CS 530 - Fall 2018
Homework 4

Due: Thursday, November 29

Reading :
Look through the first half of chapter 34 on NP. If you find topics that are new or unclear then do some reading in this chapter or elsewhere on that topic.
Read Chapter 35 sections 1,2,3 and 5.

Problems:

1. Recall that a Las Vegas algorithm is one which always is correct and runs efficiently with high probability (in polynomial time). Assume you have an n^3 expected time Las Vegas algorithm with error probability less than 60%. Having error probability less than 60% means that the probability that the algorithm runs more than n^3 steps on an input it at most .6
(i). Describe how you can use this algorithm to solve the same problem with error probability less than 10 % and the Las Vegas algorithm you propose shoudl run in expected time $O(n^3)$ time. You should justify the 10% claim for your algorithm.
(ii). Use the Las Vegas algorithm to obtain a Monte Carlo algorithm for the same problem whose error probability is $\leq 1/3$.

2. Here is a simple one step randomized algorithm for the maxcut problem.
Algorithm for constructing a cut (S,T) in a graph G:
1. Go through the list of vertices $v_1, v_2, v_3, ..., v_n$ in G.
For each vertex v_i flip a fair coin and if it comes up head put the vertex in S, otherwise put it in T. (As I said, it is pretty simple).
Questions:
(i). What is the expected value of the cut (S,T) constructed above ?
Your answer will give the expected value in terms of m, the number of edges in G.
You should show how you calculated this expected value.
(ii). Compare this algorithm to the 2-approximation for maxcut given in class.
Specifically, does it give stronger result or a weaker result (or an incomparable) result the deterministic algorithm given in class? Explain why you think that.
(iii). Now assume that the input is a weighted graph G. You still use the same algorithm as given before part (i) above.
What is the expected value of the cut (S,T) constructed above ?
This question is pretty much the same as before as well, but now the size of the cut is the sum of the weights of the edges crossing the cut.
You answer will give the expected value in terms of the weights of the edges of G.
You should show how you calculated this expected value.
(iv). Does this algorithm say anything about approximating the size of the largest max-cut in G? Can it be used to find a large cut in G ?

3. Assume we simplify the approximate max cut algorithm for G by going through the list of vertices just one, one at a time.

So the algorithm is:

Put v_1 into S.

For i = 2, 3, ..., n

Put v_i into the set S or T depending on which adds the most edges to the cut (S,T) produced so far.

Output the cut C = (S,T)

Either prove or give a counterexample: For any G, the cut C found by the algorithm is a 2-approximation to the maxcut of G.

4. Problem 35-1 on page 1134 of the textbook. Do only parts b, c, d, and e.