CSE 326 Data Structures

Dave Bacon

It's the Final Countdown…Dah Dah Dah Dah

Logisitics

- Homework 7 will be due Wed, March 7

- Read Chapter 9 of Weiss, Chapter 10.1, 10.3

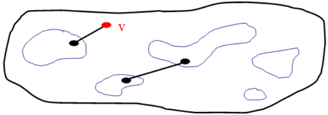
- Monday: Huffman Coding, Tablet Evaluations, Class Evaluations
- Wednesday: Final Review
- Friday: Games and NP completeness

- **Final:** Thursday March 15, 8:30-10:20 MGH 231

Kruskal's MST Algorithm

Idea: Grow a forest out of edges that do not create a cycle. Pick an ***edge with the smallest weight.***

G=(V,E)



Kruskal's Algorithm for MST

An *edge-based* greedy algorithm

Builds MST by greedily adding edges

1. Initialize with
   - empty MST
   - all vertices marked unconnected
   - all edges unmarked
2. While there are still unmarked edges
   a. Pick the lowest cost edge `(u,v)` and mark it
   b. If `u` and `v` are not already connected, add `(u,v)` to the MST and mark `u` and `v` as connected to each other

Doesn't it sound familiar?

Kruskal's Algorithm: Correctness

It clearly generates a spanning tree. Call it T_K.

Suppose T_K is *not* minimum:

- Pick another spanning tree T_{min} with *lower cost* than T_K
- Pick the smallest edge $e_1=(u,v)$ in T_K that is **not** in T_{min}
- T_{min} already has a path p in T_{min} from u to v
  - $\Rightarrow$ Adding e_1 to T_{min} will create a cycle in T_{min}
- Pick an edge e_2 in p that Kruskal's algorithm considered *after* adding e_1 (must exist: u and v unconnected when e_1 considered)
  - $\Rightarrow$ $\text{cost}(e_2) \geq \text{cost}(e_1)$
  - $\Rightarrow$ can replace e_2 with e_1 in T_{min} without increasing cost!
- Keep doing this until T_{min} is identical to T_K
  - $\Rightarrow$ T_K must also be minimal – contradiction!

Single-Source Shortest Path

- Given a graph **G** = (V, E) and a single distinguished vertex **s**, find the shortest weighted path from **s** to every other vertex in **G**.

All-Pairs Shortest Path:

- Find the shortest paths between all pairs of vertices in the graph.
- How?

Analysis

- Total running time for Dijkstra's:

  $O(|V|^2 + |E|)$ (linear scan)

  $O(|V| \log |V| + |E| \log |V|)$ (heaps)

What if we want to find the shortest path from each point to ALL other points?

Dynamic Programming

Algorithmic technique that systematically records the answers to sub-problems in a table and re-uses those recorded results (rather than re-computing them).

Simple Example: Calculating the Nth Fibonacci number.

Fib(N) = Fib(N-1) + Fib(N-2)

Dynamic Programming

Simple Example: Calculating the Nth Fibonacci number.

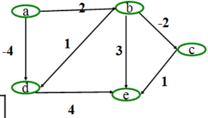
$$Fib(N) = Fib(N-1) + Fib(N-2)$$

Floyd-Warshall

```
for (int k = 1; k =< V; k++)
 for (int i = 1; i =< V; i++)
  for (int j = 1; j =< V; j++)
   if ( ( M[i][k]+ M[k][j] ) < M[i][j] )
     M[i][j] =  M[i][k]+ M[k][j]
```

Invariant: After the kth iteration, the matrix includes the shortest paths for all pairs of vertices (i,j) containing only vertices 1..k as intermediate vertices

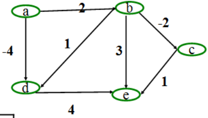
Initial state of the matrix:



	a	b	c	d	e
a	0	2	∞	-4	∞
b	∞	0	-2	1	3
c	∞	∞	0	∞	1
d	∞	∞	∞	0	4
e	∞	∞	∞	∞	0

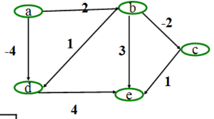
$$M[i][j] = \min(M[i][j], M[i][k] + M[k][j])$$

Floyd-Warshall



	a	b	c	d	e
a	0				
b		0			
c			0		
d				0	
e					0

Floyd-Warshall - for All-pairs shortest path



	a	b	c	d	e
a	0	2	0	-4	0
b	-	0	-2	1	-1
c	-	-	0	-	1
d	-	-	-	0	4
e	-	-	-	-	0

Final Matrix Contents

Data Compression: Huffman Coding

10.1.2 in Weiss (p.395)

Data Compression

- **Lossless** compression $X = X'$
- **Lossy** compression $X \mathrel{!=} X'$
- **Compression Ratio** $|X|/|Y|$
  - Where $|X|$ is the # of bits in X.

Lossy Compression

- Some data is lost, but not too much.

Standards:

- JPEG (Joint Photographic Experts Group)
  – stills
- MPEG (Motion Picture Experts Group)
  – Audio and video
- MP3 (MPEG-1, Layer 3)

Lossless Compression

- No data is lost.

Standards:

- Gzip, Unix compress, zip, GIF, Morse code

Lossless Compression of text

ASCII = fixed 8 bits per character

Example: “hello there”

- 11 characters * 8 bits = 88 bits

Can we encode this message using fewer bits?

Huffman Coding

- Uses *frequencies* of symbols in a string to build a **prefix code**.
- **Prefix Code** – no code in our encoding is a prefix of another code.

Letter	code
a	0
b	100
c	101
d	11

Decoding a Prefix Code

```
Loop
  start at root of tree
      loop
           if bit read = 1 then go right
           else, go left
      until node is a leaf
      Report character found!
Until end of the message
```

Decode: 11100010100110

Letter	code
a	0
b	100
c	101
d	11

Huffman Trees

Cost of a Huffman Tree containing n symbols

$$C(T) = p_1*r_1+p_2*r_2+p_3*r_3+....+ p_n*r_n$$

Where:

p_i = the probability that a symbol occurs

r_i = the length of the path from the root to the node

Creating Huffman Trees?

Be Greedy!