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 Longest Huffman Tree

Under a Huffman encoding of n symbols with frequencies $f_1, f_2, \ldots, f_n$, what is the longest a codeword could possibly be? Give an example set of frequencies that would produce this case, and argue that it is the longest possible.

2 Twenty Questions

Your friend challenges you to a variant of the guessing game 20 questions. First, they pick some word $(w_1, w_2, ..., w_n)$ according to a known probability distribution $(p_1, p_2, ..., p_n)$, i.e. word w_i is chosen with probability p_i. Then, you ask yes/no questions until you are certain which word has been chosen. You can ask any yes/no question, meaning you can eliminate any subset S of the possible words with the question "Is the word in S?".

Define the cost of a guessing strategy as the expected number of queries it requires to determine the chosen word, and let an optimal strategy be one which minimizes cost. Design an $O(n \log n)$ algorithm to determine the cost of the optimal strategy.

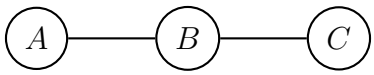
Note: *We are only considering deterministic guessing strategies in this question. Including randomized strategies doesn't change the answer, but it makes the proof of correctness more difficult.*

3 Graph Game

Given an undirected, unweighted graph G, with each node v having a value $\ell(v) \geq 0$, consider the following game.

1. All nodes are initially *unmarked* and your score is 0.
2. Choose an unmarked node u. Let $M(u)$ be the *marked* neighbours of u. Add $\sum_{v \in M(u)} \ell(v)$ to your score. Then mark u.
3. Repeat the last step for as many turns as you like, or until all the nodes are marked.

For instance, suppose we had the graph:



with $\ell(A) = 3$, $\ell(B) = 2$, $\ell(C) = 3$. Then, an optimal strategy is to mark A then C then B giving you a score of $0 + 0 + 6$. We can check that no other order will give us a better score.

(a) Is the following statement true or false? **For any graph, an optimal strategy which marks all the vertices always exists.** Briefly justify your answer.

(b) Give a greedy algorithm to find the order which gives the best score. **Give the algorithm description and runtime; no need to provide proof of correctness.**

(c) Now suppose that $\ell(v)$ can be negative. Give an example where your algorithm fails.

(d) Your friend suggests the following modified algorithm: delete all v with $\ell(v) < 0$, then run your greedy algorithm on the resulting graph. Give an example where this algorithm fails.

4 Vertex Numbering

Here is an implementation of Bellman-Ford algorithm:

Algorithm 1

Require: directed graph $G = (V, E)$ with edge weights $\{w(e) \mid e \in E\}$
Ensure: compute distances $\texttt{dist}(u)$ to each vertex u from source s

```
1: procedure BELLMAN_FORD(G = (V, E), w : e → ℝ)
2:     for u ∈ V do
3:         dist(u) ← ∞
4:         prev(u) ← null
5:     dist(s) ← 0
6:     k ← 0
7:     repeat
8:         for all edges e = (u, v) ∈ E do
9:             update(e)                                  ▷ Relax edge e
10:        k ← k + 1
11:    until no change in dist values OR k = n − 1
12:    return dist
```

It turns out that the runtime of the above algorithm can be very sensitive to the order in which we iterate through the edges and relax them. Consider the following graph G on 6 vertices:

$$\text{A} \longrightarrow \text{B} \longrightarrow \text{C} \longrightarrow \text{D} \longrightarrow \text{E} \longrightarrow \text{F}$$

Provide two ways to iterate through the edges of G, such that in one ordering the algorithm makes $2|E| = 10$ calls to `update`, while in the other ordering the algorithm terminates in $(|V| - 1)|E| = 25$ calls to `update`.

5 AMA

As a quick run-through, here are the topics we've covered so far in class:

- Asymptotics
- Recurrence Relations
- Divide-and-Conquer
- Fast Fourier Transform (FFT)
- Depth First Search (DFS)
- Topological Sort
- Strongly Connected Components (SCC)
- Breadth First Search (BFS)
- Dijkstra's
- Bellman-Ford
- Greedy

Which of these topics do you want to understand better? Is there anything you want to clarify regarding the algorithms, proofs of correctness, runtime analysis, or any other things covered in class? Please reflect on this and ask your TA any questions you may have!