Note: Your TA probably will not cover all the problems. This is totally fine, the discussion worksheets are not designed to be finished in an hour. They are deliberately made long so they can serve as a resource you can use to practice, reinforce, and build upon concepts discussed in lecture, readings, and the homework.

1 Bad Reductions

In each part we make a wrong claim about some reduction. Explain for each one why the claim is wrong.

(a) The shortest simple path problem with non-negative edge weights can be reduced to the longest simple path problem by just negating the weights of all edges. There is an efficient algorithm for the shortest simple path problem with non-negative edge weights, so there is also an efficient algorithm for the longest path problem.

(b) We have a reduction from problem B to problem A that takes an instance of B of size n, and creates a corresponding instance of A of size n^2. There is an algorithm that solves A in quadratic time. So our reduction also gives an algorithm that solves B in quadratic time.

(c) We have a reduction from problem B to problem A that takes an instance of B of size n, and creates a corresponding instance of A of size n in $O(n^2)$ time. There is an algorithm that solves A in linear time. So our reduction also gives an algorithm that solves B in linear time.

(d) Minimum vertex cover can be reduced to shortest path in the following way: Given a graph G, if the minimum vertex cover in G has size k, we can create a new graph G' where the shortest path from s to t in G' has length k. The shortest path length in G' and size of the minimum vertex cover in G are the same, so if we have an efficient algorithm for shortest path, we also have one for vertex cover.

2 Cycle Cover

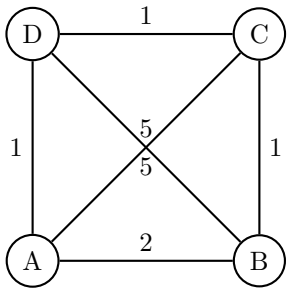
In the cycle cover problem, we have a directed graph G, and our goal is to find a set of directed cycles $C_1, C_2, \ldots C_k$ in G such that every vertex appears in exactly one cycle (a cycle cannot revisit vertices, e.g. $a \to b \to a \to c \to a$ is not a valid cycle, but $a \to b \to c \to a$ is), or declare none exists.

In the bipartite perfect matching problem, we have an undirected bipartite graph (a graph where the vertices can be split into L, R, and there are no edges between two vertices in L or two vertices in R), and our goal is to find a set of edges in this graph such that every vertex is adjacent to exactly one edge in the set, or declare none exists.

Give a reduction from cycle cover to bipartite perfect matching. *(Hint: In a cycle cover, every vertex has one incoming and one outgoing edge.)*

3 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:



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*).

4 Maximum Coverage

In the maximum coverage problem, we have m subsets of the set $\{1, 2, \ldots n\}$, denoted $S_1, S_2, \ldots S_m$. We are given an integer k, and we want to choose k sets whose union is as large as possible.

Give an efficient algorithm that finds k sets whose union has size at least $(1 - 1/e) \cdot OPT$, where OPT is the maximum number of elements in the union of any k sets. In other words, $OPT = \max_{i_1, i_2, \ldots i_k} |\cup_{j=1}^{k} S_{i_j}|$. Provide an algorithm description and justify the lower bound on the number of elements covered by your solution.

(*Hint: recall the set cover algorithm from lecture, and use that* $(1 - 1/n)^n \leq 1/e$ *for all* $n \in \mathbb{Z}$)

5 Randomization for Approximation

Oftentimes, extremely simple randomized algorithms can achieve reasonably good approximation factors.

(a) Consider Max 3-SAT (given a set of 3-clauses, find the assignment that satisfies as many of them as possible). Come up with a simple randomized algorithm that will achieve an approximation factor of $\frac{7}{8}$ in expectation. That is, if the optimal solution satisfies k clauses, your algorithm should produce an assignment that satisfies at least $\frac{7}{8}k$ clauses in expectation. You may assume that every clause contains exactly 3 distinct variables. (*Hint: what is the simplest randomized algorithm for assigning values to variables?*)

(b) Given a Max 3-SAT instance I, let OPT_I denote the maximum fraction of clauses in I satisfied by any variable assignment. What is the smallest value of OPT_I over all instances I? In other words, what is $\min_I OPT_I$? (Again, you may assume that every clause contains exactly 3 distinct variables)