1. Hao-en's method:
   I. Revised PageRank:
      i. Motivation:

      PageRank is a well-known and also the core algorithm within Google search engine. Its assumption is that the more important webpages will link with each other. In implementation, it will collect in-edges weight and deliver them to out-edges evenly. This Markov-like transportation will do iteratively until it finally converges to some certain stable status.

      Since this procedure can help identify the important nodes in the whole graph, I first implement it with some revision.

      ii. Challenges:

      PageRank is designed for no-label nodes. It is hard to add TL model knowledge onto this algorithm. In practice, if one node is labelled, I will remove that node and all edges connected to it.

      However, PageRank is not suitable for active graph, which will lose lots of nodes or edges. The reason is that in the algorithm, if one node has no out-edge, it will distributed his in-edges weight to all the nodes in the graph for the sake of retaining the Markov-like property. So, more the nodes and edges removed, the less information PageRank can provide.

      Furthermore, PageRank algorithm does not have the concept of "influence" or "threshold". I try to put transformed information, such as "influence divided by target node's threshold", into the edges, but have little improvement.

      iii. Performance:

      It is hard to defeat s1 and s2 provided by TA on the first graph. So, at the end, we do not include this algorithm in our final work.

   II. Two-layer searching:
      i. Motivation:

      The strategy provided by TA only collects information nearby the selected nodes, which is an efficient but less informative algorithm. In this method, I try to collect information not only about the selected nodes but also the nodes that is pointed by selected nodes. After doing so, I can more precisely know the benefit if I do select that node, such as, "how many nodes linked to it will definitely be propagated", "how many nodes linked to it will definitely not be propagated", and the expected number of propagated nodes pointed by it.

      ii. Challenges:

      This method has two major problems. First, the time-complexity is high, about $O(NM)$, which should be taken into concerned. Second, too many cases that needed to be considered. In other words, too many parameters needed to be trained, but few training data is available. If we want to get a set of concrete parameters, we may

have to generate graphs for training and testing set by ourselves.

iii. Performance:

The first version of this method is poor, though it is already better than the revised PageRank. In the second version, I largely scaled down (about 95%) the importance on observed influence from same color nodes when I am viewing on the node pointed by the selected node. The second version can win about 80 nodes when playing with $s1$, and 200 to 300 nodes with $s2$ on first graph.

III. Two-layer searching with greedy

i. Motivation:

Teacher mentioned in class that the greedy strategy will have at least about 60 percent performance when compared with the best strategy. So, I decided to transform the former method into a greedy-like one.

The implementation is quite simple. I do two-layer searching ten times. At each time, I will select the best node and label it with our color and then continue the procedure until ten nodes are selected.

ii. Challenge:

This strategy has all of the same problems in the above-mentioned method. Furthermore, it has a much more severe time issue, because of the ten-time selection, which will make it be terminated when running on the second graph.

In this method, I make the scale-down factor related to the iteration of selection, which is somehow a magic number.

iii. Performance:

Its performance is better than the first version of the previous strategy, but worse than the second one. It can will about 60 nodes when playing with s1, and 200 to 300 nodes with s2 on first graph.