CS 530 - Fall 2018
Quiz 3

Instructions: Please answer any 3 of the following 4 questions. Each question counts equally. Write your answers on the Answersheet. If need be, you can also use 1 piece of paper for longer answers. Put an X over the number of the problem on the answersheet which you do not want graded.

1. (**Bin Packing.)** Consider the following bin packing problem.

   There are 7 objects $o_1, ..., o_7$ with sizes .2, .5, .4, .7, .1, .3, .8 and the goal is to pack the objects into as few bins of size 1 as possible.

   (i). Show the result of the bin packing when the First Fit (FF) heuristic is carried out.

   (ii). What is the optimal bin packing for this problem?

   (iii). Show the result of the bin packing when the Best Fit decreasing (BFD) heuristic is carried out.

   (iv). Show the result of the bin packing when the Next Fit (FF) heuristic is carried out.

   (v). Which of the three heuristic methods gives the worst approximation ratio? What is that ratio ?

2. (**Traveling Salesman Problem)** The problem here is to give an example of a complete weighted but NOT Euclidean graph G with 4 vertices which has the property that when you run the 2-approximation for the of the traveling salesman problem on the graph you obtain a TSP solution whose value is more than 5 times the optimal TSP solution for that graph.

   (i). Draw your example weighted graph G.
   (ii). State why G is not Euclidean.
   (iii). State what the optimal solution for G is, and also show what solution the approximation algorithm results in. (Drawing the 2 graphs here are fine.)

3. (**Set Cover. Multiple choice - no partial credit.)** Set cover problem for each of A,B,C,D. (2 points each. No partial credit given here, so just state the answers.)

   For this problem consider the set cover instance I with U = {1,2,....10} and
   F = {{1,2,3,4,5,6}, {1,2,3,7,8}, {4,5,6,9,10}, {2,7,8}, {7,8,9}, {10}}

   A. Recall the greedy set cover heuristic which was discussed in class. For this heuristic every step of the algorithm adds to the set cover the set in F which covers the most elements from U which are not yet covered by the algorithm..

   When the greedy algorithm is run on I the size of the set cover found by the algorithm is:

   (i). 1 (ii). 2 (iii). 3 (iv). 4

   B. Define a new extended 2-greedy which works like the greedy heuristic except that at each step the algorithm finds the pair of sets from F to add to the cover C which together add the

most uncovered elements from U to C.

When the 2-greedy set cover algorithm is used on the same set cover instance the size of the set cover found by the algorithm is:

(i). 1 (ii). 2 (iii). 3 (iv). 4 (v). None of the above

C. What is the size of the optimal set cover of I ?

(i). 1 (ii). 2 (iii). 3 (iv). 4

D. Which of the following statements are true about the two set cover heuristics?

(i). The greedy algorithm always does at least as well as the 2-greedy algorithm
(ii). The 2-greedy algorithm always does at least as well as the greedy algorithm.
(iii). The 2-greedy algorithm is not efficient, it does not run in polynomial time.
(iv). The 2-greedy algorithm is more efficient than the greedy algorithm, that is, in order to magnitude it runs a time less than the greedy algorithm.
(v). None of the above.

4. (**Max Cut Approximation)** Recall that the randomized 2-approximation algorithm for MAX-CUT goes over vertices of the input graph one by one, and assigns them to one of the cut-sets uniformly and independently at random. Consider running this algorithm on the following graph $G = (V, E)$.

G =

(i). What is the size of G's maxcut? How many maxcuts are there in G ?

(ii). What is the probability that the algorithm outputs a MAX-CUT?

(iii). What is the probability that the cut output by the algorithm has size smaller than $|E|/2$?

(iv). What is the size of the max-cut found if you apply the iterative improvement max cut heuristic to find your maxcut?

or, (iv). Recall the randomized max-cut algorithms from the homework to but instead of flipping a coin for the vertices from v_1 to v_n you first order the vertices from the largest degree vertex to the smallest and then apply the randomized heuristic to this list. In this case what is the expected value of the maxcut you obtain ? Is this different from the expected value you get with out the reordering of vertices ?