CSCI 570 - Fall 2016 - HW 12

Due: *Nov 28th*

1. Consider the following heuristic to compute a vertex cover of a connected graph G. Pick an arbitrary vertex as the root and perform depth first search. Output the set of non-leaf vertices (that is, vertices of degree greater than 1) in the resulting depth first search tree.

   i Show that the output is a vertex cover for G.

   ii How good of an approximation to a minimum vertex cover does this heuristic assure? That is, upper bound the ratio of the number of vertices in the output to the number of vertices in a minimum vertex cover of G.

   Let $G = (V, E)$ denote the graph, $T = (V, E_0)$ the resulting depth first search tree, L the set of leaf vertices in the depth first search tree and $N = V L)$ the set of non leaf vertices.

   (i) Assume there is an edge $e = (u, v)$ in E that is not covered by N. This implies that both u and v are in L. Without loss of generality, assume that DFS explored u first. At this stage since e was available to DFS to leave u, the DFS would have left u to explore a new vertex, thereby making u a non leaf. Hence our assumption is incorrect and N does indeed cover every edge in E.

   (ii) We next create a matching M in G from the structure of T as follows. Recall that a matching is a set of edges such that no two distinct edges in the set share a vertex.

   For every vertex u in N, pick one edge that connects u to one of its descendants in T and call it e_u. We call the set of odd level non leaf vertices in T, *ODD* and the set of even level non leaf vertices *EVEN*. If the set *ODD* is bigger or equal to the set *EVEN*, set $BIG := ODD$. Else set $BIG := EVEN$.

   Since the total number of non leaf vertices in the tree T is $|N|$, *BIG* has size at least $\frac{|N|}{2}$. Now the edge set $M = \{uEe|u \in BIG\}$ is a matching of size at least $\frac{|N|}{2}$. Since M is a matching, to cover every edge in the matching the optimal vertex cover A has to contain at least $|M|$ vertices. (This is because in a matching no two distinct edges share a vertex). Thus A has to contain at least $\frac{|N|}{2}$ vertices while our solution has $|N|$ vertices. Hence our solution is at worst a 2-*approximation*.

It can be shown that our solution can be indeed twice as bad by considering $E = \{(a, b), (b, c)\}$ with DFS rooted at a.

2. Consider the following heuristic to compute a vertex cover of a graph $G = (V, E)$.

   Step 1. Initialize A as the empty set.
   Step 2. While E is not empty {
   Pick an edge $(u, v) \in E$ and add the vertices u and v to A
   Remove the vertices u and v from G. }
   Step 3. Output A.

   Inside Step 2, when we say that a vertex is removed from G, it is implied that all edges incident on that vertex are also removed.

   Show that A is a vertex cover for G. How good of an approximation to the minimum vertex cover does this heuristic provide? That is, upper bound the ratio of the number of vertices in A to the number of vertices in a minimum vertex cover of G.

   Every edge is either picked or removed inside step 2. If an edge is picked, then both its incident vertices are added to A. If an edge is removed, it was removed inside step 2 because at least one of its incident vertices were added to A and removed from G. Thus every edge has at least one of its ends in A.

   By construction, the set of edges that are picked form a matching (since if two edges share a vertex and one of them is picked then the other one is removed and hence never picked). The size of A is twice the size of this matching. Since every vertex cover needs to be of size at least the size of this matching, size of A is at most twice as big as the minimum vertex cover.

3. A Max-Cut of an undirected graph $G = (V, E)$ is defined as a cut C_{max} such that the number of edges crossing C_{max} is the maximum possible among all cuts. Consider the following algorithm.

   i Start with an arbitrary cut C.

   ii While there exists a vertex v such that moving v from one side of C to the other increases the number of edges crossing C, move v and update C.

   a Does the algorithm terminate in time polynomial in $|V|$?

   b Prove that the algorithm is a 2-*approximation*, that is the number of edges crossing C_{max} is at most twice the number crossing C.

   See the paragraph titled "Analyzing the Algorithm", section 12.4, page 677 in the Text.

4. Read section 11.1 (2-*approximation* algorithm for minimizing makespan)

5. Write down the problem of finding a Min-s-t-Cut of a directed network with source s and sink t as an Integer Linear Program.

$$
\begin{aligned}
\textbf{minimize} \sum_{(u,v)\in E} c(u,v).x_{(u,v)} & & & \\
\textbf{subject to}: \ x_v - x_u + x_{(u,v)} &\geq 0 & \forall (u,v) \in E & \quad (1)\\
x_u &\in \{0,1\} & \forall u \in V : u \neq s, u \neq t & \quad (2)\\
x_{(u,v)} &\in \{0,1\} & \forall (u,v) \in E & \quad (3)\\
x_s &= 1 & & \quad (4)\\
x_t &= 0 & & \quad (5)
\end{aligned}
$$

The variable x_u indicates if the vertex u is on the side of s in the cut. That is, $x_u = 1$ if and only if u is on the side of s. Setting $x_s = 1$ and $x_t = 0$ ensures that s and t are separated.

Likewise, the variable $x(u,v)$ indicates if the edge (u,v) crosses the cut.

The first constraint ensures that if the edge (u,v) is in the cut, then u is on the side of s and v is on the side of t.

For completeness, you should argue that with the above correspondence (that is, $x(u,v)$ indicating if an edge crosses the cut), every min-s-t-cut corresponds to a feasible solution and vice versa.