CS 530 Advance Algorithm - Fall 2019
Homework #4

Ziqi Tan U88387934
ziqi1756@bu.edu

November 18, 2019

Problem 1

Recall that a clique is just a complete graph.
Consider (CP) = the max-clique problem : Given a graph G, what is the largest clique contained in G?

1. Give a polynomial time algorithm to determine if a graph contains a 4-clique (that is, a cliques of four vertices). You should say why your algorithm runs in polynomial time.

   Answer:

   Let N be the number of vertices of graph G and notation C be combination.

   Algorithm:

   - Step 1: Iterate all vertices in G. Delete the vertices with less than 3 degree and delete the edges incident to these vertices. This can be done in $O(N)$ time.
   - Step 2: Do step 1 until there is no vertices with less than 3 degree. This can be done in $O(N^2)$ time.
   - Step 3: Let G' be the subgraph we get after step 1 and step 2.
     Iterate vertices in G'. In each iteration:
     - Construct a subgraph of G' by using the current vertex and three of its neighbors and the edges that connect the current vertex and its neighbors.
     - Check whether this subgraph is a complete graph where all vertices have 3 degrees. If that is true, return TURE.
   - Step 4: Return FALSE.

   Step 3 can be done in $O(N \times C_N^3 \times 3)$ time. That is $O(N^4)$.

   Therefore this algorithm runs in polynominal time $O(N^4)$.

2. State a decision version of the clique problem (DCP).

   Decision version of clique problem:

   In a graph G, is there a clique that has a size of at least k. (k is an integer and $k \leq N$.)

3. Show that DCP is in NP.

   Answer:

   Only need to prove the verification algorithm of DCP runs in polynominal time.

   Verification problem:
   Given a witness graph G', check whether G' has a size of at least k, whether the edges of G' are in G, whether G' is a complete graph. It runs in $O(N^2)$ time (N is the number of vertices of witness).

   Explain:

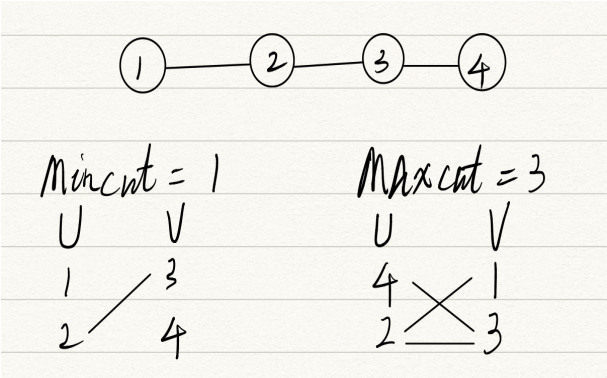
   If G' is a complete graph,

   Therefore, DCP is in NP.

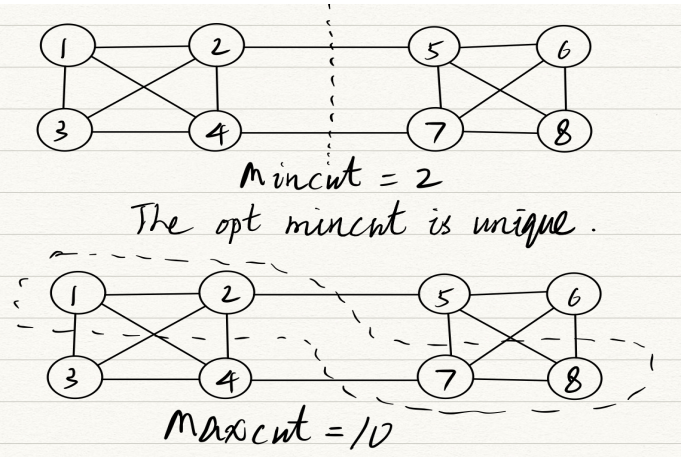
Problem 2

Recall the minimal cut problem for a graph G = (V,E). Here you need to find the smallest size cut C for G, among all cuts which are not the trivial cut. The trivial cut has all of the vertices of the graph on one side of the cut and has size 0.

1. Give an example of a graph G with 4 or more vertices whose optimal mincut has more than 1 vertex on each side of C. Say what your mincut is in your graph. Also say what the maxcut of your graph G is.



2. Give an example of a graph H with at least 3 edges whose opt mincut has exactly 1/2 of its vertices on each side of any optimal min-cut. What is the mincut in your graph H? Is the mincut in H unique ? Explain your answer. Find one such max-cut in your graph and state what the size of the maxcut.



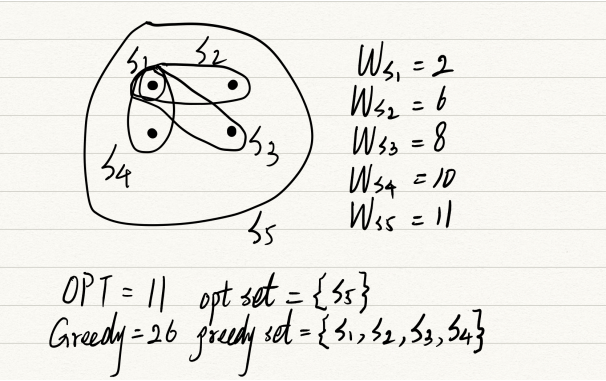
3. Give an example of a graph I with $|V| \geq 3$ and with at least $|V|$ edges whose max cut size equals its min cut size.

Problem 3

Consider the weighted Set Cover problem. In this problem each of the sets $S_1, ...S_n$ are given a non-negative weight = $cost(S_j)$ and the problem is to find a set cover of the set U which has the smallest total weight.

1. Consider using the simple greedy algorithm A which sorts the sets Si from smallest weight to largest weight, say $T_1, T_2, ..., T_n$ is the sorted list. Algorithm A works by going through the list of $T_j's$ and putting the smallest weight T_j remaining i the list into the cover C. However it will skip a set T_j if adding it does not add any new elements from U to the cover. A halts when all of the elements of U are covered.

   Give an example I for the problem where this methods results in the cover C being more than twice the optimal set cover for I. Explain why your example satisfies this requirement.



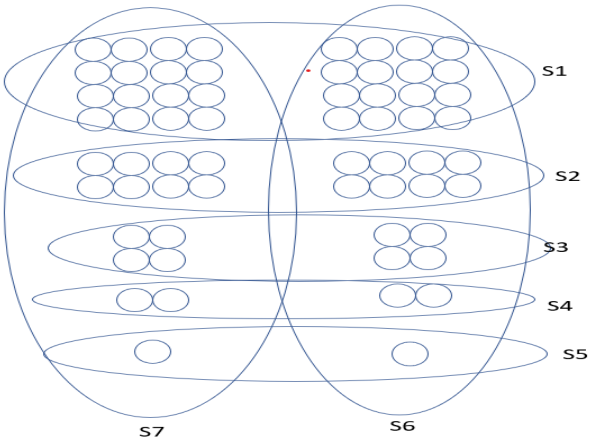
2. Now change the algorithm using the more complicated Greedy rule to add sets S_j. Greedy rule: Repeat for $t = 1, 2, 3, ...$ until all elements are covered. Let U_t be the set of uncovered elements at the beginning of iteration t and let $u_t = |U_t|$. Note that $U1 = U$, the whole universe.

   Then pick the set Sj not yet added to the cover which minimizes the ratio $cost(S_j)/(|S_j \cap U_t|)$, and add it to C.

   The problem now is the same as in (1). Give an example I of the weighted set cover problem where this new greedy method results in the size of the cover C being more than twice the optimal set cover for I. What is the cost of the set cover you obtain with this rule, and what is the cost of the optimal set cover for I?

   Answer:

   Consider this scenario where all set have weight 1.
   Optimal set cover is $\{S_6, S_7\}$ whose weight is 2.
   Greedy set cover is $\{S_1, S_2, S_3, S_4, S_5\}$ whose weight is 5.



Let's go through this greedy algorithm.

- List all the cost ratio of all sets.
  $S_1 = \frac{1}{32}, S_2 = \frac{1}{16}, S_3 = \frac{1}{8}, S_4 = \frac{1}{4}, S_5 = \frac{1}{2}, S_6 = \frac{1}{31}, S_7 = \frac{1}{31}$
  Select S_1.
- List all the cost ratio of all sets.
  $S_2 = \frac{1}{16}, S_3 = \frac{1}{8}, S_4 = \frac{1}{4}, S_5 = \frac{1}{2}, S_6 = \frac{1}{15}, S_7 = \frac{1}{15}$
  Select S_2.
- List all the cost ratio of all sets.
  $S_3 = \frac{1}{8}, S_4 = \frac{1}{4}, S_5 = \frac{1}{2}, S_6 = \frac{1}{7}, S_7 = \frac{1}{7}$
  Select S_3.
- List all the cost ratio of all sets.
  $S_4 = \frac{1}{4}, S_5 = \frac{1}{2}, S_6 = \frac{1}{6}, S_7 = \frac{1}{6}$
  Select S_4.
- List all the cost ratio of all sets.
  $S_5 = \frac{1}{2}, S_6 = \frac{1}{1}, S_7 = \frac{1}{1}$
  Select S_5.
- Greedy set cover is $\{S_1, S_2, S_3, S_4, S_5\}$ whose weight is 5.

Problem 4

Recall that in 1.5 approximation to the Euclidean TSP problem (Christofides algorithm) for a complete weighted graph G, you needed to be able to find the minimum matching of a set of vertices and edges in G.

I plan to show how to do MAXIMUM matching for bipartite graphs in class. In this problem we consider minimum weight bipartite matching. (Note: You can use negative weights on your graphs here if you like.)

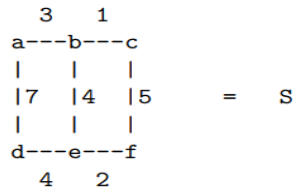
1. Assume you have an efficient method to find the maximum matching of a set of points S in a weighted graph G. Show how you could solve the Minimum weight matching

problem for S by changing the weights of G and then solving the maximum matching problem for the re-weighted graph.

Answer: Re-weighting the graph G by opposite numbers can solve the problem.

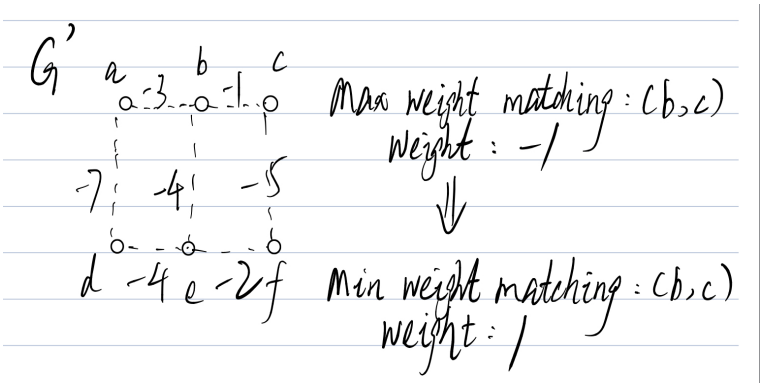
- Step 1: Given a graph G, we replace every weight by its opposite number and we get graph G'.
- Step 2: Find the maximum matching of graph G' and let it be S_m. The edges we find in set S_m will be the minimum weight matching of graph G.

2. Give an example of how your method in part (i) works on the graph G and set S pictured below. Show your work. You should assume you have the max matching algorithm, Then write S as a bipartite graph H and show what graph H' is found in part (i) whose max matching gives the min matching of H.



Answer:

We replace every weight by its opposite number and we get graph G'.
Perform algorithm to find max weight matching of G': (b,c) whose weight is -1.
The min weight matching of S is (b,c) whose weight is 1.



Problem 5

A team of three people play a game.
In the game, each player gets a hat that he/she cannot see. The hats are either blue or red, the color is decided independently and uniformly at random for each player. Each player can look at the other player's hats and then say one of three things.

- Say "red" to guess that his own hat is red.
- Say "blue" to guess that his own hat is blue.

- Say nothing and make no guess.

The players can decide on a strategy for the game but then have to answer simultaneously. The team wins if and only if at least one player guesses the color of his own hat correctly and nobody guesses the color of his own hat incorrectly. For example, if two players say nothing and one guesses the color of his own hat correctly, the team wins. As another example, if two players guess correctly and one guesses incorrectly, the team loses.

(i) Describe a strategy for which the team has a probability to win that is at least 1/2. Explain why?
Strategy with $\frac{1}{2}$ wining probability: In a game, only one player guesses the color of his own hat and the other two players say nothing.
Explain: The player who takes a guess will have $\frac{1}{2}$ probability to get the correct color of his own hat. Therefore, the probability to win is $\frac{1}{2}$.

(ii) Discuss whether it is possible to reach a probability to win of at least 3/4. If yes, describe the strategy. If not, say why there is no such strategy.

Answer: yes.

Here is a strategy for each player:
If a player finds that the color of hats of the other two players are **different**, this player **says nothing**. If a player finds that the color of hats of the other two players are **the same**, this player **answers the opposite color**.

For example, if player1 finds that player2's hat is red and player3's hat is red, then player1 guesses that his own hat is blue. if player1 finds that player2's hat is red and player3's hat is blue, then player1 does not give a guess.

The table below shows all possible situations and the corresponding answers from each player.

Situation(C_1, C_2, C_3)	Player1	Player2	Player3	Win/Lose
R R R	B	B	B	Lose
R R B	Say nothing	Say nothing	B	Win
R B R	Say nothing	B	Say nothing	Win
R B B	R	Say nothing	Say nothing	Win
B R R	B	Say nothing	Say nothing	Win
B R B	Say nothing	R	Say nothing	Win
B B R	Say nothing	Say nothing	R	Win
B B B	R	R	R	Lose

$C_i (i = 1, 2, 3)$ in situation denotes that color Red (R) or Blue (B).
Situation(C_1, C_2, C_3) means that Player1's hat is C_1, Player2's hat is C_2 and Player3's hat is C_3.

Therefore, the probability to win is $\frac{6}{8} = \frac{3}{4}$.