HW6

Dylan Phoutthavong

May 4th, 2025

CSCI 3412

1. Implementing Dijkstra's algorithm in Python (10 points)

```
.venv(base) dylanpho@Dylans-MacBook-Air-4 HW6 % python dijkstra_trace.py
Node(A with Weight 0) is added to the 'Visited' {'A'} # selected
Relaxed: vertex B: OLD: Infinity, NEW: 4, Paths: {'B': 'A'}
Relaxed: vertex C: OLD: Infinity, NEW: 2, Paths: {'C': 'A'}
Relaxed: vertex E: OLD: Infinity, NEW: 7, Paths: {'E': 'A'}
Node(C with Weight 2) is added to the 'Visited' {'A', 'C'} # selected
Relaxed: vertex F: OLD: Infinity, NEW: 6, Paths: {'F': 'C'}
Node(B with Weight 4) is added to the 'Visited' {'A', 'B', 'C'} # selected
Relaxed: vertex D: OLD: Infinity, NEW: 9, Paths: {'D': 'B'}
Node(F with Weight 6) is added to the 'Visited' {'A', 'B', 'C', 'F'} # selected
Relaxed: vertex H: OLD: Infinity, NEW: 7, Paths: {'H': 'F'}
Node(E with Weight 7) is added to the 'Visited' {'A', 'B', 'C', 'E', 'F'} # selected
Relaxed: vertex G: OLD: Infinity, NEW: 9, Paths: {'G': 'E'}
Node(H with Weight 7) is added to the 'Visited' {'A', 'B', 'C', 'E', 'F', 'H'} # selected
No unvisited outgoing edges from the node, H
Node(D with Weight 9) is added to the 'Visited' {'A', 'B', 'C', 'D', 'E', 'F', 'H'} # selected
No unvisited outgoing edges from the node, D
Node(G with Weight 9) is added to the 'Visited' {'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H'} # selected
No edge relaxation is needed for node, G

Final shortest distances from source node:
A: 0
B: 4
C: 2
D: 9
E: 7
F: 6
G: 9
H: 7
```

2. Application of Dijkstra's SSSP and MST algorithms (15 points)

```
.venv(base) dylanpho@Dylans-MacBook-Air-4 HW6 % python Dijkstra_SSSP_and_Prim_MST.py

Dijkstra Output:

Distance from Denver to 'Atlanta': 2221 with path(Denver to Dallas to Atlanta)
Distance from Denver to 'Boston': 2839 with path(Denver to Boston)
Distance from Denver to 'Chicago': 1474 with path(Denver to Chicago)
Distance from Denver to 'Dallas': 1064 with path(Denver to Dallas)
Distance from Denver to 'Denver': 0 with path(Denver)
Distance from Denver to 'Houston': 1426 with path(Denver to Dallas to Houston)
Distance from Denver to 'LA': 1335 with path(Denver to LA)
Distance from Denver to 'Memphis': 1411 with path(Denver to Memphis)
Distance from Denver to 'Miami': 3194 with path(Denver to Dallas to Atlanta to Miami)
Distance from Denver to 'NY': 2619 with path(Denver to Chicago to NY)
Distance from Denver to 'Philadelphia': 2594 with path(Denver to Washington to Philadelphia)
Distance from Denver to 'Phoenix': 2486 with path(Denver to Dallas to Phoenix)
Distance from Denver to 'SF': 1894 with path(Denver to LA to SF)
Distance from Denver to 'Seattle': 2879 with path(Denver to LA to Seattle)
Distance from Denver to 'Washington': 2395 with path(Denver to Washington)
```

```
Prim's MST Output:

Denver is selected. Distance: 0
Dallas is selected. Distance: 1064
Houston is selected. Distance: 362
Memphis is selected. Distance: 675
Atlanta is selected. Distance: 1157
Miami is selected. Distance: 973
LA is selected. Distance: 1335
SF is selected. Distance: 559
Seattle is selected. Distance: 1092
Philadelphia is selected. Distance: 1413
Washington is selected. Distance: 199
Phoenix is selected. Distance: 1422
Chicago is selected. Distance: 1474
NY is selected. Distance: 1145
Boston is selected. Distance: 306

                Edge                            Weight

        Memphis    Philadelphia .................1413
        Atlanta           Miami .................973
         Dallas         Atlanta .................1157
        Chicago              NY .................1145
         Dallas         Phoenix .................1422
         Denver              LA .................1335
         Denver          Dallas .................1064
         Dallas         Memphis .................675
             LA              SF .................559
             SF         Seattle .................1092
         Dallas         Houston .................362
         Denver         Chicago .................1474
             NY          Boston .................306
   Philadelphia      Washington .................199

Total MST:       13176
```

3. **TinyZip and TinyUnzip (25 points)**

Homeworks > HW6 > tinyzip_output.txt

```
Character  Weight     Huffman Code
----------------------------------------
'space'    1623       111
'e'        885        010
't'        656        1011
'o'        604        1010
'i'        542        1000
'a'        539        0111
'n'        450        0011
's'        419        0001
'r'        413        0000
'h'        385        11011
'l'        328        11000
'd'        254        01101
'f'        220        00101
'm'        181        110101
'c'        176        110100
'u'        175        110011
'g'        167        100111
'w'        146        100101
'y'        124        011001
'b'        110        011000
'p'        93         1100101
'v'        81         1100100
'.'        75         1001100
','        71         1001001
'\\n'      58         0010011
'k'        51         0010010
'N'        24         00100010
'I'        23         00100000
'j'        20         100110110
'A'        18         100110100
'W'        18         100110101
'T'        14         100100000
'L'        12         001000010
'"'        12         001000110
'G'        9          1001000111
'!'        8          1001000010
"'"        8          1001000101
'S'        8          1001000110
'M'        7          0010001111
'q'        6          0010000110
'C'        6          0010000111
```

```
'z'         6           0010001110
'x'         5           10011011110
'F'         4           10010000110
'B'         4           10010001001
'O'         4           10011011100
'Y'         3           100110111110
'P'         3           100110111111
'-'         2           100100010000
':'         2           100100010001
';'         2           100110111010
'R'         1           1001000011100
'J'         1           1001000011101
'H'         1           1001000011110
'D'         1           1001000011111
'_'         1           1001101110110
'E'         1           10011011101110
'?'         1           10011011101111

--- Compression Stats ---
Huffman cost: 39654
ASCII cost: 72488
Huffman vs ASCII: 45%
Optimal FCL cost: 54366
Huffman vs FCL: 27%
King.txt size: 9061
King.zip size: 4956
King.unzipped.txt size: 9061
```

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

NewsAPI Word Cloud for 'Denver'


```
--- OBST vs BST Comparison for 'Denver' Query ---
BST Cost (random order): 2.5815
OBST Cost (optimal):     1.5163
Improvement:             41.26%

Summary:
For the 'Denver' query, OBST reduced the total search cost by 41.26% compared to a randomly built BST.
```

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.

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

We convert everything into cents to avoid decimal issues:

- → $269.63 = 26,963 cents

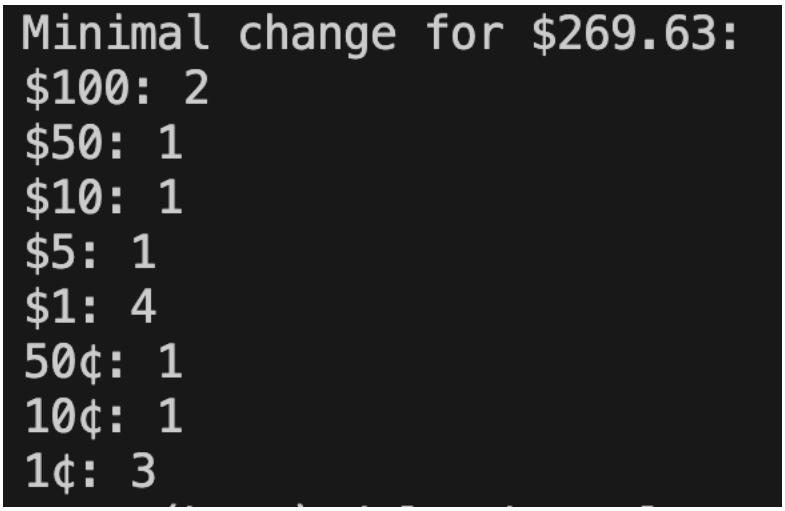
Using the greedy method:

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



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.