

CPE231RC

Algorithms

Lecture8

Greedy Technique

Dr. Prapong Prechaprapranwong



Computer Engineering



King Mongkut's
University of
Technology
Thonburi

Greedy Technique

The greedy approach suggests constructing a solution through a sequence of steps until a complete solution by each step the choice made must be:

- feasible - it must satisfy the constraints
- locally optimal - it must be the best local choice among all feasible choices
- irrevocable - once made, it cannot be changed
- greedy – choose the most benefit choice

- Not guarantee globally solution but still be useful for an approximate solution

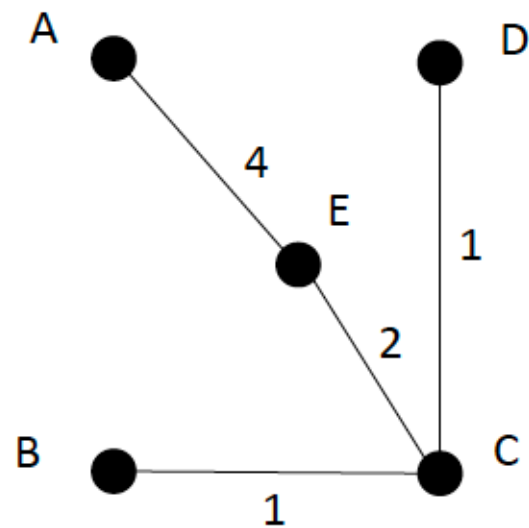
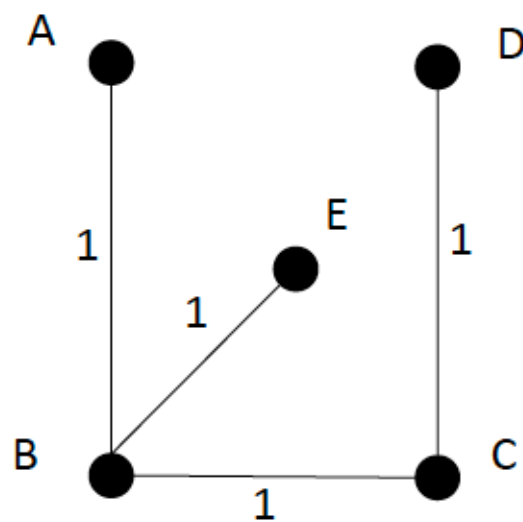
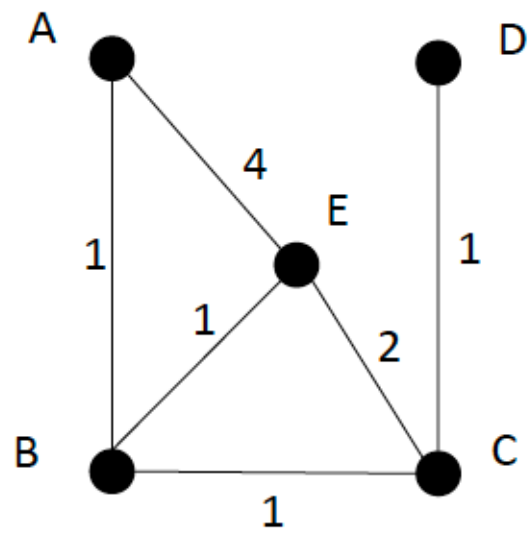
Examples:

- **Minimum spanning Tree:** Prim's and Kruskal's algorithms
- **Single-source shortest path:** Dijkstra's algorithm
- **Huffman code (data compression):** Huffman tree

Minimum Spanning Tree

Spanning tree of a connected graph G : a connected acyclic subgraph of G that includes all of G 's vertices

Minimum Spanning Tree of a weighted, connected graph G : a spanning tree of G of the minimum total weight



Both tree are spanning tree
Left tree is minimum spanning tree

Minimum Spanning Tree

Prim's Algorithm constructs a minimum spanning tree through a sequence of expanding subtree. On each iteration the spanning tree is expanded by an edge of the smallest weight

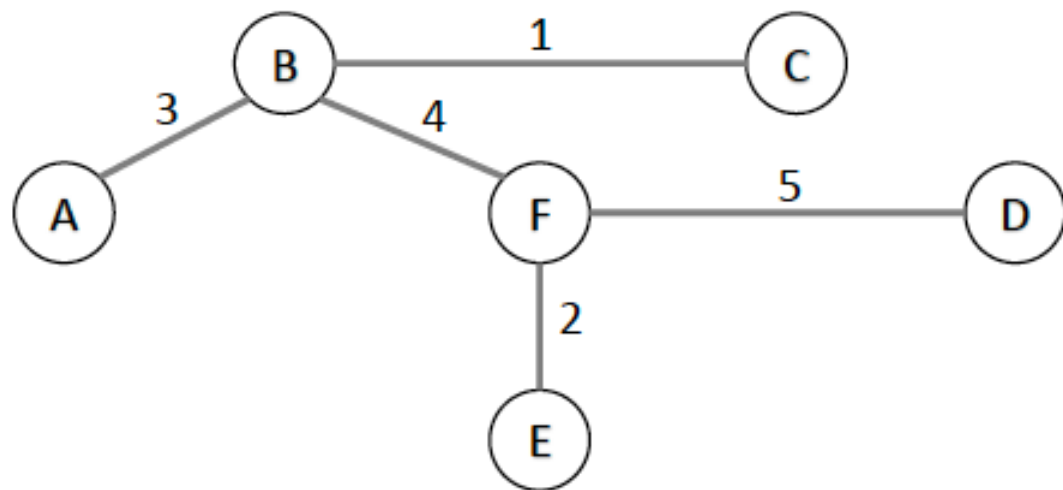
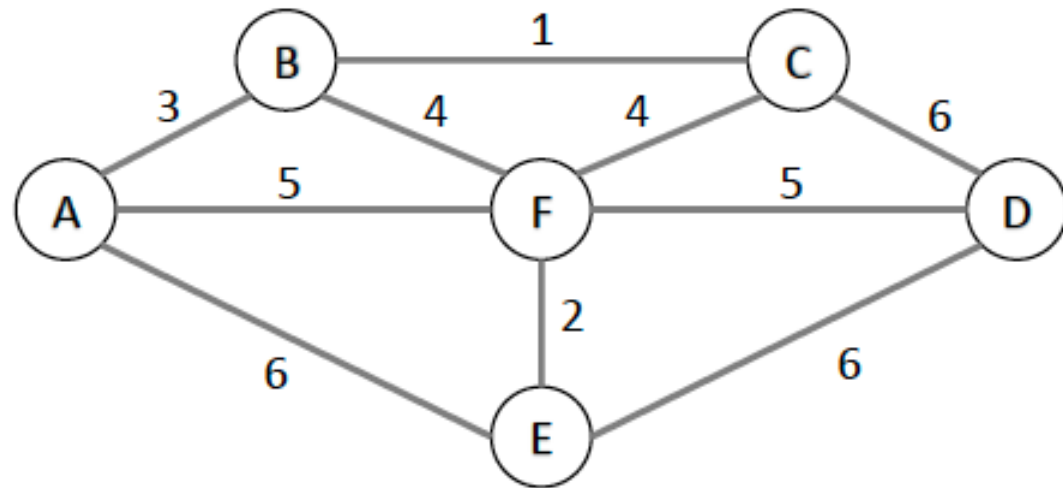
Step by step

- 1) Add arbitrary vertex (node) from G in T .
- 2) Grow T by adding a vertex from G closest to those already in T with the corresponding edge.
- 3) Repeat 2) until all the vertices in G are added in T .
- 4) The algorithm does not necessarily give a unique MST.

Algorithm Prim

- 1: Input: A weighted connected graph $G = \langle V, E \rangle$
 - 2: Output: E_T , the set of edges composing a minimum spanning tree of G
 - 3: $V_T \leftarrow \{v_0\}$
 - 4: $E_T \leftarrow \phi$
 - 5: **for** $i \leftarrow 1$ to $|V| - 1$ **do**
 - 6: Find a min-weight edge $e^* = (v^*, u^*)$ among $v \in V_T, u \in V - V_T$
 - 7: $V_T \leftarrow V_T \cup \{u^*\}$
 - 8: $E_T \leftarrow E_T \cup \{e^*\}$ $O(|E| \log |V|)$
 - 9: **return** E_T
-

Example: MST by using Prim's Algorithm



Tree vertices

Remain vertices

A(,)

B(A,3) C(-,∞) D(-,∞) E(A,6) F(A,5)

B(A,3)

C(B,1) D(-,∞) E(A,6) F(B,4)

C(B,1)

D(C,6) E(A,6) **F(B,4)**

F(B,4)

D(F,5) **E(F,2)**

E(F,2)

D(F,5)

D(F,5)

MST = [A→B→C→F→E→D] with total weight = 15

Minimum Spanning Tree

Kruskal's Algorithm constructs a minimum spanning tree as expanding sequence of subgraphs that are always acyclic with the smallest weight edges

Step by step

- 1) Sort the edges in the ascending order. $T = \phi$
- 2) Select an edge with smallest weight and add to T .
- 3) Select the edge (u, v) with smallest weight such that T is acyclic if added.
- 4) Add the edge in Step 3 to T .
- 5) Repeat step 3) until all the nodes are included in T .



Kruskal's algorithm looks easier than Prim's but harder to implement

ALGORITHM Kruskal (G)

//Kruskal's algorithm for constructing a minimum spanning tree

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

//Output: E_T , the set of edges composing a minimum spanning tree of G sort E in nondecreasing order of the edge weights $w(e_{i_1}) \leq \dots \leq w(e_{i_{|E|}})$

$E_T \leftarrow \emptyset$; $ecounter \leftarrow 0$ //initialize the set of tree edges and its size

$k \leftarrow 0$ //initialize the number of processed edges

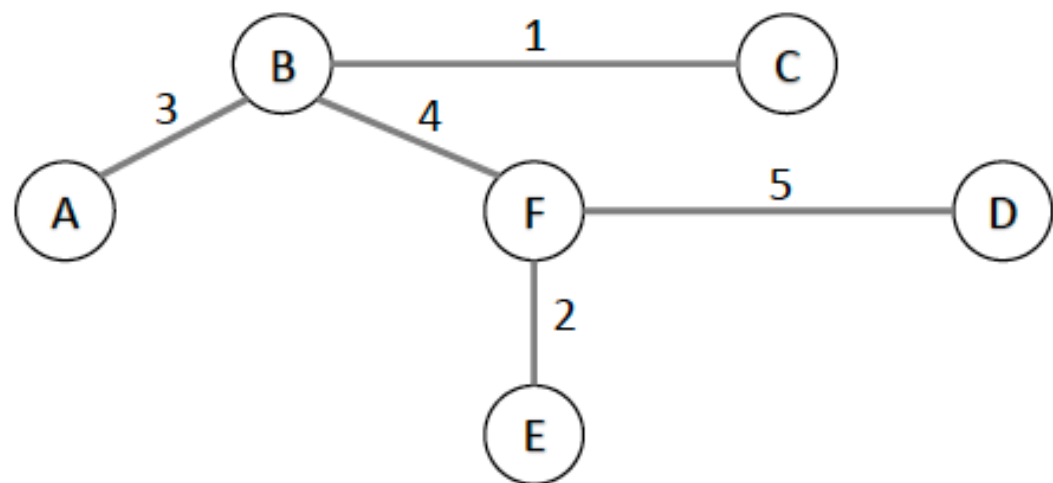
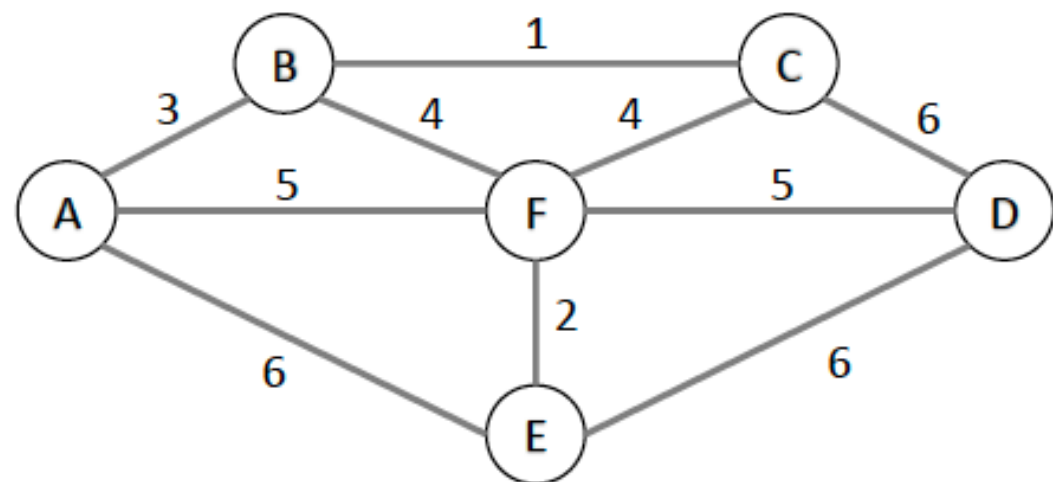
while $ecounter < |V| - 1$ **do**

$k \leftarrow k + 1$

 if $E_T \cup \{e_{ik}\}$; $ecounter \leftarrow ecounter + 1$

return E_T

Example: MST by using Kruskal's Algorithm



Tree edges	Sorted list of edges
	BC(1) EF(2) AB(3) BF(4) CF(4) AF(5) DF(5) AE(6) CD(6) DE(8)
BC(1)	EF(2) AB(3) BF(4) CF(4) AF(5) DF(5) AE(6) CD(6) DE(8)
EF(2)	AB(3) BF(4) CF(4) AF(5) DF(5) AE(6) CD(6) DE(8)
AB(3)	BF(4) CF(4) AF(5) DF(5) AE(6) CD(6) DE(8)
BF(4)	CF(4) AF(5) DF(5) AE(6) CD(6) DE(8)
DF(5)	

MST = [A→B→C→F→E→D] with total weight = 15

Single Source Shortest paths Problem

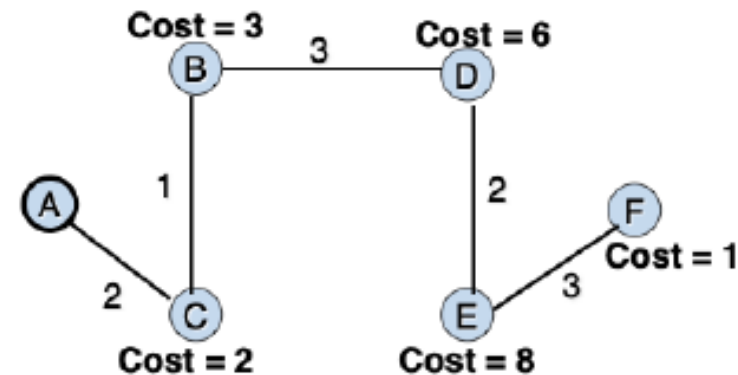
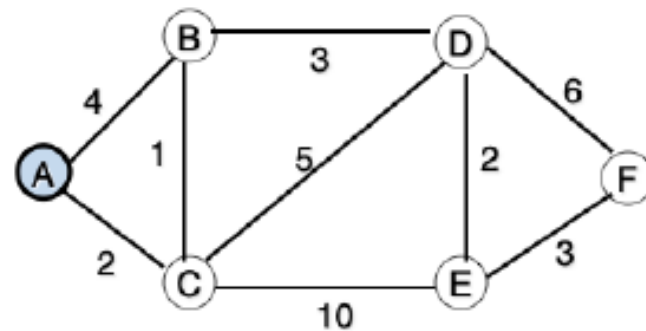
Dijkstra's Algorithm find shortest paths from a vertex (called "source") to all its other vertices in a graph in order of their distance from a given source.

Similar to **Prim's algorithm**

- Dijkstra's algorithm compares path lengths and add edge weights while Prim's compares just the edge weights

Application of the shortest path

- transportation planning
- packet routing in communication networks
- shortest paths in social networks
- pathfinder in games
- ...



ALGORITHM *Dijkstra* (G,s)

//Dijkstra's algorithm for single-source shortest paths

//Input: A weighted connected graph $G = (V,E)$ with nonnegative weights and its vertex s

//Output: The length d_v of a shortest path from s to v and its penultimate vertex p_v for every vertex v in V

Initialize(Q) // initialize priority queue to empty

for every vertex v in V

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

Insert(Q,v,d_v) // initialize vertex priority in the priority queue

$d_s \leftarrow 0$; Decrease(Q,s,d_s) // update priority of s with d_s

$V_T \leftarrow \emptyset$

for $i \leftarrow 0$ **to** $|V| - 1$ **do**

$u^* \leftarrow \text{DeleteMin}(Q)$ // delete the minimum priority element

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

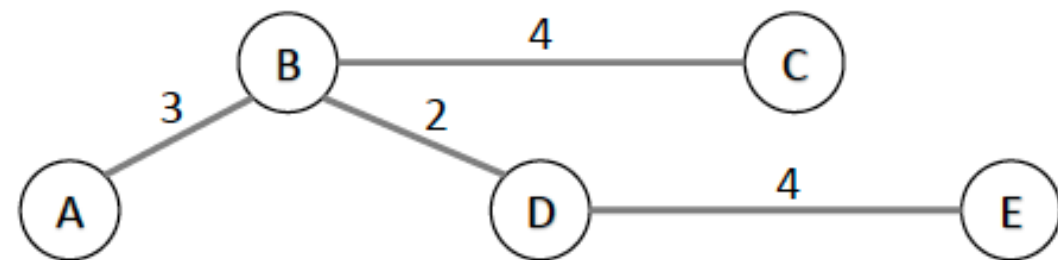
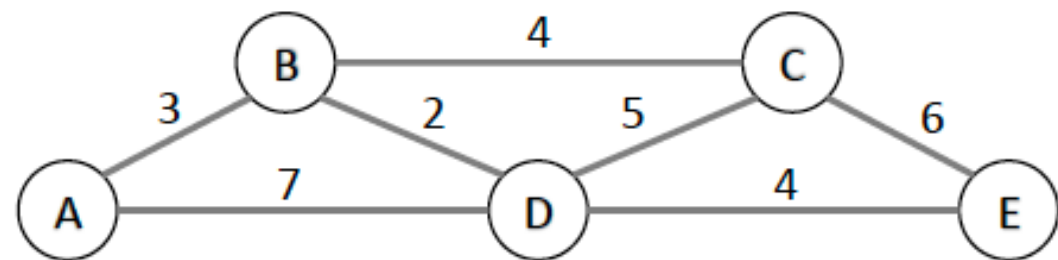
for every vertex u in $V - V_T$ that is adjacent to u^* **do**

if $d_{u^*} + w(u^*,u) < d_u$

$d_u \leftarrow d_{u^*} + w(u^*,u)$; $p_u \leftarrow u^*$

Decrease(Q,u,d_u)

Example: Single source shortest path by using Dijkstra's Algorithm



Tree vertices	Remain vertices
A(,)	B(A,3) C(-, ∞) D(A,7) E(-, ∞)
B(A,3)	C(B,3+4) D(B,3+2) E(-, ∞)
D(B,5)	C(B,7) E(D,5+4)
C(B,7)	E(D,9)
E(D,9)	

from a to b: a – b of length 3

from a to d: a – b – d of length 5

from a to c: a – d – c of length 7

from a to e: a – b – d – e of length 9

Practice I Knapsack Problem with Greedy Technique

Try to solve Knapsack Problem again with different technique

10 min

Data Compression Problem

ASCII is "American Standard Code for Information Interchange" , ASCII code is the numerical representation of a character

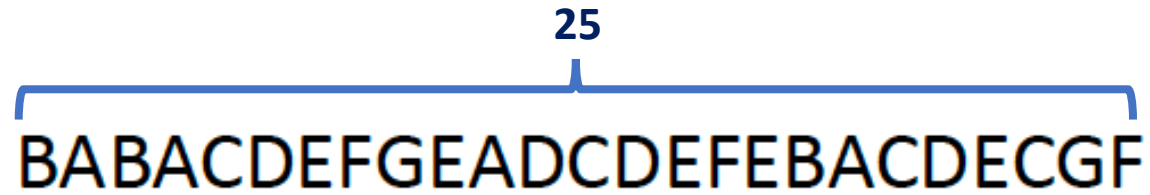
- ASCII characters use *fixed-length codes* (8 bits = 256 characters ?) e.g.,
A = 65 = 01000001 *codeword*
- For a text file with 1000 characters, file size is 8 x 1000 bits
- Suppose the file contains only 7 unique characters 'A' to 'G' , we can represent by _____ bits

Code	Char	Code	Char	Code	Char	Code	Char	Code	Char	Code	Char
32	[space]	48	0	64	@	80	P	96	`	112	p
33	!	49	1	65	A	81	Q	97	a	113	q
34	"	50	2	66	B	82	R	98	b	114	r
35	#	51	3	67	C	83	S	99	c	115	s
36	\$	52	4	68	D	84	T	100	d	116	t
37	%	53	5	69	E	85	U	101	e	117	u
38	&	54	6	70	F	86	V	102	f	118	v
39	'	55	7	71	G	87	W	103	g	119	w
40	(56	8	72	H	88	X	104	h	120	x
41)	57	9	73	I	89	Y	105	i	121	y
42	*	58	:	74	J	90	Z	106	j	122	z
43	+	59	;	75	K	91	[107	k	123	{
44	,	60	<	76	L	92	\	108	l	124	
45	-	61	=	77	M	93]	109	m	125	}
46	.	62	>	78	N	94	^	110	n	126	~
47	/	63	?	79	O	95	_	111	o	127	[backspace]

Example:

- How large of the following message representing by ASCII ?
- How large of the following message representing by your own fixed-length size codes ?
- What is the compression ratio?

25
BABACDEFGEADCDEFEBACDECGF



Variable-length encoding assigns codewords of different lengths to different symbols

Example:

- How large of the following message representing by variable-length code?
- What is the compression ratio?

BABACDEFGEADCDEFEBACDECGF

Symbol
A
B
C
D
E
F
G

Huffman's Algorithm

Step 1

Initialize n one-node trees and label them with the symbols of the alphabet given. Record the frequency of each symbol in its tree's root to indicate the tree's weight

Step 2

Repeat finding two trees with the smallest weight (Greedy Technique). Make them the left and right subtree of a new tree and record the sum of their weights in the root of the new tree as its weight

A Tree constructed by this algorithm is called a **Huffman tree** and the codes is called a **Huffman code**

Example:

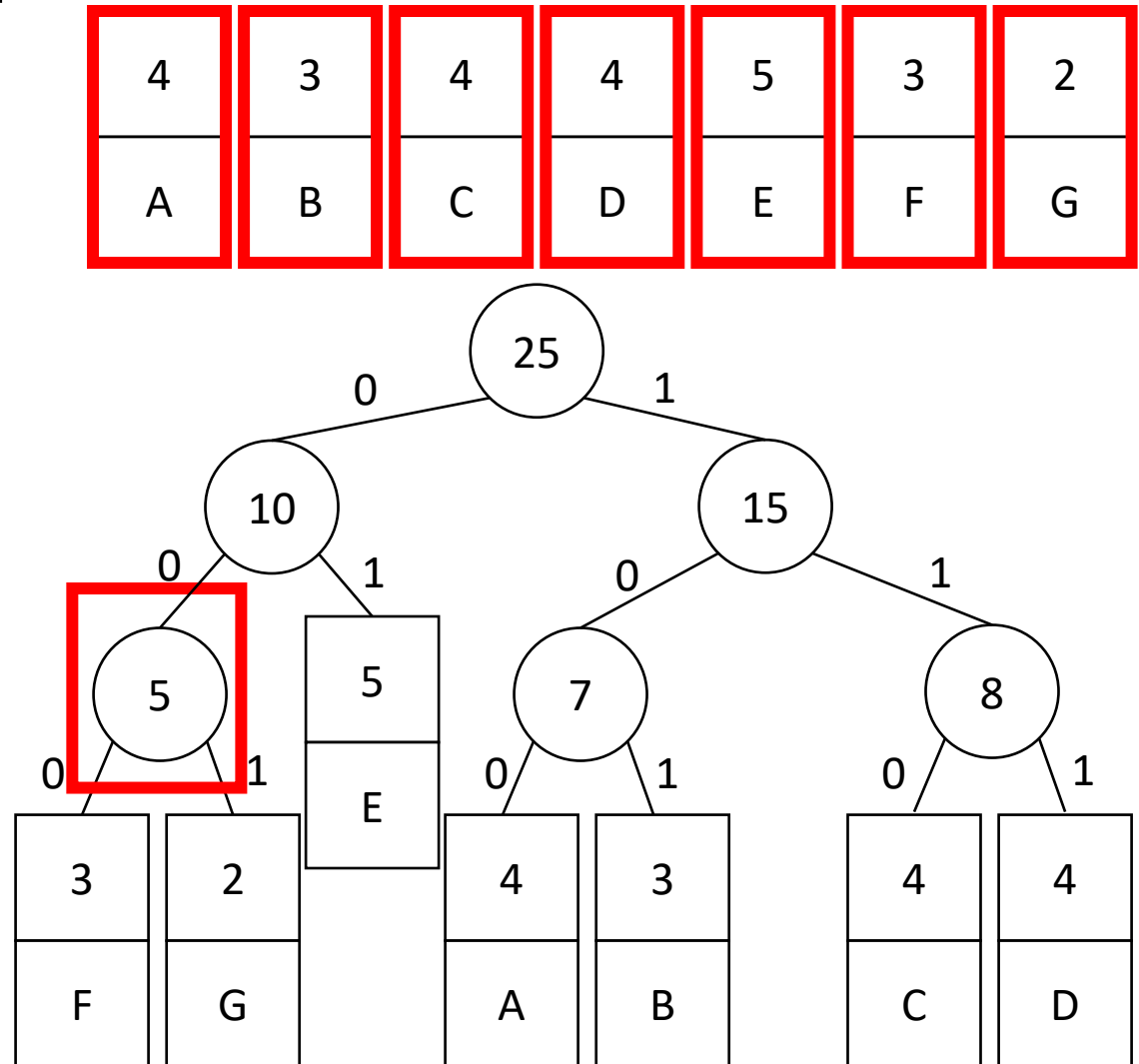
- Construct the Huffman tree representing this message and What is the Huffman code table ?
- What is the compression ratio?

BABACDEFGEADCDEFEBACDEC GF

Symbol	Frequency	Codewords
A	4	100
B	3	101
C	4	110
D	4	111
E	5	01
F	3	000
G	2	001

Total: $3 \times 4 + 3 \times 3 + 3 \times 4 + 3 \times 4 + 2 \times 5 + 3 \times 3 + 3 \times 2 = 70$

compression ratio = $(200 - 70) / 200 = 0.65$



Practice II Create the Huffman Tree for this message

message : "a greedy algorithm is a simple and intuitive algorithm that is used in optimization problems"

30 min

Summary

- Greedy algorithm use information available at each phase to makes decisions without considering the globally information
- The solution possibly is not the best solution for every problem
- However, it is simple and can be quick-win solution