Assignment 9

35- 1.1
Give an example of a graph for which APPROX-VERTEX-COVER always yields a suboptimal solution.

We could select the graph that consists of only two vertices and a single edge between them. Then, the approximation algorithm will always select both of the vertices, whereas the minimum vertex cover is only one vertex. more generally, we could pick our graph to be k edges on a graph with 2k vertices so that each connected component only has a single edge. In these examples, we will have that the approximate solution is off by a factor of two from the exact one.

35 -2.2
Show how in polynomial time we can transform one instance of the traveling-salesman problem into another instance whose cost function satisfies the triangle inequality. The two instances must have the same set of optimal tours. Explain why such a polynomial-time transformation does not contradict Theorem 35.3, assuming that P ¤ NP.

Let m be some value which is larger than any edge weight. If c(u, v) denotes the cost of the edge from u to v, then modify the weight of the edge to be H(u, v) = c(u, v) + m. (In other words, H is our new cost function). First we show that this forces the triangle inequality to hold. Let u, v, and w be vertices.

Then we have H(u, v) = c(u, v) + m ≤ 2m ≤ c(u, w) + m + c(w, v) + m = H(u, w) + H(w, v). Note: it's important that the weights of edges are nonnegative. Next we must consider how this affects the weight of an optimal tour. First, any optimal tour has exactly n edges (assuming that the input
graph has exactly n vertices).

Thus, the total cost added to any tour in the original setup is nm. Since the change in cost is constant for every tour, the set of optimal tours must remain the same. To see why this doesn't contradict Theorem 35.3, let H denote the cost of a solution to the transformed problem found by a ρ-approximation algorithm and H∗ denote the cost of an optimal solution to the transformed problem. Then we have H ≤ ρH∗ .

We can now transform back to the original problem, and we have a solution of weight C = H − nm, and the optimal solution has weight C ∗ = H∗ − nm. This tells us that C + nm ≤ ρ(C ∗ + nm),so that C ≤ ρ(C ∗) + (ρ − 1)nm. Since ρ > 1, we don't have a constant approximation ratio, so there is no contradiction.

35- 3.1

Consider each of the following words as a set of letters: farid; dash; drain; heard; lost; nose; shun; slate; snare; threading. Show which set cover GREEDY-SET-COVER produces when we break ties in favor of the word that appears first in the dictionary.

Since all of the words have no repeated letters, the first word selected will be the one that appears earliest on among those with the most letters, this is “thread”. Now, we look among the words that are left, seeing how many letters that aren’t already covered that they contain. Since “lost” has four letters that have not been mentioned yet,and it is first among those that do, that is the next one we select. The next one we pick is “drain” because it has two unmentioned letters. This only leaves “shun” having any unmentioned letters, so we pick that, completing our set. So, the final set of words in our cover is {thread, lost, drain, shun}

35 -3.2
Show that the decision version of the set-covering problem is NP-complete by reducing it from the vertex-cover problem.

A certificate for the set-covering problem is a list of which sets to use in the covering, and we can check in polynomial time that every element of X is in at least one of these sets, and that the number of sets doesn’t exceed k. Thus, set-covering is in NP. Now we’ll show it’s NP-hard by reducing it from the vertex-cover problem. Suppose we are given a graph G = (V, E). Let V 0 denote the set of vertices of G with all vertices of degree 0 removed. To each vertex $v \in V$, associate a set Sv which consists of v and all of its neighbors, and let F = {Sv|v $\in$ V 0}. If $S \subset V$ is a vertex cover of G with at most k vertices, then {Sv|v $\in$ S$\cap$V 0} is a set cover of V 0 with at most k sets. On the other hand, suppose there doesn’t exist a vertex cover of G with at most k vertices. If there were a set cover M of V 0 with at most k sets from F, then we could use {v|Sv $\in$ M} as a k-element vertex cover of G, a contradiction. Thus, G has a k element vertex cover if and only if there is a k-element set cover of V 0 using elements in F. Therefore set-covering is NP-hard, so we conclude that it is NP-complete.

35- 4.3
In the MAX-CUT problem, we are given an unweighted undirected graph G D .V; E/. We define a cut .S; V S / as in Chapter 23 and the weight of a cut as the number of edges crossing the cut. The goal is to find a cut of maximum weight. Suppose that for each vertex , we randomly and independently place in S with probability 1=2 and in V S with probability 1=2. Show that this algorithm is a randomized 2-approximation algorithm.

Let's first recall the algorithm:

```
APPROX-MAX-CUT(G)
   for each v in V
      flip a fair coin
      if heads
         add v to S
      else
         add v to V – S
```

This algorithm clearly runs in linear time. For each edge (u,v)∈E, define the event Auv to be the event where edge (i,j) crosses the cut (S,V−S), and let 1Auv be the indicator random variable for Auv.

The event Aij occurs if and only if the vertices u and v are placed in different sets during the main loop in APPROX-MAX-CUTAPPROX-MAX-CUT. Hence,

$$\begin{aligned}\mathbf{P}\{A_{uv}\} &= \mathbf{P}\{u \in S \wedge v \in V - S\} + P\{u \in V - S \wedge v \in S\} \\ &= \frac{1}{2} \cdot \frac{1}{2} + \frac{1}{2} \cdot \frac{1}{2} \\ &= \frac{1}{2}.\end{aligned}$$

Let opt opt denote the cost of a maximum cut in G, and let |c=|(S,V−S)|, that is, the size of the cut produced by APPROX-MAX-CUTAPPROX-MAX-CUT. Clearly c=∑(u,v)∈E1Auv. Also, note that opt≤|E| (this is tight if G is bipartite). Hence,

$$\begin{aligned}\mathbf{E}[c] &= \mathbf{E}\left[\sum_{(u,v)\in E} 1_{A_{uv}}\right] \\ &= \sum_{(u,v)\in E} \mathbf{E}[1_{A_{uv}}] \\ &= \sum_{(u,v)\in E} \mathbf{P}\{A_{uv}\} \\ &= \frac{1}{2}|E| \\ &\geq \frac{1}{2}\mathtt{opt}\end{aligned}$$

Hence, $E[opt/C] \leq 1/2 |E| |E| = 2$, and so APPROX-MAX-CUTAPPROX-MAX-CUT is a randomized 2-approximation algorithm.