CS 4820, Spring 2019 **Homework 6, Problem 2**
Name: Kevin Klaben NetID: kek228 Collaborators: Quinn Lui

(2) *(15 points)* m tourists are wandering in the streets of Manhattan, and they are hungry. Fortunately, there are m hotels as well, and you have been assigned the task to match tourists with hotels. We make the following assumptions:

- Model Manhattan as the $n \times n$ grid $[0, n-1] \times [0, n-1]$.
- Tourists can walk either horizontally or vertically on this grid. The time taken by a tourist to walk from (x_1, y_1) to (x_2, y_2) is given by $|x_1 - x_2| + |y_1 - y_2|$.
- Tourists can walk in parallel on a segment. (This models the fact that roads are wide enough for multiple people to walk simultaneously.)
- Each hotel can accommodate at most one tourist.
- Each tourist has a list of at most r hotels (which is a subset of the m available hotels) which they would be happy to go to.

You are given as input the initial locations of the m tourists, their hotel preferences, and the locations of the m hotels. Assume that time $T = 0$ initially. Define the cost of an allocation to be the minimum time T_0 such that all tourists reach their respective hotels (starting from their initial locations) at time $T = T_0$. Recall that we are assuming that the tourists walk simultaneously to their hotels.

Design an algorithm to find the minimum cost allocation that makes all the tourists happy (the algorithm should output 'No' if no such allocations exist). The running time of your algorithm should be $O(m^2 r \log n)$.

Example: Suppose there are 2 tourists a_1, a_2 located at $(0,0)$ and $(1,0)$ respectively. Further suppose there are 2 hotels h_1, h_2 located at $(0,1)$ and $(1,1)$ respectively. Let the lists of both a_1 and a_2 be $\{h_1, h_2\}$ and $r = 2$. Then, an assignment of a_1 to t_1 and a_2 to t_2 makes both of them happy. Further, under this allocation t_1 and t_2 reach their respective hotels at time $T = 1$. In the other assignment, which is a_1 to h_2 and a_2 to h_1 also makes both of them happy. However, they both reach their hotels at time $T = 2$, which is worse than the previous allocation. Thus, in this case your output should be (t_1, h_1) and (t_2, h_2).

Algorithm:
The algorithm will operate as follows: This problem can be reduce to bipartite matching using the Ford-Fulkerson problem. On one side will be m nodes representing each tourist while the other side will be the m hotels. Edges connecting the side will be the r hotels that each tourist finds acceptable. These edges will be assigns weight equivalent to the time it takes for that tourist to reach that hotel and thus each edge weight will be the —(tourist x)-(hotel x)—+—(tourist y)-(hotel y)—. An application of the divide and conquer principle will be applied to run the bipartite matching algorithm using Ford-Fulkerson as follows. The bounds initially will be min=-1 and max=2n+1. The midpoint of the range will be taken and Ford-Fulkerson bipartite matching will be run using only edges with less than or equal to weight than the midpoint. If the

Ford-fulkerson algorithm is able to find a perfect matching then max now equals this midpoint and another iteration is done with these new bounds. If the Ford-Fulkerson algorithm does not find a perfect matching then the min is now equal to the midpoint and another iteration is done. The iterations will stop when the min=max-1, in the case that max is 2n+1, "no" is returned, otherwise the matching associated with running Ford-Fulkerson with max as the max value for edge weight is returned. Each iteration of the loop will require all edges to be iterated over to determine which are less than or equal to the max value.

Runtime Analysis:
The preprocessing step to assign a weight to each edge will require iterating over each potential hotel for each tourist. As there are m tourists and r agreeable hotels this will take O(mr) time. The number of iterations of the Ford-Fulkerson bipartite matching algorithm will be logn times. This is because each time the algorithm runs it eliminates half of the range from being the potential answer. As this range is of order n, the number of times the Ford;Fulkerson algorithm will run will be O(logn). To prepare to run the Ford-Fulkerson with edges of max of midpoint weight, all edges must be iterated over and as there are mr edges, this will take O(mr) time. Now that the algorithm is setup to be run, the runtime of the Ford-Fulkerson algorithm is O((number of edges)C) where C is the maximum capacity. In this case as the source node will have an edge with weight 1 pointing to each tourist, thus, the max capacity for a cut is m. The number of edges is mr. Thus, the ford-fulkerson algorithm takes O(m^2r). Overall the runtime will be O(logn)*(O(mr)+O(m^2r))=O(m^2rlogn).

Proof of Correctness:
Claim: The algorithm will output the matching with minimal time for each tourist to reach an acceptable hotel (or "no" if no matching exists).
Proof:
For the sake of contradiction, suppose there exists a perfect matching with a lower amount of time for each tourist to reach a different agreeable hotel than the matching output by the algorithm. As the algorithm only returns the matching associated with max, when min=max-1 and the time for the supposed matching is less than max thus the supposed matching is associated with a time 0 to min. However, If this matching exists then this means that each edge within this matching is less than or equal to min. However, in order for the value of min to have been assigned to min in the first place, the Ford-Fulkerson bipartite matching algorithm must have been run with all edges less than or equal to min as a part of the matching and have found no perfect matching. However, there is a supposed perfect matching which has a maximum time of min or less. This is a contradiction and thus the claim holds.