CSE548 Fall 2019 Analysis of Algorithms Homework5

Tianao Wang 112819772

December 10, 2019

1. This problem is NP problem. If we have a time schedule, we can easily check whether there is overlap.
   We can use independent set to reduce this problem in order to prove it is NP Hard. We construct a graph G(V,E). In this graph, we have m edges which means m slices of time intervals. Each edge's endpoint vertex means a job is done in this time interval. Two jobs will not overlap if and only if they are not two endpoints on the same edge. If we can find an independent set size of k in a graph G, we can say we can accept k jobs without overlap.

2. This problem is NP problem. If we have a set X, we can check its size of intersections with other sets.
   We can use 3D Matching to reduce this problem in order to prove it is NP Hard. Assume that we have three sets X,Y,Z and each of them has size of n. Then we have a set T of m triples from X,Y,Z. For every element i, which $i \in X \cup Y \cup Z$, we create a set A_j in triples and it contains i. Then we just need to check whether a set $M \subset T$ has an intersection with each set A_j which size of 1. If there exsits M, we can say this problem is also be solved.

3. This problem is NP problem. If we have a circle, we can add these weights of edges can check whether the sum is zero or not.
   We can use the subset sum problem to reduce this problem. In subset sum problem, if we can find a subset add up to A. If we have a graph G and each edge has weight, we start from a vertex O and find a circle from O. If the last one edge is -A and other edges add to A, we can say there exsits a circle.

4. This problem is NP problem. If we have k sets, we can check whether there is a distance which is greater than B.
   We can use k graph coloring problem to reduce this problem. We can cluster these points which distances are smaller than B into one set and we will get many sets. Then for these set which distance is grater than B, we add an edge to these sets. Then we can run k graph coloring in this graph. If the graph can do k coloring, we can say objects be partitioned into k sets, so that no two points in the same set are at a distance greater than B from each other.

5. We can make a set called M. Then, we add a legal matching in T. Continue this process until we can not add new matching to M. We can call the maximum possible size M*. For every matching in M*, we can find at least one vertex is included in M. Otherwise, we can add this matching to M and this will be a contradiction. Every matching in M has three vertices and these vertices can be at most three matching in maximum possible size solution. So a 3-dimensional matching of size at least 1/3 times the maximum possible size

6. (a) If $v \in S$, it is ok because this will not change the total weight. If $v \notin S$ which means v is not choosen by the algorithm. According to the algorithm, there must be a v' which is

equal or grater than v and v is deleted. Otherwise, the algorithm will choose v not v' and this will be a contradiction.

(b) According to the definition, the algorithm will always choose the heaviest vertex and we will delete 4 vertices connecting to this vertex. The worst situation is that we one vertex and the other 4 vertices are all same weight as this choosen vertex. At the same time, these 4 deleted vertices are all in the maximum total weight independent set. In this cases, we can only get 1/4 times of the maximum total weight. (Also, this problem assumes all nodes are distinct which means the weight we get by "heaviest-first" greedy algorithm will be more than 1/4 times than maximum total weight)