CS 170 Homework 5 (Optional)

Due 10/3/2022, at 10:00 pm (grace period until 11:59pm)

1 Study Group

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

As this homework is optional, it will be worth no points.

2 Bounding Sums

Let \( f(\cdot) \) be a function. Consider the equality

\[
\sum_{i=1}^{n} f(i) \in \Theta(f(n)),
\]

Give a function \( f_1 \) such that the equality holds, and a function \( f_2 \) such that the equality does not hold.

3 True and False Practice

For the following problems, justify your answer or provide a counterexample.

(a) True or False: Kruskal’s works with negative edges.

(b) We modify a graph \( G \) with negative edges by adding a large positive constant to each edge, making all edges positive. Let’s call this modified graph \( G' \).
   True or False: If we run Dijkstra’s on \( G' \), the resulting shortest paths on \( G' \) are also the shortest paths on \( G \).

(c) Let \( G = (V, E) \) be a DAG with positive edge weights. We first run Dijkstra’s algorithm to compute the distance from the source \( s \) to every other vertex \( v \). Afterwards, we store the vertices in increasing order of their distance from \( s \).
   True or False: this sequence of vertices be a valid topological sort of \( G \).

(d) Let \( G = (V, E) \) be an undirected graph. Let \( G' = (V', E') \) where \( V' = V \cup \{u\} \) and \( E' = E \cup E_u \), where \( E_u \) is some set of edges that include \( u \).
   True or False: any MST of \( G \) is the subset of some MST of \( G' \).

4 Agent Meetup

Manhattan has an “amazing” road system where streets form a checkerboard pattern, and all roads are either straight North-South or East-West. We simplify Manhattan’s roadmap by assuming that each pair \( x, y \), where \( x \) and \( y \) are integers, corresponds to an intersection.
As a result, the distance between any two intersections can be measured as the \textit{Manhattan distance} between them, i.e. $|x_i - x_j| + |y_i - y_j|$. You, working as a mission coordinator at the CS 170 Secret Service Agency, have to arrange a meeting between two of $n$ secret agents located at intersections across Manhattan. Hence, your goal is to find the two agents that are the closest to each other, as measured by their Manhattan distance.

In this problem, you will devise an efficient algorithm for this special purpose. As mentioned before, you can assume that the coordinates of the agents are integer values, i.e. the $i$th agent is at location $(x_i, y_i)$ where $x_i, y_i$ are integers.

\textit{Note: This problem is very geometric, we suggest you draw examples when working on it!}

(a) Let $(a, b)$ be an arbitrary intersection. Suppose all agents $i$ for which $x_i > a$ are Manhattan distance strictly greater than $d$ apart from each other, where $d > 0$. Prove that we can have up to 8 agents that satisfy $a \leq x_i \leq a + d$ and $b - d \leq y_i \leq b + d$ for each agent $i$. In other words, show that we can fit up to 8 agents within this rectangle (visualized below) without two of these agents being Manhattan distance $d$ or less apart?

(b) Design a divide and conquer algorithm to find the minimum Manhattan distance between any two agents in $O(n \log^2 n)$ time. \textbf{Give a three-part solution.} Note that the optimal runtime is $\Theta(n \log n)$, but it is not required for full credit.

\textit{Hint: Try sorting the list of agents by y-coordinate, and then sorting the list of agents by x coordinate. Note that since all distances are integer values, we only need to consider pairs that are $d - 1$ values away from each other, where $d$ is the smallest distance that we have computed so far.}

5 Money Changing

Fix a set of positive integers called \textit{denominations} $x_1, x_2, \ldots, x_n$ (think of them as the integers 1, 5, 10, and 25). The problem you want to solve for these denominations is the following:
Given an integer \( A \), express it as

\[
A = \sum_{i=1}^{n} a_i x_i
\]

for some nonnegative integers \( a_1, \ldots, a_n \geq 0 \).

(a) Under which conditions on the denominations \( x_i \) are you able to do this for all integers \( A > 0 \)?

(b) Suppose that you want, given \( A \), to find the nonnegative \( a_i \)'s that satisfy

\[
A = \sum_{i=1}^{n} a_i x_i,
\]

and such that the sum of all \( a_i \)'s is minimal —that is, you use the smallest possible number of coins. Define a greedy algorithm for this problem. (Your greedy algorithm may not necessarily solve the problem, i.e., it may give a suboptimal answer on some inputs)

(c) Show that the greedy algorithm finds the optimum \( a_i \)'s in the case of the denominations

1, 5, 10, and 25, and for any amount \( A \).

(d) Give an example of a denomination where the greedy algorithm fails to find the optimum \( a_i \)'s for some \( A \). (Do you know of an actual country where such a set of denominations exists?)

6 Box Union

There are \( n \) boxes labeled 1, \ldots, \( n \), and initially they are each in their own stack. You want to design an algorithm or data structure that supports the following two operations:

- \( \text{put}(a, b) \): this puts the stack that \( a \) is in on top of the stack that \( b \) is in.
- \( \text{under}(a) \): this returns the number of boxes under \( a \) in its stack.

The amortized time per operation should be the same as the amortized time for \( \text{find}(\cdot) \) and \( \text{union}(\cdot, \cdot) \) operations in the union find data structure.

Hint: use “disjoint forest” and augment nodes to have an extra field \( z \) stored. Make sure this field is something easily updateable during “union by rank” and “path compression”, yet useful enough to help you answer \( \text{under}(\cdot) \) queries quickly. It may be useful to note that your algorithm for answering \( \text{under} \) queries gets to see the \( z \) values of all nodes from the query node to its tree’s root if you do a \( \text{find} \).

7 Coding Questions

For this week’s coding questions, we’ll be implementing Kosaraju’s Algorithm for finding strongly connected components (hw5_coding_SCC.ipynb) and the Union Find Algorithm which is used in Kruskal’s Algorithm (hw5_coding_MST.ipynb). You can find the coding problems in the hw5 folder [here](https://example.com). You can also download the file to your personal computer from [here](https://example.com) and complete the exercises locally. Please complete the Jupyter Notebook and make sure to submit your work for each problem to the corresponding subpart on Gradescope.
(a) **Kosaraju’s Algorithm** (hw5_coding_SCC.ipynb)

(b) **Union Find and Kruskal’s Algorithm** (hw5_coding_MST.ipynb)

Notes:

- **Submission Instructions:** Please merge your completed Jupyter Notebook with your written solutions for other questions and submit one merged pdf file to Gradescope.

- **OH/HWP Instructions:** While we will be providing conceptual help on the coding portion of homeworks, OH staff will not look at your code and/or help you debug.

- **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.