Note: Your TA may not get to all the problems. This is totally fine, the discussion worksheets are not designed to be finished in an hour. The discussion worksheet is also a resource you can use to practice, reinforce, and build upon concepts discussed in lecture, readings, and the homework.

1 Traveling Salesman Problem

In the lecture, we learned an approximation algorithm for the Traveling Salesman Problem based on computing an MST and a depth first traversal. Suppose we run this approximation algorithm on the following graph:

```
A  B
C  D
```

The algorithm will return different tours based on the choices it makes during its depth first traversal.

1. Which DFS traversal leads to the best possible output tour?

2. Which DFS traversal leads to the worst possible output tour?

3. What is the approximation ratio given by the algorithm in the worst case for the above instance? Why is it worse than 2? (Hint: Consider the triangle inequality on the graph).
2 Independent Set Approximation

In the Max Independent Set problem, we are given a graph $G = (V, E)$ and asked to find the largest set $V' \subseteq V$ such that no two vertices in $V'$ share an edge in $E$.

Given an undirected graph $G = (V, E)$ in which each node has degree $\leq d$, give an efficient algorithm that finds an independent set whose size is at least $1/(d + 1)$ times that of the largest independent set. Only the main idea and the proof that the size is at least $1/(d + 1)$ times the largest solution’s size are needed.

3 Local Search for Max Cut

Sometimes, local search algorithms can give good approximations to NP-hard problems. In the Max-Cut problem, we have a graph $G(V, E)$ and we want to find a cut $(S, T)$ with as many edges crossing as possible. One local search algorithm is as follows: Start with any cut, and while there is some vertex $v \in S$ such that more edges cross $(S - v, T + v)$ (or some $v \in T$ such that more edges cross $(S + v, T - v)$), move $v$ to the other side of the cut. Note that when we move $v$ from $S$ to $T$, $v$ must have more neighbors in $S$ than $T$.

(a) Give an upper bound on the number of iterations this algorithm can run for (i.e. the total number of times we move a vertex).

(b) Show that when this algorithm terminates, it finds a cut where at least half the edges in the graph cross the cut.