

## Chapter 9: Greedy Algorithms

From the first day to this, sheer greed was the driving spirit of civilization. (Friedrich Engels)

Introduction . . . . .	2
Example of Making Change . . . . .	3
<b>Prim's Algorithm</b> . . . . .	<b>4</b>
Minimum Spanning Trees . . . . .	4
Prim's Algorithm . . . . .	5
Example of Prim's Algorithm . . . . .	6
Example of Prim's Algorithm, Continued . . . . .	7
Comments on Prim's Algorithm . . . . .	8
<b>Kruskal's Algorithm</b> . . . . .	<b>9</b>
Kruskal's Algorithm . . . . .	9
Example of Kruskal's Algorithm . . . . .	10
Example of Kruskal's Algorithm, Continued . . . . .	11
Comments on Kruskal's Algorithm . . . . .	12
<b>Dijkstra's Algorithm</b> . . . . .	<b>13</b>
Single-Source Shortest Paths . . . . .	13
Dijkstra's Algorithm . . . . .	14
Example of Dijkstra's Algorithm . . . . .	15
Example of Dijkstra's Algorithm, Continued . . . . .	16
Comments on Dijkstra's Algorithm . . . . .	17
<b>Huffman Trees</b> . . . . .	<b>18</b>
Encodings . . . . .	18
Optimal Prefix Codes . . . . .	19

Huffman's Algorithm . . . . .	20
Example of Huffman's Algorithm . . . . .	21
Example of Huffman's Algorithm, Continued . . . . .	22

## Introduction

Greedy algorithms is a technique for solving problems with the following properties:

- The problem is an optimization problem, to find the solution that minimizes or maximizes some value (cost/profit).
- The solution can be constructed in a sequence of steps/choices.
- For each choice point:
  - The choice must be feasible.
  - The choice looks as good or better than alternatives.
  - The choice cannot be revoked.

CS 3343 Analysis of Algorithms

Chapter 9: Slide – 2

## Example of Making Change

- Suppose we want to give change of a certain amount (say 24 cents).
- We would like to use fewer coins rather than more.
- We can make a solution by repeatedly choosing a coin  $\leq$  to the current amount, resulting in a new amount.
- The greedy solution is to always choose the largest coin value possible (for 24 cents: 10, 10, 1, 1, 1, 1).
- If there were an 8-cent coin, the greedy algorithm would not be optimal (but not bad).

CS 3343 Analysis of Algorithms

Chapter 9: Slide – 3

## Prim's Algorithm

4

### Minimum Spanning Trees

- A *spanning tree* of a connected graph is a tree containing all the vertices.
- A *minimum spanning tree* of a weighted graph is a spanning tree with the smallest weight.
- The *weight* of a spanning tree is the sum of the edge weights.

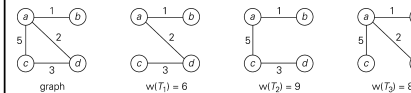


FIGURE 9.1 Graph and its spanning trees;  $T_1$  is the minimum spanning tree

CS 3343 Analysis of Algorithms

Chapter 9: Slide – 4

### Prim's Algorithm

**algorithm** *Prim*( $G$ )

// Returns the MST by Prim's Algorithm

// Input: A weighted connected graph  $G = (V, E)$

// Output: Set of edges comprising a MST

$V_T \leftarrow \{\text{any vertex in } G\}$

$E_T \leftarrow \emptyset$

**for**  $i \leftarrow 1$  **to**  $|V| - 1$  **do**

$e \leftarrow$  minimum-weight edge  $(v, u)$   
with  $v \in V_T$  and  $u \in V - V_T$

$V_T \leftarrow V_T \cup \{u\}$

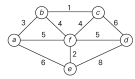
$E_T \leftarrow E_T \cup \{e\}$

**return**  $E_T$

CS 3343 Analysis of Algorithms

Chapter 9: Slide – 5

## Example of Prim's Algorithm



Tree vertices	Remaining vertices	Illustration
a(-, -)	<b>b(a, 3)</b> c(-, ∞) d(-, ∞) e(a, 6) f(a, 5)	
<b>b(a, 3)</b>	<b>e(b, 1)</b> d(-, ∞) e(a, 6) f(b, 4)	

CS 3343 Analysis of Algorithms

Chapter 9: Slide - 6

## Example of Prim's Algorithm, Continued

c(b, 1)	d(c, 6) e(a, 6) <b>f(b, 4)</b>	
f(b, 4)	d(f, 5) <b>e(f, 2)</b>	
e(f, 2)	<b>d(f, 5)</b>	
d(f, 5)		

**FIGURE 9.2** Application of Prim's algorithm. The parenthesized labels of a vertex in the middle column indicate the nearest tree vertex and edge weight; selected vertices and edges are shown in bold.

CS 3343 Analysis of Algorithms

Chapter 9: Slide - 7

## Comments on Prim's Algorithm

- This algorithm repeatedly chooses the smallest-weight edge from the tree so far to the other vertices.
- If a spanning tree has a weightier edge between  $V_T$  and  $V - V_T$ , it can be improved by replacing it with  $e$ .
- We can put  $V_T$  edges into a priority queue, then dequeue edges until one goes between  $V_T$  and  $V - V_T$ .
- Prim's Algorithm using a binary heap to implement a priority queue is  $O(E \log E) = O(E \log V)$ .

CS 3343 Analysis of Algorithms

Chapter 9: Slide - 8

## Kruskal's Algorithm

9

### Kruskal's Algorithm

**algorithm** *Kruskal*( $G$ )

// Returns the MST by Kruskal's Algorithm

// Input: A weighted connected graph  $G = (V, E)$

// Output: Set of edges comprising a MST

sort the edges  $E$  by their weights

$E_T \leftarrow \emptyset$

**while**  $|E_T| + 1 < |V|$  **do**

$e \leftarrow$  next edge in  $E$

**if**  $E_T \cup \{e\}$  does not have a cycle **then**

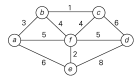
$E_T \leftarrow E_T \cup \{e\}$

**return**  $E_T$

CS 3343 Analysis of Algorithms

Chapter 9: Slide - 9

## Example of Kruskal's Algorithm



Tree edges	Sorted list of edges	Illustration
bc	bc 1, ef 2, ab 3, bf 4, cf 4, af 5, df 5, ac 6, ce 6, de 8	
bc, ef	bc 1, ef 2, ab 3, bf 4, cf 4, af 5, df 5, ac 6, ce 6, de 8	

CS 3343 Analysis of Algorithms

Chapter 9: Slide – 10

## Example of Kruskal's Algorithm, Continued

ef	bc 1, ef 2, ab 3, bf 4, cf 4, af 5, df 5, ac 6, ce 6, de 8	
ab	bc 1, ef 2, ab 3, bf 4, cf 4, af 5, df 5, ac 6, ce 6, de 8	
bf	bc 1, ef 2, ab 3, bf 4, cf 4, af 5, df 5, ac 6, ce 6, de 8	
df	bc 1, ef 2, ab 3, bf 4, cf 4, af 5, df 5, ac 6, ce 6, de 8	

FIGURE 9.4 Application of Kruskal's algorithm. Selected edges are shown in bold.

CS 3343 Analysis of Algorithms

Chapter 9: Slide – 11

## Comments on Kruskal's Algorithm

- This algorithm repeatedly chooses the smallest-weight edge that does not form a cycle.
- Kruskal's Algorithm is  $O(E \log V)$  using efficient cycle detection.
- Disjoint Set Implementation for Detecting Cycles:
  - Put each vertex into a singleton set.
  - $e$  is chosen if its vertices are in different sets.
  - When  $e$  is chosen, the two sets are unioned.

CS 3343 Analysis of Algorithms

Chapter 9: Slide – 12

## Dijkstra's Algorithm

### Single-Source Shortest Paths

- The *single-source shortest paths problem* finds the shortest paths from a given vertex (the *source vertex*) to all the other vertices.
- Dijkstra's algorithm can be used if there are no negative edges.
- The Bellman-Ford algorithm is used if there are negative edges.
- Floyd's algorithm is used for all-pairs shortest paths if there are no negative weights.

CS 3343 Analysis of Algorithms

Chapter 9: Slide – 13

### Dijkstra's Algorithm

**algorithm** *Dijkstra*( $G, s$ )

// Solves SSSP by Dijkstra's Algorithm

// Input: A weighted connected graph  $G = (V, E)$

// with no negative weights, and source vertex  $v$

// Output: The length and path from  $s$  to every  $v$

**for**  $v \in V$  **do**

$d_v \leftarrow \infty$ ;  $p_v \leftarrow \text{null}$ ; unmark  $v$

$d_s \leftarrow 0$

**while**

$u \leftarrow$  unmarked vertex with smallest  $d_u$ ; mark  $u$

**for each**  $v$  adjacent from  $u$  **do**

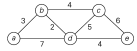
**if**  $v$  is unmarked **and**  $d_u + w(u, v) < d_v$  **then**

$d_v \leftarrow d_u + w(u, v)$ ;  $p_v \leftarrow u$

CS 3343 Analysis of Algorithms

Chapter 9: Slide – 14

## Example of Dijkstra's Algorithm



Tree vertices	Remaining vertices	Illustration
a(-, 0)	<b>b(a, 3)</b> c(-, ∞) d(a, 7) e(-, ∞)	
b(a, 3)	c(b, 3+4) <b>d(b, 3+2)</b> e(-, ∞)	
d(b, 5)	<b>e(b, 7)</b> e(d, 5+4)	

CS 3343 Analysis of Algorithms

Chapter 9: Slide – 15

## Example of Dijkstra's Algorithm, Continued



The shortest paths (identified by following nonnumeric labels backward from a destination vertex in the left column to the source) and their lengths (given by numeric labels of the tree vertices) are

from a to b: a - b      of length 3  
 from a to d: a - b - d    of length 5  
 from a to c: a - b - c    of length 7  
 from a to e: a - b - d - e of length 9

FIGURE 9.10 Application of Dijkstra's algorithm. The next closest vertex is shown in bold.

CS 3343 Analysis of Algorithms

Chapter 9: Slide – 16

## Comments on Dijkstra's Algorithm

- This algorithm repeatedly processes the closest unmarked vertex  $u$ .
- When  $u$  is marked, the SP to  $u$  is known.  
 Basis: First value of  $u$  is  $s$ , whose SP is known.  
 Induction: Assume SPs to previously marked vertices are known.  
 SPs to unmarked vertices must start with some marked vertices.  
 $u$  is closest unmarked vertex.
- Dijkstra's Algorithm using a binary heap to implement a priority queue is  $O(E \log V)$ .

CS 3343 Analysis of Algorithms

Chapter 9: Slide – 17

## Huffman Trees

18

### Encodings

- We can represent an  $n$ -character set by assigning a *codeword*, a bit string, to each character.
- *Fixed-length encoding* (e.g., ASCII) assigns  $m$ -bit strings to each character,  $m \geq \log_2 n$ .
- *Variable-length encoding* (e.g., Morse code) assigns different lengths.
- To avoid ambiguity, use *prefix codes*, i.e., no codeword is a prefix of another codeword.

CS 3343 Analysis of Algorithms

Chapter 9: Slide – 18

### Optimal Prefix Codes

- Problem: If the probabilities of the characters are known, what is the best binary prefix code?
- "Best" means the expected length of a codeword, sum of probabilities times code length.
- Any binary tree with edges labeled with 0's and 1's and characters assigned to its leaves yields a prefix code.
- The two smallest probabilities must have the same length.
- If made siblings, the size of the problem decreases by one.

CS 3343 Analysis of Algorithms

Chapter 9: Slide – 19

## Huffman's Algorithm

**algorithm** *Huffman*( $W[0..n-1]$ )

// Computes optimal prefix code.

// Input: Array  $W$  of character probabilities

// Output: The Huffman tree.

**for**  $i \leftarrow 0$  **to**  $n - 1$  **do**

$T \leftarrow$  node labeled with character  $i$

$T.weight \leftarrow W[i]$ ; add  $T$  to set  $S$

**for**  $i \leftarrow 0$  **to**  $n - 2$  **do**

$L, R \leftarrow$  remove two min. weight trees from  $S$

$T \leftarrow$  node with children  $L$  and  $R$

$T.weight \leftarrow L.weight + R.weight$

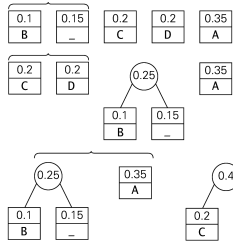
add  $T$  to set  $S$

**return**  $T$

CS 3343 Analysis of Algorithms

Chapter 9: Slide – 20

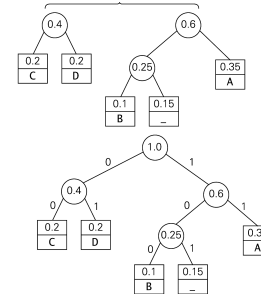
## Example of Huffman's Algorithm



CS 3343 Analysis of Algorithms

Chapter 9: Slide – 21

## Example of Huffman's Algorithm, Continued



**FIGURE 9.11** Example of constructing a Huffman coding tree  
CS 3343 Analysis of Algorithms

Chapter 9: Slide – 22