CS 4820, Spring 2017 **Homework 2, Problem 2**
Name: Yuxiang Peng
NetID: yp344
Collaborators: jl3455, zl542

(2) *(10 points)*
Suppose we are given a graph $G = (V, E)$ with costs $(c_e)_{e \in E}$ on the edges and costs $(c_v)_{v \in V}$ on the vertices. We wish to designate a subset of the vertices as *active* and a subset of the edges as *navigable*, such that every inactive vertex can be joined to at least one active vertex by a path made up of navigable edges. Design an algorithm to choose the set of active vertices, A, and the set of navigable edges, F, so as to minimize their combined cost, $\sum_{v \in A} c_v + \sum_{e \in F} c_e$.

We will transform this problem into a minimum spanning tree problem. Suppose the optimal solution of the original problem is A. We create an imaginary vertex connecting all the active vertices in A with edge weights as the costs of the connected vertices. We now reach a feasible spanning tree solution B for all the original vertices plus the imaginary vertex since all the inactive vertices are connected to active vertices which are connected to the imaginary vertices. Solution A contains no cycles and thus B contains no cycle. Suppose the optimal solution of the MST problem is B', then cost(B')≤cost(B). If we make the vertices connected to the imaginary vertex in B' active vertices with costs as the weights of the edges connected to the imaginary vertex. Then we delete the imaginary vertex and edges connected to it. Finally we make the rest of the vertices inactive vertices. We reach a feasible solution A' to the original problem since all the inactive were connected to the imaginary vertex through the active vertices, then cost(A)≤cost(A'). Therefore the optimal solution of the original problem is the same optimal solution of the transformed MST problem in terms of the total cost. Next we just need to find the optimal solution to this transformed MST problem using Prim's algorithm. The details of the algorithm and correctness are given in the lecture notes. Using a priority queue with n ExtractMin and m ChangeKey operations, the algorithm could run at O(m log n) as explained on Pg 150 of the textbook.