CS 325 -002 Quiz 2

Lyell Read

TOTAL POINTS

52 / 50

QUESTION 1

1 Q1 **2 / 2**

✓ **- 0 pts** Correct

QUESTION 2

2 Q2 **2 / 2**

✓ **- 0 pts** Correct

QUESTION 3

3 Q3 **2 / 2**

✓ **- 0 pts** Correct

QUESTION 4

4 Q4 **2 / 2**

✓ **- 0 pts** Correct

QUESTION 5

5 Q5 **0 / 2**

✓ **- 2 pts** Click here to replace this description.

QUESTION 6

6 Q6 **2 / 2**

✓ **- 0 pts** Correct

QUESTION 7

7 Q7 **2 / 2**

✓ **- 0 pts** Correct

QUESTION 8

8 Q8 **2 / 2**

✓ **- 0 pts** Correct

QUESTION 9

9 Q9 **2 / 2**

✓ **- 0 pts** Correct

QUESTION 10

10 Q10 **2 / 2**

✓ **- 0 pts** Correct

QUESTION 11

11 Q11 **2 / 2**

✓ **- 0 pts** Correct

QUESTION 12

12 Huffman Codes **10 / 10**

✓ **- 0 pts** Correct

QUESTION 13

Q13 8 pts

13.1 a) **4 / 4**

✓ **- 0 pts** No and counter example

13.2 b) **3 / 4**

✓ **- 1 pts** error in counter example

QUESTION 14

Fib 10 pts

14.1 a) Recursive **4 / 4**

✓ **- 0 pts** Correct

14.2 b) DP **4 / 4**

✓ **- 0 pts** Correct

14.3 c) Running time **2 / 2**

✓ **- 0 pts** Correct

QUESTION 15

15 Extra Credit **5 / 0**

+ **0 pts** Incorrect

+ **5 pts** Correct

✓ + **2 pts** **not an efficient algorithm**

✓ + **3 pts** **correct part a)**

+ **2 pts** correct part b)

+ **2 pts** part a) minor error

+ **1 pts** incomplete

CS 325 – 002 Winter 2020

Quiz 2

You may use a 3"x5" notecard and calculator. You have 50 minutes to complete the exam. Do not leave early, remain seated. When time is called pass your exam to the aisle. Failure to submit your exam in a timely manner will result in a zero.

Print your name: LYELL READ

Student ID: 933-609-535

Multiple choice & T/F. Fill in the circle that corresponds to the best answer. 2 pts each.

1. Let T be a complete binary tree with n vertices. Finding a shortest path (measured by number of edges) from the root of T to a given vertex v ∈T takes

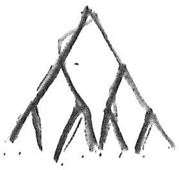
→ Sorted? no.

● O(n)

~~○ O(lgn)~~

~~○ Ω(n^2)~~

○ Θ(nlgn)



18 here 15 only

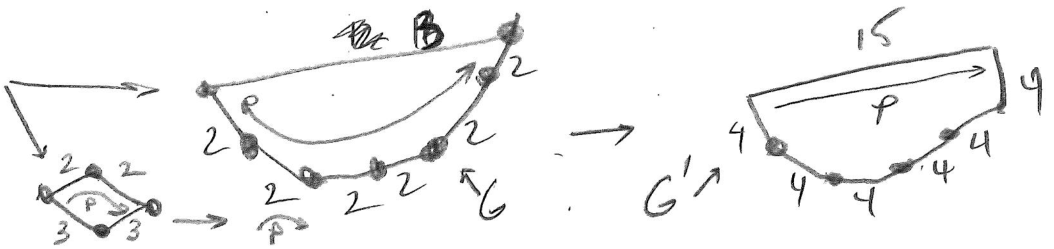
s → t

2. Given a weighted directed graph G = (V,E,w) and a shortest path P from s to t, if we add 2 to the weight of every edge in G to produce G'=(V,E,w'), is P also a shortest path from s to t in G'?

○ Never

● Sometimes

● Always



3. Given a weighted graph G = (V,E,w) and a minimum spanning tree T, if we add 2 to the weight of every edge in G to produce G'=(V,E,w'), is T also a minimum spanning tree in G'?

○ Never

○ Sometimes

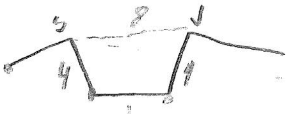
● Always

Kruskals will behave the same, as all edges will sort the same.

4. Given a weighted graph G, let T be a minimum spanning tree of G. Then, for any pair of vertices s and v, the shortest path from s to v in G is **always** the path from s to ~~t~~ V in T. *Select one*

○ True

● False



5. If a dynamic programming problem satisfies the optimal-substructure property, then a locally optimal solution is globally optimal.

● True

○ False

Name: LYELL READ	OSU ID: 933-609-535

6. The running time of a dynamic programming algorithm is **always** Θ(P) where P is the number of subproblems.

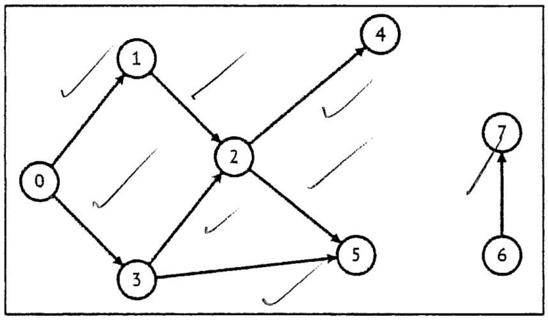
○ True

● False

what if s.P. takes P time as well?

p^2

7.



Which of the following is a topological sort of the graph above.

○ ~~7, 6, 5, 2, 4, 3, 1, 0~~

● 6, 7, 0, 1, 3, 2, 5, 4 ✓

○ 0, 1, 2, 3, 4, 5, 6, 7

○ ~~0, 3, 2, 5, 1, 4, 6, 7~~

○ None of the above

6 >> 7

8. In an undirected weighted graph with distinct edge weights, the lightest edge is in the MST.

● True

☒ False

kruksals — Guarentees this

9. All dynamic programming problems can be solved by using a greedy choice algorithm.

○ True

● False

10. Which of the following is an example where the greedy method does not achieve the optimal solution to the Coin Change problem. D is the set of denominations and A is the amount to make change.

○ D = { 1, 5, 10, 25} and A = 30

● D = { 1, 7, 11} and A = 14

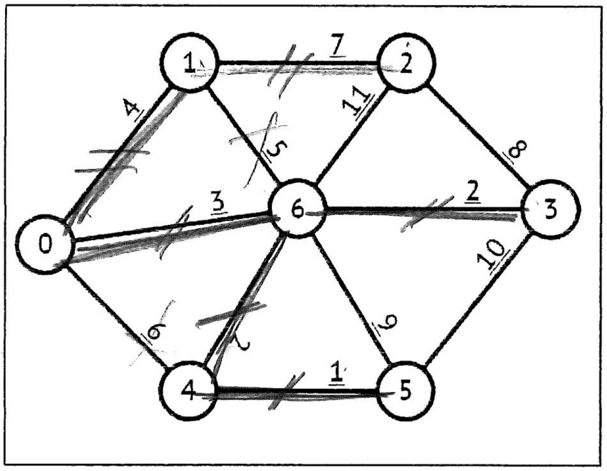
○ D = { 1, 7, 11} and A = 22

○ D = { 1, 3, 8, 13, 25} and A = 26

○ D = { 1, 5, 10, 25, 50 } and A = 52

○ None of the above

11. (2 pts)



Kruksals:
lowest wt
Check NO ↻
ADD

In the above graph what is the weight of the MST T? 19

95

1 2 2 3 4 7

4

8

10

1

19 chk

1 + 2 + 2 + 3 + 4 + 7

5 + 3 + 4 + 7

8 + 7 + 4 = 19

Name: LYELL READ

OSU ID: 933-609-535

12. *(10 pts)* Suppose we have an alphabet with only five letters A, C, E, J , S which occur with the following frequencies:

- A = 50, C = 12, E = 75, J = 15, S = 40

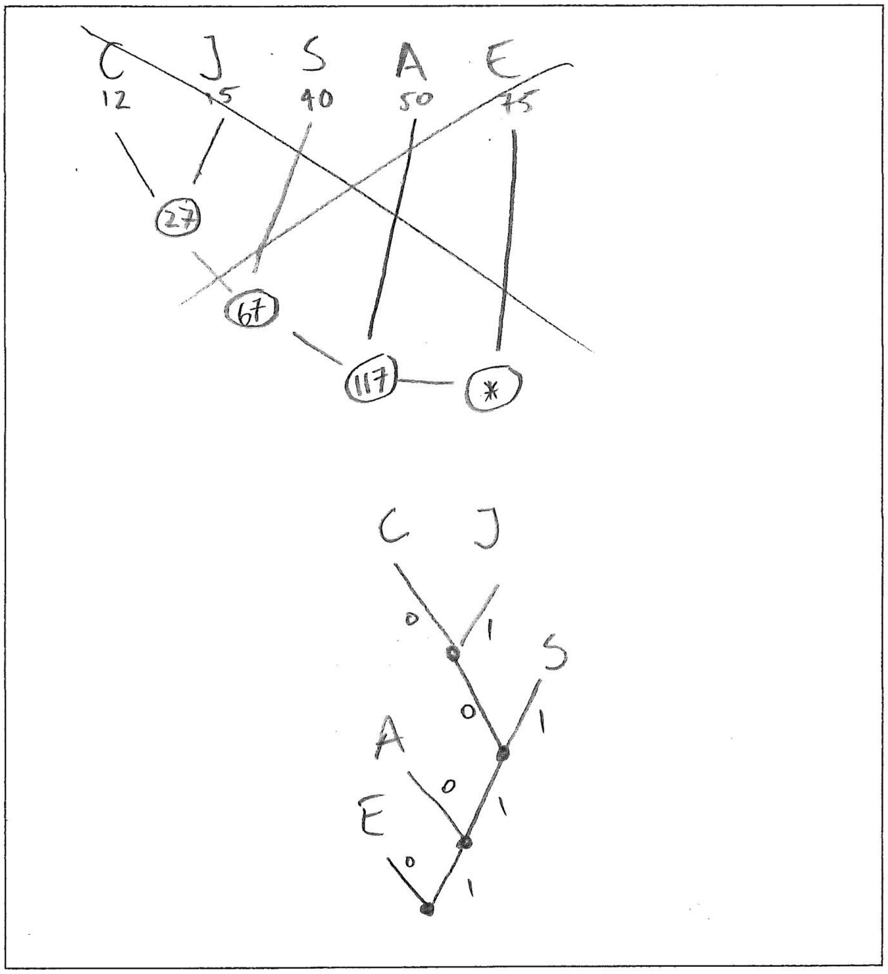
Construct a Huffman code using the following guidelines while constructing the code

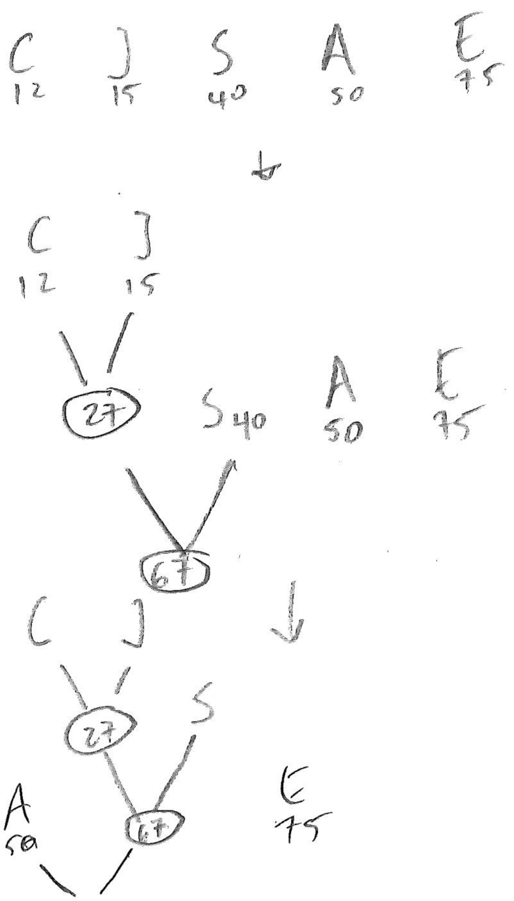
- the lowest frequency node is the left child in the tree while the higher frequency node is the right child
- when creating the code the left branch is assigned a 0 while the right branch is assigned a 1.
- ties are broken by selecting the node with the minimum height.

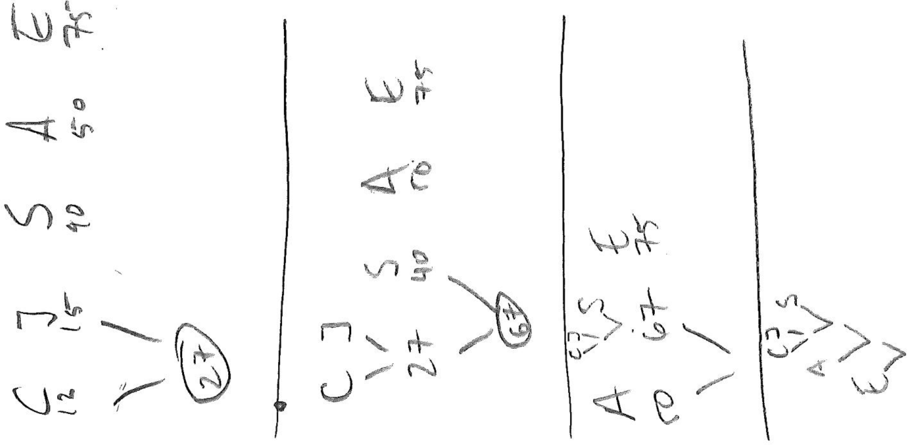
The Huffman binary coding are:

A =	10
C =	1100
E =	0
J =	1101
S =	111

Huffman Tree



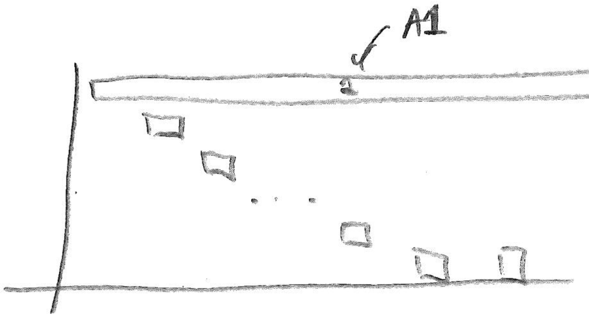




13. *(8 pts)* Recall the activity-selection problem from HW 4 in which you are given activities with start and finishing times and the goal is to maximize the number of compatible activities completed. You proved that the last-to-start greedy choice selection criteria yielded an optimal solution. Are either of the below greedy choice selection methods optimal? If “yes”, give an informal proof. If "no" give a counter example showing that the criteria are not optimal.

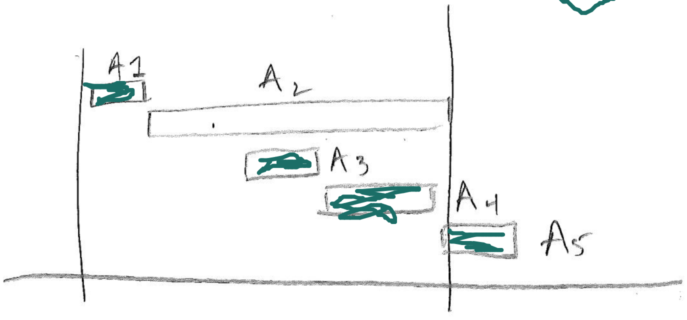
a) **Method 1**: Always select the activity that has earliest starting time.

NOT OPTIMAL



this algorithm would choose A1, whereas there are many other activities that could be completed in the same time

b) **Method 2**: Always select the activity that takes the least amount of time.



NOT OPTIMAL this would choose A2, once A1 is complete, which forgoes completion of A3, A4. (it is unclear if this algorthm takes the lowest every time it finishes a task or if it waits to find a Task with a length value below a threshhold).

Name: WELL READ

OSU ID: 933-609-535

14. *(10 pts)* The terms in the Fibonacci sequence are given by:

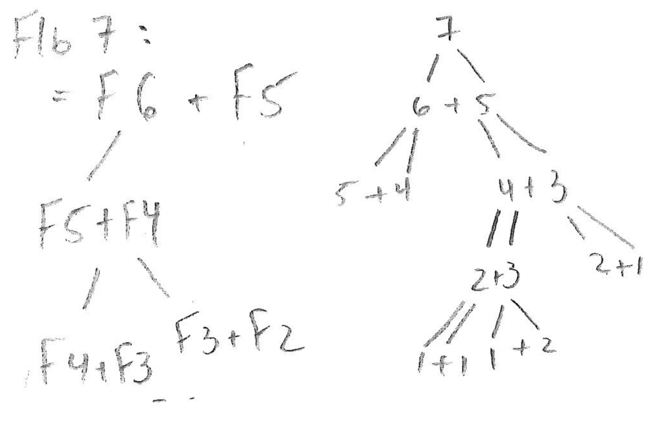
$F_1 = 1,\ F_2 = 1;\quad F_n = F_{n-1} + F_{n-2}$

1 2 3 4 5 6 7

1 1 2 3 5 8 13 21

a) Give the pseudocode for a recursive algorithm to calculate the nth term in the Fibonacci sequence.

```
Fib (n) :=
    if n = 0:
        ret 0
    if n = 1:
        ret 1
    if n = 2:
        ret 1
    else: return fib(n-1) + fib(n-2)
```



b) Give the pseudocode for a dynamic programming algorithm to compute the nth term in the Fibonacci sequence. Is your code bottom-up [✓ or memorized DP?]

"memoized" :)

~~bottom up~~

```
O[1]=1, O[2]=1
O[3..sizeof(O)]=INT_MIN
Fib (n, O):
    ~~for~~ if O[n] =! int_min:
        return O[n]
    else:
        tmp= fib(n-1,O) + fib(n-2,O)
        O[n]=tmp
        return tmp
```

c) Compare the running time of the recursive algorithm to the running time of the DP algorithm.

recursive algo~~t~~rithm will run in 2^n time, while DP will run in less than that. It will run about $O(n^2)$ or so time ~~[illegible]~~ (the question asks for comparison, so DP < Rec). Bottom up would run in $O(n)$ time.

EXTRA CREDIT. (5 pts)

Consider a variation of the 0-1 Knapsack problem where all items have the same value of v. In other words, we have n items where w_i is the weight of the i^{th} item, every item has a value of v, and the capacity of the knapsack is W. We want to find a subset of items whose total weight is at most W and whose total value is as large as possible. (This is knapsack without repetition; each item can be chosen at most once).

a) Verbally describe and give pseudocode for an **efficient** algorithm for this problem.

As all values are the same, i.e for an item of weight W_{small}, we get the same return for item of weight W_{large}, this is just a metter of getting to exacty the value of W in the bag, using the smallest items successively:

- Sort items from small to large weights
- Pop from start of list until the next pop will exceed bag weight

P.C.: NOTE: Best implemented as a stack, where pop from top is efficie

```
sort w;
while ( + w[0] + corrent weight) < total weight
    V += Value of one item
    W+ = w[0]; remove w[0]
```

b) What is the asymptotic running time of your algorithm? Explain.

this algorthum will be composed of a sort (Quick sort used, at n log n time) plus popping x<n items therefore, this will take nlogn time, as the popping can be ignored.