Day 26

DIJKSTRA'S ALGORITHM

HUFFMAN CODES

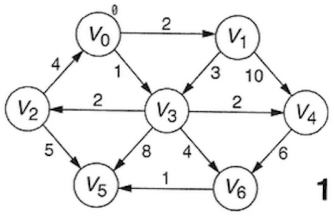
Dijkstra's Algorithm

Single-source shortest paths problem: Given a weighted connected graph G, find shortest paths from source vertex s to each of the other vertices

Start with the specified vertex s.

In the example, this is v_0

Annotate the cost so far, i.e. 0



1

Image source: figure 14.21 of Weiss: Data Structures and Problem solving Using Java, 4th edition

Dijkstra's Algorithm

From the current node, i.e. v_0, look at all edges out of it.

Annotate the destination nodes of the edges with the edge cost.

In the example, we annotated v_1 and v_3

Pick the node with the lowest cost.

In the example, that is node v_3

Before moving on, mark v_0 as done.

We have found the lowest cost path to it.

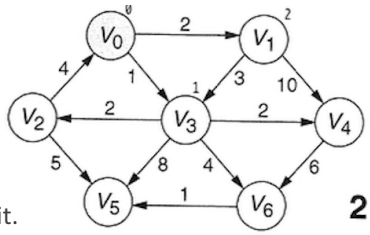


Image source: figure 14.21 of Weiss: Data Structures and Problem solving Using Java, 4th edition

Dijkstra's Algorithm

From the current node, i.e. v_1, look at all edges out of it.

This time there are no nodes to annotate or update:

- For node v_3 we have already found the lowest cost path
- For node v_4, there is a lower cost path through v_3.

Pick the node with the lowest cost.

In the example, we could pick v_2 or v_4

We pick v_4

Before moving on, mark v_1 as done.

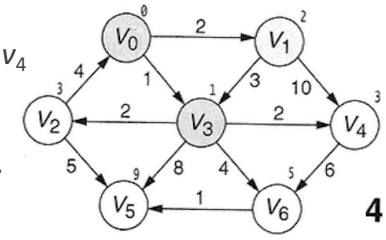
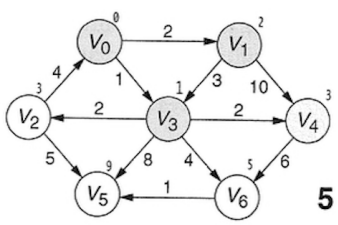


Image source: figure 14.21 of Weiss: Data Structures and Problem solving Using Java, 4th edition

Dijkstra's Algorithm

From node v_4, there are no updates.

From node v_2, the cost to v_5 improves by 1.



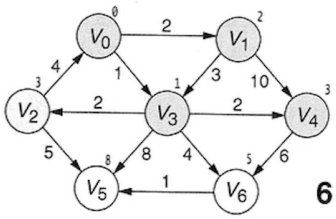


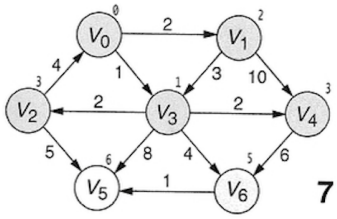
Image source: figure 14.21 of Weiss: Data Structures and Problem solving Using Java, 4th edition

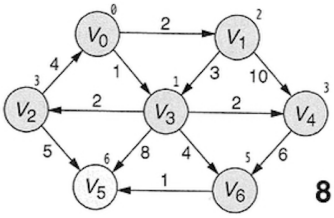
Dijkstra's Algorithm

From node v_6, the cost to v_5 improves from 8 to 6.

The final node to be considered is node v_5, but no edges lead out of it.

We are done.





Animation: https://www.cs.usfca.edu/~galles/visualization/Dijkstra.html

Image source: figure 14.21 of Weiss: Data Structures and Problem solving Using Java, 4th edition

Dijkstra's Analysis

Only works for graphs with nonnegative weights

Efficiency

- $O(|V|^2)$ for graphs represented by weight matrix and unordered-array implementation of priority queue
- $O(|E|\log|V|)$ for graphs represented by adjacency lists and min-heap implementation of priority queue

Correctness? (You will prove on A12!)

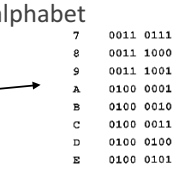
Encoding

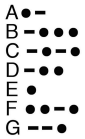
Encoding: assignment of bit strings to alphabet characters

Codewords: bit strings assigned for characters of alphabet

Two types of codes:

- *fixed-length encoding* (e.g., ASCII)
- *variable-length encoding* (e,g., Morse code)
  - Why is Morse code for E designed to be shorter than F?





File Compression

Compress file so as to reduce the length of the file.

Use variable-length encoding.

Take advantage of frequencies of characters.

Huffman Encoding

figure 12.1
A standard coding scheme

Character	Code	Frequency	Total Bits
a	000	10	30
e	001	15	45
i	010	12	36
s	011	3	9
t	100	4	12
sp	101	13	39
nl	110	1	3
Total			**174**

Huffman Encoding

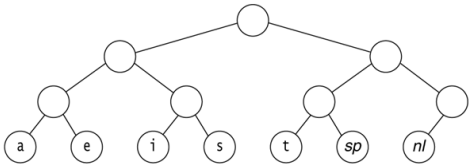
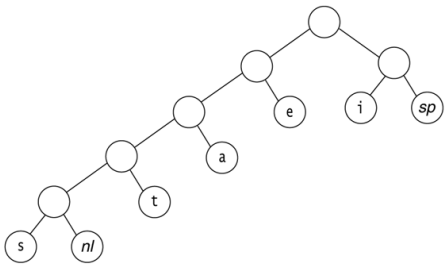


figure 12.2
Representation of the original code by a tree

Huffman Encoding

Prefix-free codes: no codeword is a prefix of another codeword

figure 12.4
An optimal prefix code tree



Huffman Encoding

figure 12.5
Optimal prefix code

Character	Code	Frequency	Total Bits
a	001	10	30
e	01	15	30
i	10	12	24
s	00000	3	15
t	0001	4	16
sp	11	13	26
nl	00001	1	5
Total			146

What is the improvement over the fixed-length encoding? It used 174 bits.

Huffman's Algorithm

1. Initialize *n* one-node trees with alphabet characters and the tree weights with their frequencies.

figure 12.6
Initial stage of Huffman's algorithm

Huffman's Algorithm

2. Join two binary trees with smallest weights into one (as left and right subtrees) and make its weight equal the sum of the weights of the two trees.

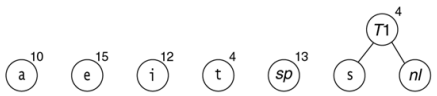


figure 12.7
Huffman's algorithm after the first merge

Huffman's Algorithm

Continue step (2) until there is one tree left.

figure 12.8
Huffman's algorithm after the second merge

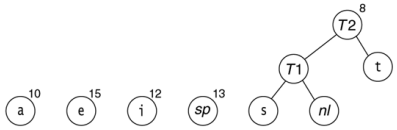
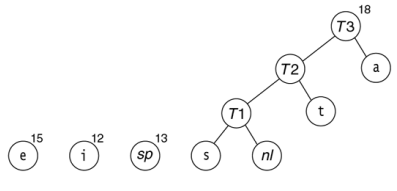
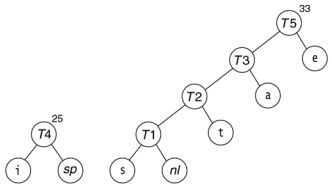


figure 12.9
Huffman's algorithm after the third merge



Huffman's Algorithm

figure 12.11
Huffman's algorithm after the fifth merge



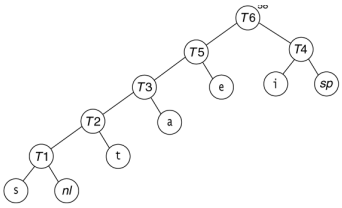


figure 12.12
Huffman's algorithm after the final merge