we G_2 now if every edge in one graph has a corresponding edge in the other graphs then only we return TRUE because the graphs are isomorphic. Since this entire algorithm takes $O(n^2)$ time hence GRAPH-ISOMORPHISM Є NP(n is the number of vertices)\

4. Suppose $NP \neq co - NP$.
   Let $L \in NP \backslash co - NP$. Since $P \subset NP \cap co - NP$, and $L \notin NP \cap co - NP$
   We get $L \notin P$
   Thus $P \neq NP$.

5. For all the values of x1 x2 and x3
   x1-F x2-F x3-F we get ψ-F
   x1-F x2-F x3-T we get ψ-F
   x1-F x2-T x3-F we get ψ-F
   x1-T x2-F x3-F we get ψ-F
   x1-F x2-T x3-T we get ψ-F
   x1-T x2-T x3-F we get ψ-F
   x1-T x2-F x3-T we get ψ-F
   x1-T x2-T x3-T we get ψ-F
   Since in all the cases ψ is False. Hence ψ is not satisfiable.

6. Since the problem is in disjunctive normal form we have 'and' clauses of bunch of variables $\wedge Q_i (Q_i\ stands\ of\ and\ of\ a\ bunch\ of\ vaeiables)$. Now in an 'and' clause if there is a variable present with its negation like $Q_i \wedge \sim Q_i$ then this will return False and it is clearly not satisfiable. However if each variable doesn't have their negation then we can just pick the appropriate value to assign to each variable. Assuming there are 'm' clauses and 'n' variables then each clause can be evaluated O(n^2) times and the time complexity for entire algorithm is O(mn^2). Hence this problem is polynomial-time solvable.
7.
   a. Time complexity of the dynamic programming-based algorithm is O(nW), where n is the number of items and W is the capacity of the knapsack.
   b. With the above given time complexity it does not verify in polynomial-time because it increases linearly with size of input 'n' and exponentially with the size of W because W is basically the number of bits to store in that place and with every bit it increases exponentially. This algorithm is rather pseudo-polynomial. Hence the complexity doesn't say that knapsack problem is NP-complete, and the above analysis does not prove that P=NP.

8. We can guess that two partitions have equal sums
   SUBSET-SUM is defined as given a set of integers S and a target number t, find a subset such that the members of that subset has a sum equal to t. Let s be the sum of members of S, the S = S ∪ { s – 2t } into PARTITION.
   If PARTITION accepts then the reduction works. We have to prove (S, t) ∈ SUBSET-SUM if and only if S` ∈ PARTITION sum of members of S is 2s-2t. If a set of numbers in S sum to t then the remaining sum to s-t. Hence there exists a partition of S such that each of the two partitions of S adds to s-t. Let's say there exists a partition of S such that the sum of members in each partition adds to s-t. One of these partitions contains number s-2t. By removing this number we get a set whose sum equals t, and all the numbers present in S. Since the SUBSET-SUM problem is NP-complete, this proves that the set-partition problem is NP-Hard. See that it is NP, just let the certificate be the set of elements of S that forms one side of the partition. It is linear time to add and make sure that they are exactly half the sum of all the elements in S.