Choose 6 out of these 11 problems. Write/type/scan your answer and submit/email it to me before Feb 2, 23:59. You can discuss your problems with your classmates but you have to do it yourself.

1. Is the following an **NP**-optimization problem? Given an undirected graph $G = (V,E)$, a cost function on vertices $c : V \rightarrow \mathbf{Q}_+$, and a positive integer k, find a minimum cost vertex cover for G containing at most k vertices.

   Hint: Can valid instances be recognized in polynomial time (such an instance must have at least one feasible solution)?

2. Let A be an algorithm for a minimization **NP**-optimization problem Π such that the expected cost of the solution produced by A is $\leq \alpha\text{OPT}$, for a constant $\alpha > 1$. What is the best approximation guarantee you can establish for Π using algorithm A?

   Hint: A guarantee of $2\alpha - 1$ follows easily. For guarantees arbitrarily close to α, run the algorithm polynomially many times and pick the best solution. Apply Chernoff's bound.

3. For the set cover problem, let us define frequency of an item to be the number of sets that contain the item and let f be the frequency of the most frequent item. Let 2SC denote the restriction of set cover to instances having $f = 2$. Show that 2SC is equivalent to the vertex cover problem, with arbitrary costs, under approximation factor preserving reductions.

4. The hardness of the Steiner tree problem lies in determining the optimal subset of Steiner vertices that need to be included in the tree. Show this by proving that if this set is provided, then the optimal Steiner tree can be computed in polynomial time.

5. Let G be a complete undirected graph in which all edge lengths are either 1 or 2 (clearly, G satisfies the triangle inequality). Give a 4/3 factor algorithm for TSP in this special class of graphs.

   Hint: Start by finding a minimum 2-matching in G. A 2-matching is a subset S of edges so that every vertex has exactly 2 edges of S incident at it.

6. Consider the greedy algorithm for the knapsack problem. Sort the objects by decreasing ratio of profit to size, and then greedily pick objects in this order. Show that this algorithm can be made to perform arbitrarily badly.

7. Show that a strongly **NP**-hard problem cannot have a pseudo-polynomial time algorithm, assuming $\mathbf{P} \neq \mathbf{NP}$.

8. For the bin packing problem, give an example on which First-Fit does at least as bad as 5/3 OPT.

9. In the minimum Makespan Scheduling, we have a very simple factor 2 approximation algorithm simply by scheduling a job to the machine having the least amount of work so far. We have learned that the tight bound of this algorithm when we have m*m jobs of a 1-unit time and one another job of m-unit time.

   This suggests sorting the jobs by decreasing processing times before scheduling them. Show that this leads to a 4/3 factor algorithm. Provide a tight example for this algorithm.

10. For Euclidian TSP problem, show that we may assume that the length of the bounding square can be taken to be L = 4n2 and that there is a unit grid defined on the square such that each point lies on a gridpoint.

    Hint: Since we started with the smallest axis-parallel bounding square, its length is a lower bound on OPT. Therefore, moving each point to a grid point can increase the length of the tour by at most OPT/n2.

11. Consider the max-flow (min-cut) problem. Show that each extreme point solution for the LP of the min-cut problem is 0/1, and hence represents a valid cut.