Shefali Emmanuel

8.4

#1 Apply Warshall's Algorithm = compute the transative closure

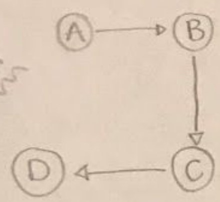
Adjacency Matrix

$$\begin{array}{c|cccc} & A & B & C & D \\ \hline A & 0 & 1 & 0 & 0 \\ B & 0 & 0 & 1 & 0 \\ C & 0 & 0 & 0 & 1 \\ D & 0 & 0 & 0 & 0 \end{array}$$

Transative Closure

$$\begin{array}{c|cccc} & A & B & C & D \\ \hline A & 0 & 1 & 1 & 1 \\ B & 0 & 0 & 1 & 1 \\ C & 0 & 0 & 0 & 1 \\ D & 0 & 0 & 0 & 0 \end{array}$$

Digraph



there is a path in the directed graph
check for each A, B, C, D
ex: A → B ✓
→ C ✓
→ D ✓

#6

(A) Explain how Warshall's alg can be used to determine if a given digraph is a dag?
Is it a good alg for this question?

a digraph = dag if the transative closure has a 1 on the main diagional, which means that each element goes back to itself
which is cubic [specific to this algorithm]

this is a good algorithm for this problem since it gets the job done
however, according to its complexity it is not a great choice

(B) Is it a good idea to apply warshall to an undirected graph?

no, a dfs or bfs would be more efficent for its transative closure also it was intended to be used with a directed graph

Goal: find the ones = 1 outside the main diagional

Shefali Emmanuel

(#7) Solve the all-pairs shortest path problem = Flody's Algorithm

	A	B	C	D	E
A	0	2	∞	1	8
B	6	0	3	2	∞
C	∞	∞	0	4	∞
D	∞	∞	2	0	3
E	3	∞	∞	∞	0

=

0	2	3	1	4
6	0	3	2	5
10	12	0	4	7
6	8	2	0	3
3	5	6	4	0

9.1

(#3) Job Scheduling

n jobs of known duration $\underbrace{t_1, t_2, t_3, \dots t_n}_{\text{time to finish}}$ by a single processor

*any order

*one @ a time — create a schedule that finds min(waitTime + ExecutionTime)

Goal: find a schedule that minimizes total time spent by all the jobs in system

* time spent by 1 job = Σ $\underbrace{\text{time spent on 1 job}}_{\text{execution}}$ + waiting time

Design a greedy alg & does it alway yield the optimal sol?

for this case yes, most times they are very efficent

$\boxed{\text{step 1}}$ sort all (n) jobs from shortest → longest time to finish & execute (base case) $\{i_1, i_2, i_3, \dots i_n\}$

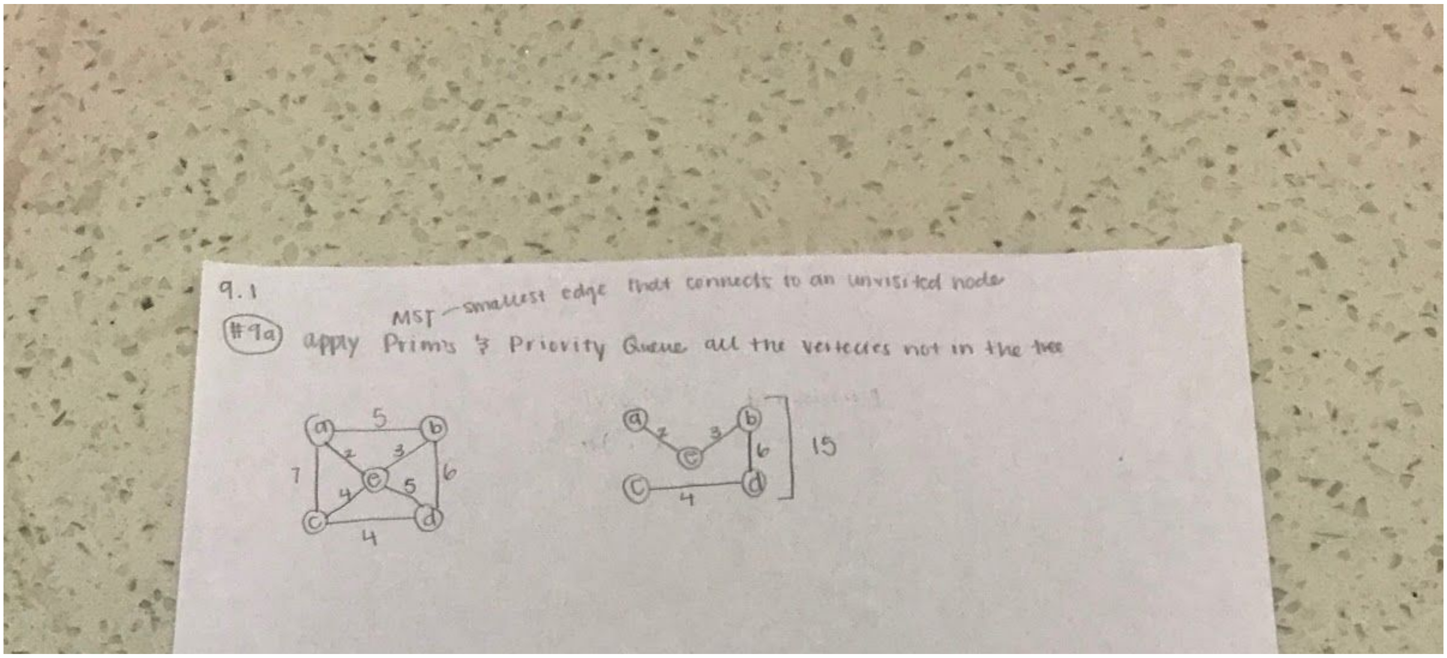
$\boxed{\text{Step 2}}$ if (the jobs are executing in increasing order of execution time) continue,

{

$\underbrace{t_{i_k} > t_{i_{k+1}}}$ then the total time can be decreased

AKA getMin (NwaitTime, NexecutionTime)

return a sorted list of these results from smallest to largest



Tree Vertices	Remaining Vertices
a(-,-)	b(a,5) c(a,7) d(-,∞) **e(a,2)**
e(a,2)	**b(e,3)** c(e,4) d(e,5)
b(e,3)	c(-,∞) **d(b,6)**
d(b,6)	**c(d,4)**
c(d,4)	

7.2

(#3) # of comparisons in binary text of 1000 zeros?

(a) 00001 $n=5$

12345

└ will keep shifting by one so $1000 - 4 = 996$

$n-1$

(b) 10000

└ sucessful

└ un sucessful

$5 \cdot 996 = 4980$

(c) 01010

└ sucessful

└ unsucessful

$2 \cdot 498 = 996$

all evens from 0....994

(#7) # of comparisons with boyer More "

(a) 996

(b) $5 \cdot 200 = 1000$

(c) $2 \cdot 249 = 498$

Shefali Emmanuel

7.2

(#2) Searching for genes in DNA using Horsepool's Alg

text = {A, C, G, T}

patt = gene / gene segment

(a) construct the shift table of chromosome 10

1 2 3 4 5 6 7 8 9 10

T C C T A T T C T T

C	T	C	A	G
G(C)	1	2	5	10

(b) Apply Horsepool to locate the pattern

Pattern: TCCTATTCTT

Text : TTATAGATCTCGTATTCTTTTATAGATCTCCTATTCTT

T → 2c 1s

C → 1c 2s

T → 2c 1s

A → 1c 5s

T → 8c 1s

T → 3c 1s

T → 3c 1s

A → 1c 5s

T → 2c 1s

C → 1c 2s

C → 1c 2s

T → 2c 1s

A → 1c 5s

T → 10c → stop

c = comparison

s = shift

11.3

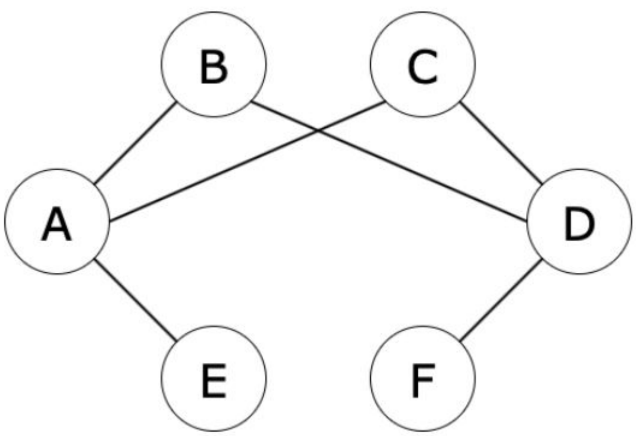
#9 (iii) Determine for a given graph $G = (V, \mathcal{E})$ and a positive integer $m \leq |V|$, whether G contains a independent set of size m or more $\rightarrow$ in NP

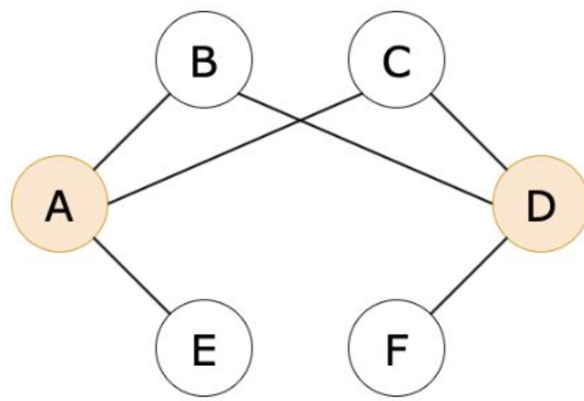
$S \subseteq V$ if there are no edges between vertices in S

a set is independent if and only if it is a clique in the compliment graph

Make S' = V-S (compliment)

a. S' = {A,D}





Therefore, this graph has an independent set

11.3 Continued to next page

Prove that a Vertex Cover can be reduced to a Clique

DEFINITION G has a clique of size k if and only if G' has a vertex cover of size |V|-k

undirected Graph $G = (\overset{\text{vertex}}{V}, \overset{\text{edges}}{\mathcal{E}})$

now we get the compliment (* hint from #9 in book)

$\bar{G} = (V, \bar{\mathcal{E}})$

↳ $\{(u,v) : u \in V, u \neq u, (v,v) \notin \mathcal{E}\}$

∴ the compliment of G ($\bar{G}$) has the same verteciers as G, but none of the same edges, all the opposite edges

if (there is a clique of $\overset{\text{minimum \# of verteciers}}{V'}$ in $\bar{G}$){
vertexCover = V − V'
}

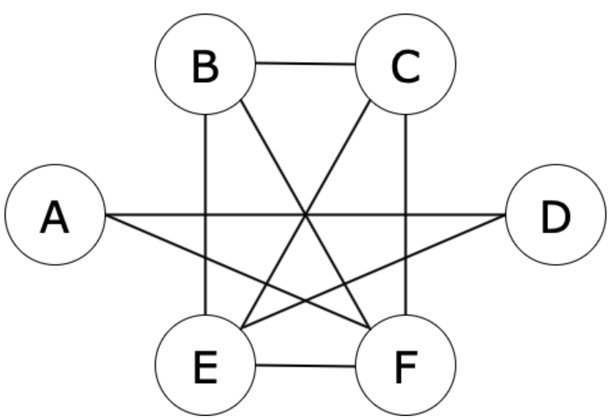
General Process

$G(V,\mathcal{E}) \rightarrow \boxed{K} \rightarrow G(V,\bar{\mathcal{E}}) \rightarrow \boxed{|V|-k} \rightarrow$ yes/No

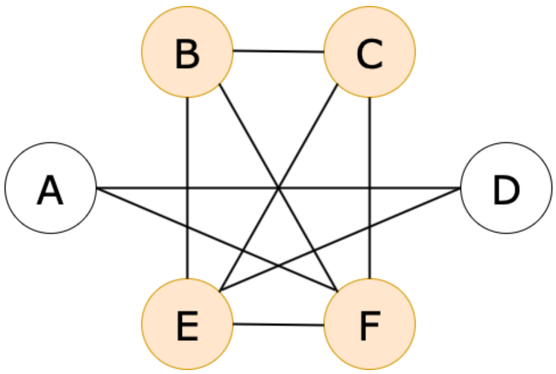
M = n nodes; Clique; Vertex Cover; polynomial time

(i)

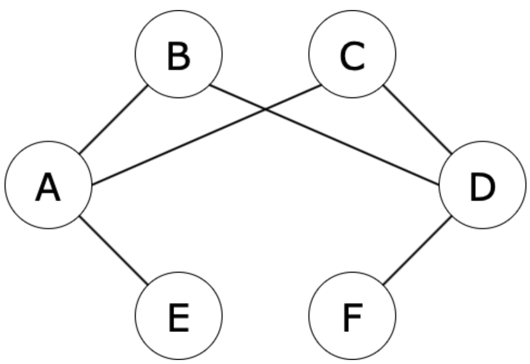
1. You have undirected graph G=(V,E) & int M



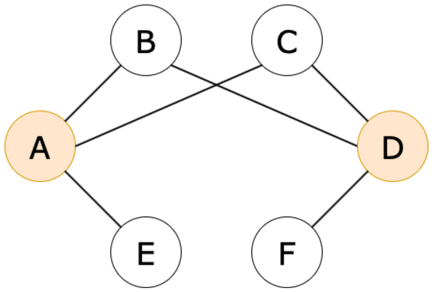
2. Get the Clique (S) of this graph with |S| = m
    a. S = {B,C,E,F}
    b. m = 4



3. Make S' = V-S (compliment)
    a. S' = {A,D}



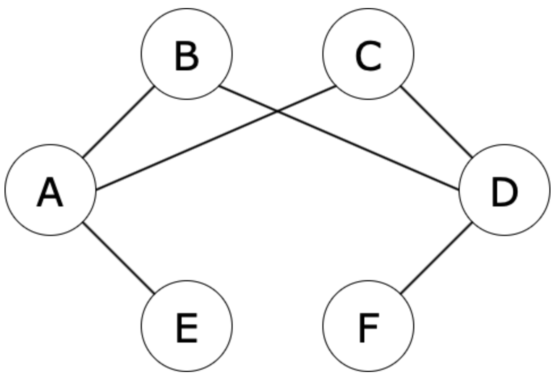
4. To show S' is a Vertex Cover= V-S={A,D}
    a. Consider any edge (c,d) ∈ E'
        i. Then (c,d) ∉ E
        ii. At least one of c or d is not in S (since S forms a clique)
        iii. At least one of c or d is in S'
        iv. therefore (c, d) is covered by S'



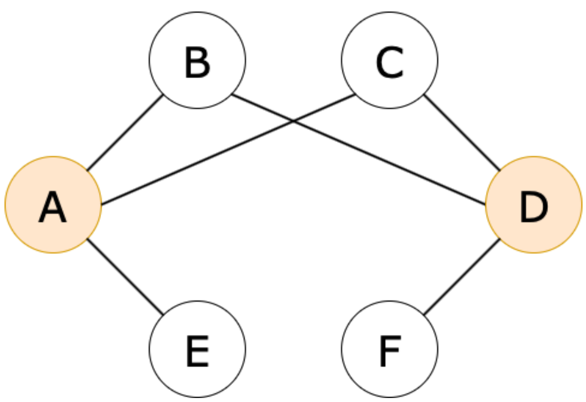
(ii)

Prove it backwards

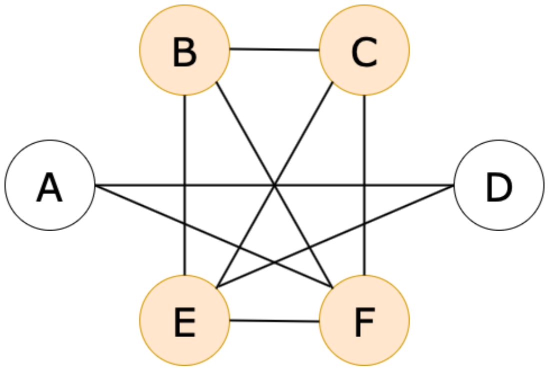
5. Make G'



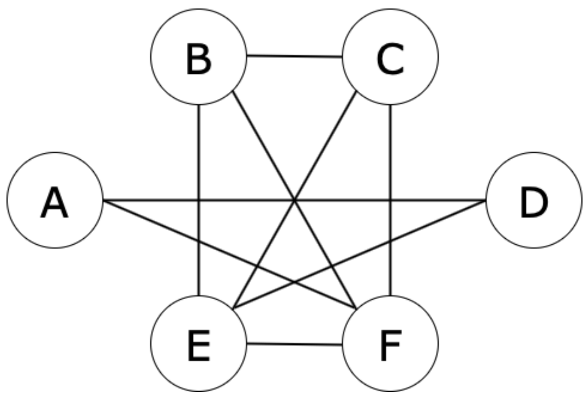
6. Get the Vertex Cover (S') of this graph
    a. |S'| = |V| - k



7. Consider S= V-S'
    a. Therefore |S| = k
8. Show that S is a clique



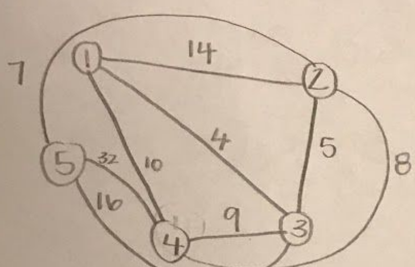
a. Consider any edge (c,d) ∈ E'
   i. if (c,d) ∈ E', then either c ∈ S' or d ∈ S' or both
   ii. By the contrapositive rule, if c ∉ S' and d ∉ S', then they are both in E proving S is a Clique in G



12.3

#1 Apply the nearest neighbor alg

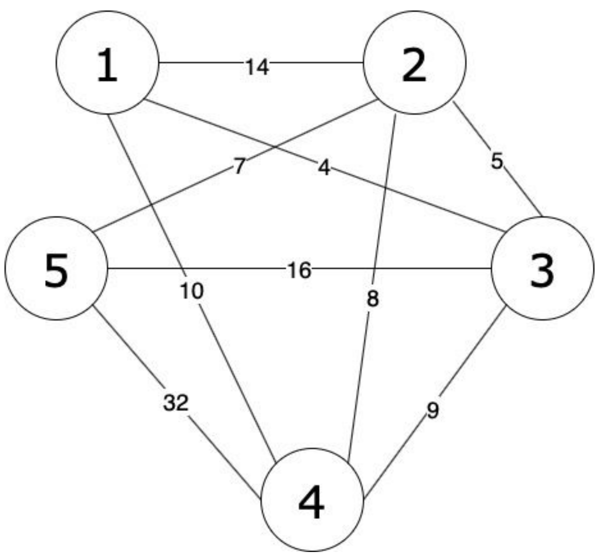
	1	2	3	4	5
1	0	14	4	10	∞
2	14	0	5	8	7
3	4	5	0	9	16
4	10	8	9	0	32
5	∞	7	16	32	0

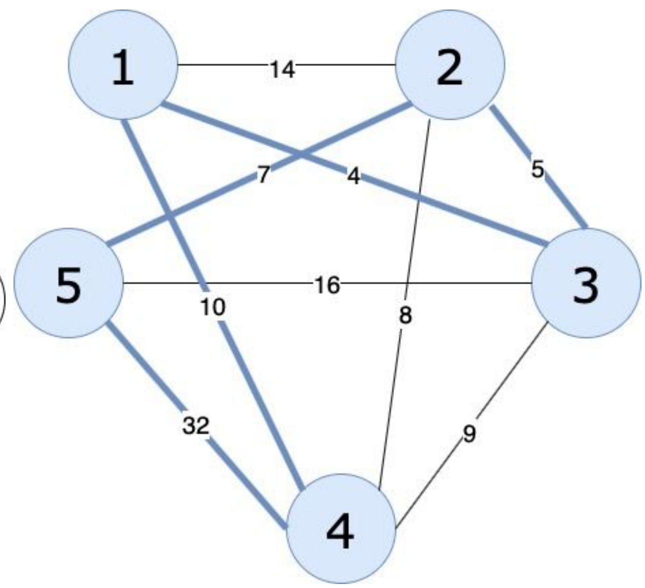


1. choose the edge with the smallest cost & use it as the 1st edge
2. chose from the edges that are connected from the current vertex to vertices that have NOT been visited
3. when you have visited ALL vertex's return to the start

$1 \to 3 \to 2 \to 5 \to 4 \to 1 = 58$

which is the same as





b compute the accuracy ratio of this solution

#1 find the length of the optimal tour

1 > 2 > 3 > 5 > 4 > 1] 77
1 > 2 > 4 > 5 > 3 > 1] 74
1 > 2 > 5 > 3 > 4 > 1] 56
1 > 2 > 5 > 4 > 3 > 1] 66
* 1 > 4 > 2 > 5 > 3 > 1] 45
1 > 4 > 5 > 2 > 3 > 1] 58

$$\text{accuracy ratio} = \frac{S_a}{S_*} = \frac{58}{45} = 1.29$$

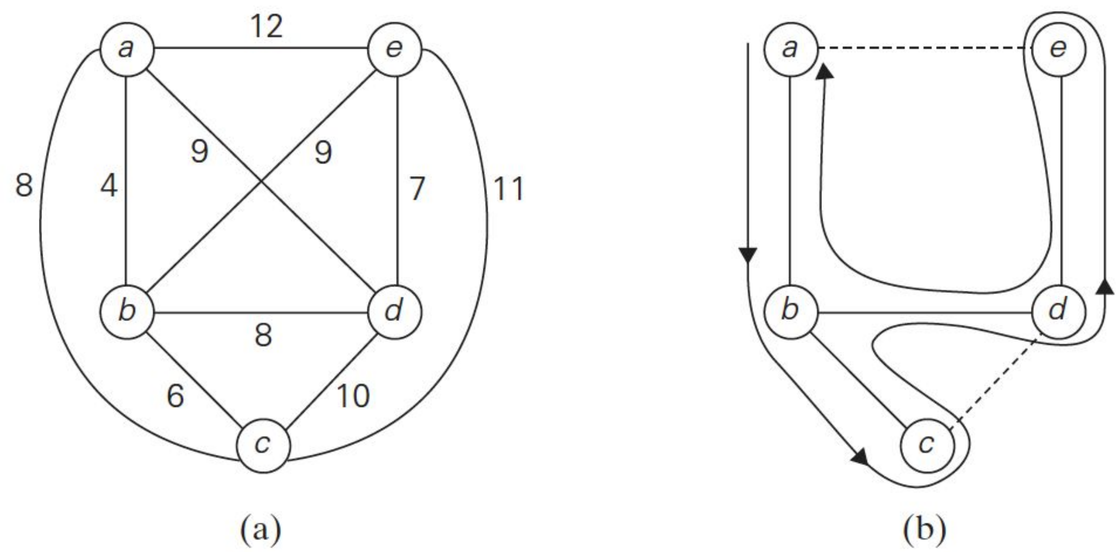
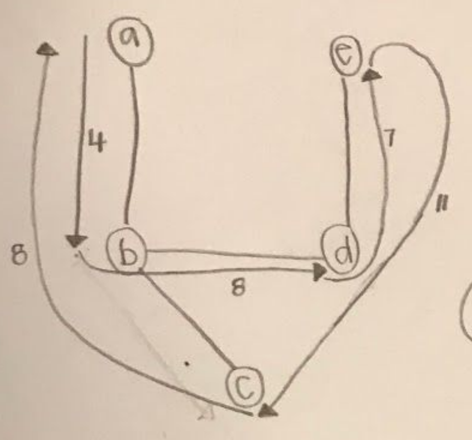


FIGURE 12.11 Illustration of the twice-around-the-tree algorithm. (a) Graph. (b) Walk around the minimum spanning tree with the shortcuts.

12.3

#3 apply twice-around-the-tree alg



① create MST

② start at a
└ DFS

③ delete all repeated v

38

ex b in book =

4 + 6 + 10 + 7 + 12 = 39

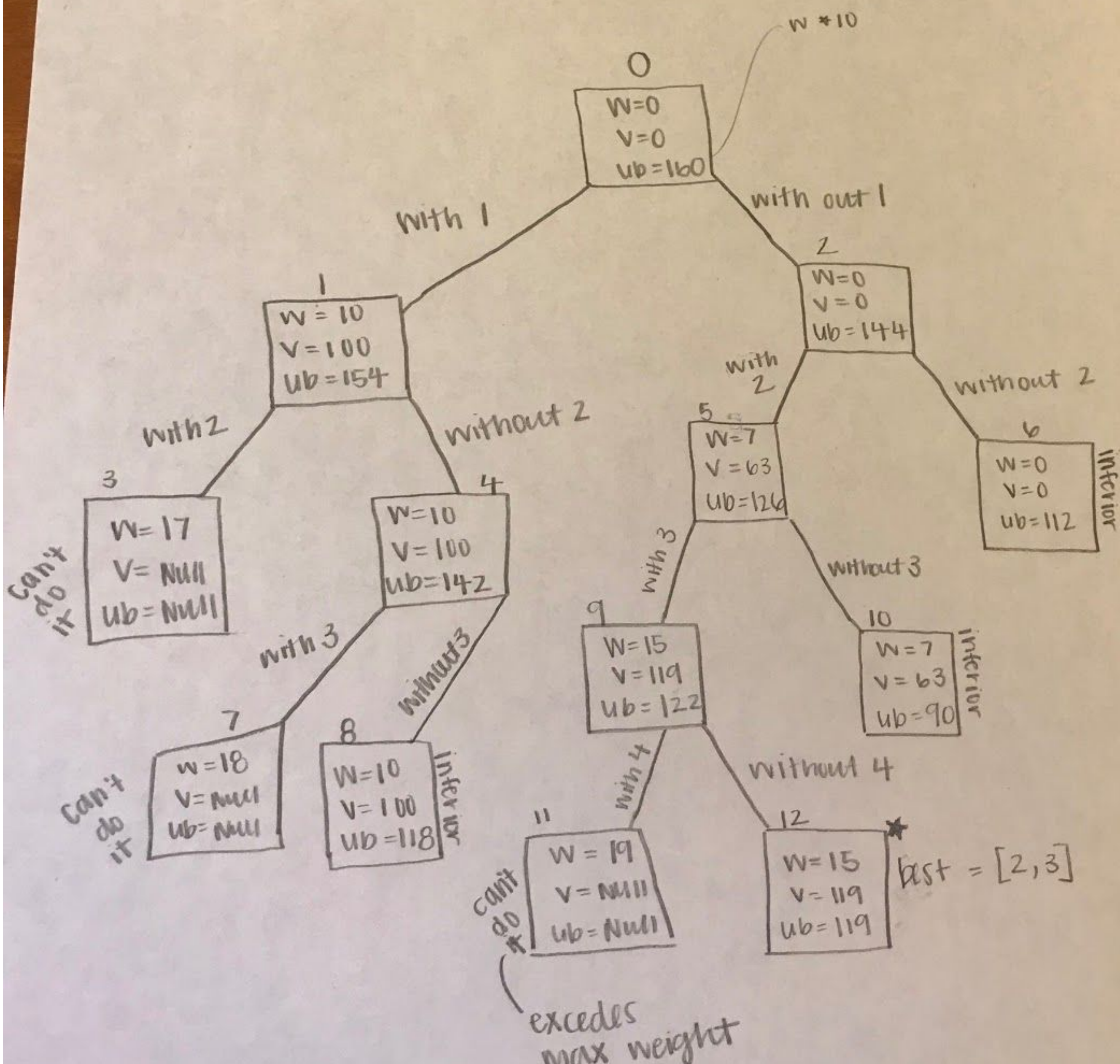
different values

12.2

#5 Solve the following instance of the knapsack problem by branch-and-bound

item	weight	value
1	10	$100
2	7	$63
3	8	$56
4	4	$12

W = 16 ← max weight



12.2

#6

a. Suggest a more sophisticated bounding function for solving the knapsack problem

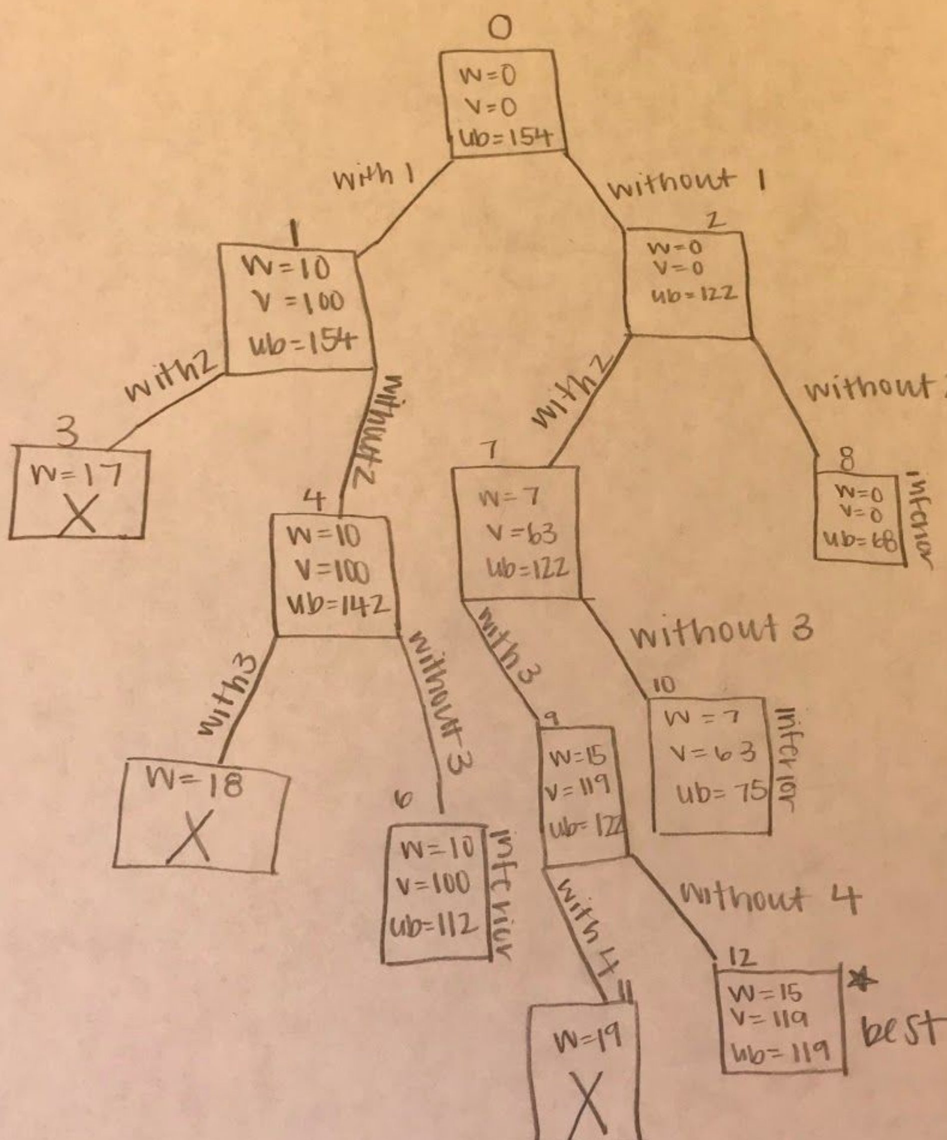
ub(s) = value of any subset that can be obtained by adding items to S

upper bound = add all the values until max weight is hit

= value + (next w − prev item / next w) next value

ex = 100 + (6 / 7) 63 = 154

b.



12.3

#7 First Fit Alg

place each of the items in the order given into the 1st bin the item fits in

(if there are no bins > make new bin & add 2 end of bin list

(a) apply

Max amt = largest S_x = 0.7

$S_1 = 0.4$

$S_2 = 0.7$

$S_3 = 0.2$

$S_4 = 0.1$

$S_5 = 0.5$

Bin 1	Bin 2	Bin 3
S_1, S_3, S_4	S_2	S_5

NOT OPTIMAL bc less bins the better

Bin 1	Bin 2
S_1, S_5	S_2, S_3, S_4

(b) what is the worst case? n^2

checking i-1 bins if all items are greater than the origional bin size

(c) Prove that FF is a 2-approximation alg

part 1

for any i there will be more than 1 bin by the nature of the alg

part 2

# of bins by approximate alg $< 2\sum_{i=1}^{n} s_i$

part 3

$B_{FF} < 2\sum_{i=1}^{n} s_i \leq 2B^*$

$B_{FF} = B^* < 2B^*$

$\sum_{k=1}^{B_{FF}} S_k = \sum_{k=1,\, k\neq \bar{k}}^{B_{FF}} S_k + S_{k=\bar{k}}$

size of items that must excede 1

index of bin with smallest sum of the item size

index of any bin in solution

part 4

Best Solution

$B_{FF} \leq \lceil 1.7B^* \rceil$