CS 170 Homework 4

Due **Monday 2/16/2026, at 10:00 pm (grace period until 11:59pm)**

Study Group

List the names and SIDs of the members in your study group. If you have no collaborators, explicitly write "none".

1 Implementing Horn Formula Satisfiability (Solo Question; 10 points)

Show how to implement the stingy algorithm for Horn formula satisfiability in time that is linear in the length of the formula (the number of occurrences of literals in it).

Specifically, recall that you are given n boolean variables, and a set of clauses that come in one of the following two forms, for any $k \geq 0$:

1. "$x_1 \wedge \cdots \wedge x_k \implies y$" meaning that if variables $x_1, \ldots, x_k$ are all true, then variable y is also true. (If $k = 0$, the clause "$\implies y$" simply means y must be true.)
2. "$\bar{x}_1 \vee \cdots \vee \bar{x}_k$" meaning that at least one of the variables $x_1, \ldots, x_k$ is false.

Your algorithm should either assign the variables to "true" or "false" such that all clauses are satisfied, or else determine that no such assignment exists. If a given variable z is involved in c_z clauses, your algorithm should spend $O(c_z)$ time on updates involving z.

2 Scheduling (5 points)

There are n jobs to be run on a computer. Each job j starts at some time t_j, runs for exactly one hour, and gives a reward payment r_j if it finishes. Naturally, two jobs whose starting times differ by less than an hour will overlap, and cannot both be run.

A natural greedy algorithm is to sort the jobs from highest to lowest reward, and then repeatedly select the remaining job with the highest reward that does not overlap any previously selected jobs. Either give an exchange argument showing that this algorithm finds an optimal solution, or give a counterexample where it does not find one.

3 Ternary Huffman (10 points)

Trimedia Disks Inc. has developed "ternary" hard disks. Each cell on a disk can now store values 0, 1, or 2 (instead of just 0 or 1). To take advantage of this new technology, in this problem you will construct a modified Huffman algorithm for compressing sequences of characters from an alphabet of size n, where the characters occur with known frequencies $f_1, \ldots, f_n$. Your algorithm should encode each character with a variable-length codeword over the values 0, 1, 2 such that no codeword is a prefix of another codeword and so as to obtain the maximum possible compression. This prefix-free code is represented by a ternary tree, in which each node has ≤ 3 children.

(a) (5 points) Show that if the optimal code has a full ternary tree (meaning every non-leaf node has 3 children), then the Huffman algorithm is optimal, where now we repeatedly group together the lowest 3 frequencies (instead of the lowest 2).

(b) (5 points) Now argue that in general, there is a prefix-free optimal code for which at most one non-leaf node in the tree has 2 children, and all other non-leaf nodes have 3 children. Use this fact to give a version of the Huffman algorithm that gives an optimal ternary code for arbitrary n and $f_1, \ldots, f_n$.

4 Frenemies (7 points)

A group of n guests shows up at a house for a party, but the host knows that m pairs of these guests are frenemies. Being frenemies is mutual (if A is a frenemy of B, then B is a frenemy of A), and a guest can be frenemies with multiple other guests. There are two rooms in the house, and the host wants to distribute guests among the rooms, breaking up as many pairs of frenemies as possible by assigning them to different rooms. The guests are all waiting outside the house and are impatient to get in, so the host needs to assign them to the two rooms quickly, even if this means that it's not the best possible solution. Come up with a $O(n+m)$-time algorithm that breaks up at least half of the pairs of frenemies.

Please provide an algorithm description, justify that it breaks up at least $m/2$ pairs of frenemies, and justify the runtime.

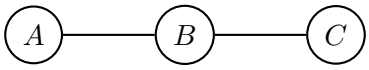
5 Graph Game (7 points)

Given an undirected, unweighted graph G, with each node v having a value $\ell(v) \geq 0$. You will mark all the vertices one at a time, adding points as follows:

1. Initially, no vertices are marked, and your score is 0
2. When you choose an unmarked vertex u to mark, look at all neighbors of u that have already been marked. For each such neighbor v, you gain $\ell(v)$ points.
3. Continue until all vertices are marked.

Your goal is to choose the order of the marking that maximizes your total score.

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.

Give a greedy algorithm to find the order which gives the best score. Explain why your algorithm works, and analyze the running time.

6 Short Paths (10 points)

You are given a weighted directed graph $G = (V, E)$ with no negative-weight cycles (but some edges may have negative weight).

(a) (8 points) Give an efficient algorithm to find the shortest path from a source vertex s to a target vertex t that uses at most some specified number K of negative-weight edges.

(b) (2 points) Does your algorithm still work if there are negative-weight cycles in G?

7 Entropy or Coding (10 points)

For full credit, you should do **one of the following two** questions. (You may solve the other for fun if you want!)

1. **Entropy and Huffman Encoding**

   Consider a distribution over n possible symbols, with probabilities $p_1, \ldots, p_n$.

   (a) (8 points) Assume that each p_i is of the form $1/2^{\ell_i}$ for a positive integer ℓ_i. Suppose a long sequence of m symbols is drawn from the distribution, and each symbol i appears exactly mp_i times. Show that the Huffman encoding of this sequence has length $m \cdot H$, where

   $$H = \sum_{i=1}^{n} p_i \log \frac{1}{p_i}.$$

   denotes the *entropy* of our distribution.

   Hint: Show that the Huffman encoding of symbol i has length ℓ_i.

   (b) (Optional; 0 points) Show that if we instead allow arbitrary probabilities p_i, the Huffman encoding still has length at most $m(1 + H)$.

   (c) (Open ended, do as little or much as you find interesting; 2 points) If we set $n = 26$ and let p_i be the frequency of the ith letter of the alphabet in English language, we could compute the entropy $H_{\text{English}} = \sum_{i=1}^{26} p_i \log(1/p_i)$. Part (b) shows that an n-letter message can be compressed to $m(1 + H_{\text{English}})$ bits. Would you expect there to exist an alternative scheme achieving shorter encoding length?

   Hint: If you're interested, feel free to look up/ask an LLM about Shannon's paper "A Mathematical Theory of Communication," *or look up how LLMs represent language using* tokens*. If we let each token be a symbol in our Huffman encoding scheme, would the encoding be shorter or longer than when each letter is a symbol?*

2. **Coding Graph Algorithms**

   In this coding question, we'll implement some shortest paths algorithms. There are two ways that you can access the notebook and complete the problems:

   (a) **On Datahub**: click here and navigate to the `hw04` folder.

   (b) **On Local Machine**: `git clone` (or if you already cloned it, `git pull`) from the coding homework repo,

   `https://github.com/Berkeley-CS170/cs170-sp26-coding`

   and navigate to the `hw04` folder. Refer to the `README.md` for local setup instructions.

   Notes:

   - *Submission Instructions:* Please download your completed submission `.zip` file and submit it to the Gradescope assignment titled "HW04 (Coding)".

- *Getting Help:* Conceptual questions are always welcome on Edstem and office hours; *note that support for debugging help during OH will be limited*. If you need debugging help first try asking on the public Edstem threads. To ensure others can help you, make sure to:

  (a) Describe the steps you've taken to debug the issue prior to posting on Ed.

  (b) Describe the specific error you're running into.

  (c) Include a few small but nontrivial test cases, alongside both the output you expected to receive and your function's actual output.

  If staff tells you to make a private Ed post, make sure to include *all of the above items* plus your full function implementation. If you don't provide them, we will ask you to provide them.

- *Academic Honesty Guideline:* We realize that code for some of the algorithms we ask you to implement may be readily available online, but we strongly encourage you to not directly copy code from these sources. Instead, try to refer to the resources mentioned in the notebook and come up with code yourself. That being said, we **do acknowledge** that there may not be many different ways to code up particular algorithms and that your solution may be similar to other solutions available online.

