**HW6**

**Dylan Phoutthavong**

**May 4th, 2025**

**CSCI 3412**

1. **A screenshot of a computer program

   AI-generated content may be incorrect.Implementing Dijkstra's algorithm in Python (10 points)**
2. **Application of Dijkstra's SSSP and MST algorithms (15 points)**

A computer screen with white text

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

1. **A screenshot of a computer program

   AI-generated content may be incorrect.TinyZip and TinyUnzip (25 points)**

A screenshot of a computer

AI-generated content may be incorrect.

1. **[Extra credit problems]**
   1. **[Problem 1: Word Cloud and OBST, 10 points]**
      1. Yes!

A close-up of words

AI-generated content may be incorrect.

A black background with white text

AI-generated content may be incorrect.

For the “Denver” News API query, we generated a word cloud and extracted the top 50 most frequent words with their relative probabilities. These probabilities were used to construct both a regular Binary Search Tree (BST) and an Optimal Binary Search Tree (OBST).

The randomized BST had a total search cost of **2.5815**, while the OBST, constructed using dynamic programming to minimize expected lookup cost, had a significantly lower cost of **1.5163**.

This results in a **41.26%** improvement in lookup efficiency, reinforcing the value of using OBST when access probabilities are known.

* 1. **[Problem 2: Minimal Changes, 5 points]**
     1. Example: Representing $269.63

We convert everything into cents to avoid decimal issues:

* → $269.63 = 26,963 cents

Using the greedy method:

1. Start with the largest denomination ($100 = 10,000¢)
2. At each step, use as many of that denomination as possible
3. Subtract the value, and repeat with the next smaller denomination

Result for $269.63:

* 2 × $100 bills (20000¢)
* 1 × $50 bill (5000¢)
* 1 × $10 bill (1000¢)
* 1 × $5 bill (500¢)
* 4 × $1 bills (400¢)
* 2 × 25¢ coins (50¢)
* 1 × 10¢ coin (10¢)
* 3 × 1¢ coins (3¢)

**Total coins and bills = 2 + 1 + 1 + 1 + 4 + 2 + 1 + 3 = 15**

**A black screen with white text

AI-generated content may be incorrect.**

**Assumption Where Greedy Might Fail**

The greedy algorithm may fail if denominations are non-canonical (they don’t follow standard coinage).

Example: if we had coins {1¢, 3¢, 4¢} and wanted to make 6¢, greedy would choose 4¢ + 1¢ + 1¢ = 3 coins, but optimal is 3¢ + 3¢ = 2 coins.