CS 600 Advanced Algorithms

Sri Dhanush Reddy Kondapalli

CWID: 10476150

Homework 9

1 R-17.8.8 Given the CNF formula $B = (x_1).(\overline{x_2} + x_3 + x_5 + \overline{x_6}).(x_1 + x_4).(x_3 + \overline{x_5})$, show the reduction of B into an equivalent input for the 3SAT problem.

$B = (x_1 + a + b).(x_1 + \overline{a} + b).(x_1 + a + \overline{b}).(x_1 + \overline{a} + \overline{b}).(\overline{x_2} + x_3 + c).(\overline{c} + x_5 + \overline{x_6}).(x_1 + x_4 + d).(x_1 + x_4 + \overline{d}).(x_3 + \overline{x_5} + e).(x_3 + \overline{x_5} + \overline{e})$

2 R-17.8.12 Professor Amongus has just designed an algorithm that can take any graph G with n vertices and determine in $O(n^k)$ time whether G contains a clique of size k. Does Professor Amongus deserve the Turing Award for having just shown that $P = NP$? Why or why not?

The clique problem is NP complete. Reduction from vertex-cover.
Input: G= (V, E) and integer k>0

1. Non-deterministically select a subset C of nodes of G.
2. Check to see whether all nodes in c are linked and if G has all edges linking nodes in C.
3. If yes, accept.
4. Else reject.

To demonstrate $P = NP$, every issue in NP must be solved in polynomial time. It must be solvable in polynomial time for any value of k. The value of k is not

specified.

No, Professor Amongus does not deserve the Turing Award for demonstrating that $P = NP$ because there are infinitely many problems in NP and justifying polynomial time for each problem is realistically impossible.

3 C-17.8.28 Define HYPER-COMMUNITY to be the problem that takes a collection of n web pages and an integer k, and determines if there are k web pages that all contain hyperlinks to each other. Show that HYPER-COMMUNITY is NP-complete.

HYPER-COMMUNITY is in NP, since a non-deterministic machine simple theory "k" website pages and ensure that they are all related with one other.

We reduce from the autonomous set to show that hyper-networks are NP-hard. Assume we have Graph G with n verices and we need to find an independent collection of size k. We build G' on the same n vertices as G, where (v, w) is an edge in G' if and only if it is not an edge in G. Because we only need to iterate over all pairs of vertices, this reduction obviously takes polynomial time.

Currently, if G' contains a large number of "k" mutually connected vertices, they must be from an Independent Set in G. If, on the other hand, G contains an Independent Set of size k, then those k vertices should all be associated in G'.

Because Independent Set becomes Hyper-Community, Hyper-Community must be NP-Complete.

4 A-17.8.35 Show that the decision version of the problem Dr. Drama has asked you to solve is NP-complete.

The problem described above is a vertex cover problem. A graph G(V, E) and a positive integer k are examples of vertex cover problems. Now check it is NP-complete or not.
For a problem to be NP-complete, it must satisfy the two condition:
1. Proof that the problem is in NP
2. Proof that the problem is NP-Hard

Proof that the problem is in NP.

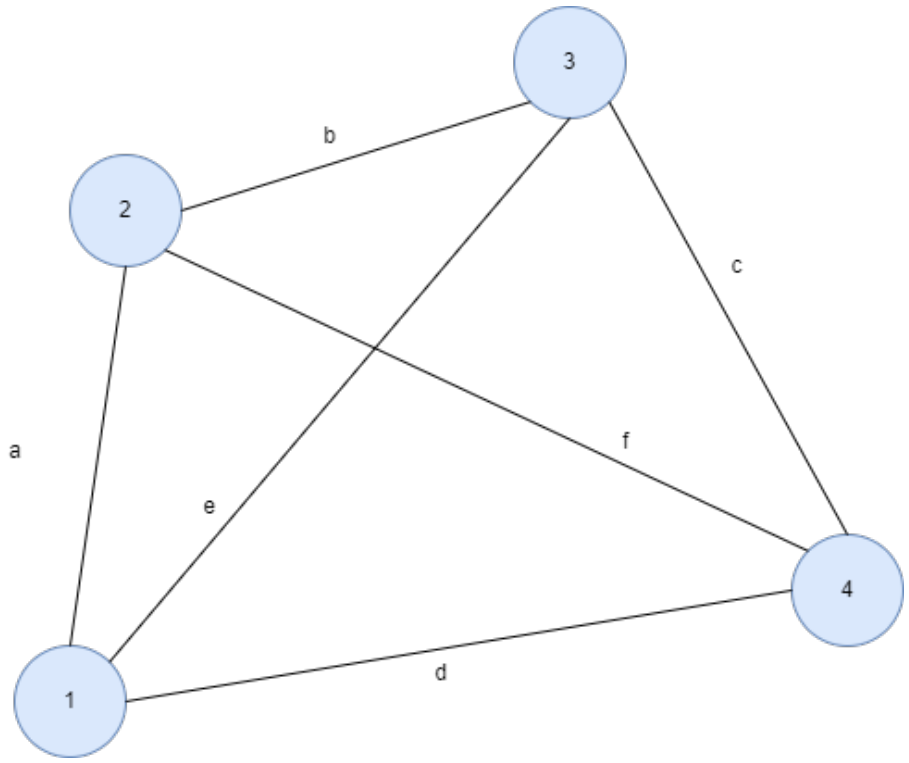
1. Choose a non-deterministic subset W of size k.
2. Remove all edges adjacent to v from set X for each vertex v in W. If the number of edges removed equals k, the set X becomes empty.
3. If yes, accept.
4. Else reject.

As above algorithm can be done in polynomial time thus it is in NP.
Proof that the problem is NP-Hard.

1. Reduce 3SAT to VERTEX-COVER for NP-Hard.
2. Let S be the Boolean formula in CNF, with three literals in each clause.
3. Make a graph G and an integer k such that G has a vertex cover if and only if S is satisfied.
4. Vertex cover must include one of the two vertices, according to the "truth setting" and "satisfaction setting" components.

According to theorem 17.9 given in the textbook
Vertex-Cover is NP complete.

5 C-18.6.19 Show that an optimal solution to the Euclidean TSP is a simple polygon, that is, a connected sequence of line segments such that no two ever cross.



The Euclidean TSP would look like this
and according to traingle law
a + b <= e – (1)
b + c <= f – (2)
c + d <= e – (3)
a + d <= f – (4)

Now consider the TSP paths 1 – 2 – 3 – 4 – 2 and the path 2 – 3 – 1 – 4 – 2.
We will choose path 2 - 3 - 1 - 4 - 2 based on the best path solution, where edges e and f are only of interest if
b + e + f + d <= a + b + c + d
e + f <= a + c
a + 2b + c <= e + f <= a + c by adding (1) and (2)
a + 2b + c <= a + c

2b <= 0
b <= 0

Which is only feasible if 2 and 3 are the same vertices or if the euclidean distance assumption of distance >= 0 is violated, both of which are not possible. As a result, the best solution to the Euclidean TSP is a simple polygon, which is a linked sequence of line segments that never cross.

6 A-18.6.26 Describe a simple greedy algorithm for assigning boxes to trucks and show that your algorithm uses a number of trucks that is within a factor of 2 of the optimal number of trucks. You may assume that no box weighs more than M pounds.

Consider a greedy algorithm that fills trucks one by one. Consider boxes to be "large" if they weigh more than M/2 pounds and "small" otherwise to demonstrate that this is a 2-approximation procedure. In both the greedy and optimum solutions, each large box will be transported in a distinct vehicle. Let G_B represent the number of trucks in the greedy solution with large boxes and H_B represent the number of trucks in the optimum solution with big boxes. Then $G_B = H_B$.

Consider the amount of trucks that exclusively carry little cargo. According to the greedy answer, every vehicle packed with just little boxes must have a weight of at least M/2. We could have placed one more tiny package inside the vehicle. Let G_S be the number of vehicles in the greedy solution that only contains small boxes and H_S be the number of trucks in the optimum solution that only contains small boxes. It should be noted that the optimal method may be able to put certain tiny boxes into vehicles that also hold a large box, but the greedy solution may not.
$G_S \leq 2H_S + H_B$.
In other words, the greedy solution's truck count, $G_S + G_B$, satisfies the following conditions:
$G_S + G_B \leq 2H_S + 2H_B = 2(H_S + H_B)$.
Therefore, this greedy solution is a 2-approximation algorithm.