csce350 — Data Structures and Algorithms
Fall 2020 — Final Exam

Evan

first/given name

Owre

last/family name

approximately 7:30 PM

time downloaded

- ✓ Read the questions carefully and make sure to give the answers asked for. Pay particular attention to **boldface** words. Don't give a beautiful answer to the wrong question.
- ✓ You have 180 minutes to complete this test.
- ✓ A single 8.5″ × 11″ sheet of notes in your own handwriting is required.
- ✓ No calculators nor other reference material are allowed.
- ✓ Show enough work to convince your instructor that you know what you're doing. Mark your answers clearly.
- ✓ Make sure you have all 23 pages, including the three reference sheets at the end.
- ✓ Partial credit will be awarded for incorrect answers that demonstrate partial understanding of the relevant concepts. Therefore, it is to your advantage to explain your reasoning and show your work. However, meaningless or irrelevant writing will not earn partial credit.
- ✓ Review the submission checklist at the end before uploading.

I understand that it is the responsibility of every member of the Carolina community to uphold and maintain the University of South Carolina's Honor Code. I certify that I have neither given nor received unauthorized aid on this exam.

signature

Problem	Value	Your Score
1	66	
2	10	
3	12	
4	10	
5	10	
6	12	
7	12	
8	12	
9	12	
10	14	
11	10	
12	10	
13	10	
Total	200	

Problem 1 *(66 points)*

Multiple choice. **Choose** the best answer for each question. (2 points each)

A B C D

1. In a hash table, collisions can be resolved by ______.

   A. either chaining or probing
   B. neither chaining nor probing
   C. probing but not chaining
   D. chaining but not probing

A B C D

2. Here is a code for encoding a set of 5 characters:

Character	A	B	C	D	E
Code	00	001	1	110	111

This code is ______.

   A. variable length but not prefix-free
   B. fixed length but not prefix-free
   C. fixed length and prefix-free
   D. variable length and prefix-free

A ~~B~~ C D

3. This recurrence is used for the dynamic programming solution to the knapsack problem:

$$V[i,j] = \begin{cases} \max(V[i-1,j], V[i-1,j-w_i]+v_i) & \text{if } j \geq w_i \\ \underline{V[i-1,j]} & \text{if } j < w_i \end{cases}$$

The underlined portion corresponds to the case in which ______.

   A. we take item j
   B. we have room for item j, but choose not to take it
   C. we take item i
   D. we do not have room for item i

A B C D

4. The code generated by a Huffman tree is ______.

   A. prefix-free only if the tree has height at most $\lceil \log n \rceil$
   B. never prefix-free
   C. always prefix-free
   D. prefix-free only if the probabilities are all equal

A B C D

5. Benny has an instance of the knapsack problem with 5 items. When he uses the dynamic programming algorithm from this class to solve it, he is surprised to see that *all* of the cells in the DP table have the value 0. What happened to Benny?

A. There is more than enough capacity to take all of the items, so the unused capacity is 0.
B. The items all have the same weight, so DP cannot choose between them.
C. The items all have the same value, so DP cannot choose between them.
D. Each item in Benny's instance has a weight greater than the capacity.

A B C D

6. Which of the following statements is true?

A. For a given problem, the run times of all correct algorithms have the same order of growth.
B. For each problem, there exists a single correct algorithm.
C. In a stack, elements are inserted at one end and removed from the other end.
D. A graph is a collection of nodes along with a collection of edges connecting those nodes.

A B C D

7. After the partition step of quicksort, the ______ element is guaranteed to be in its correct final position.

A. median
B. largest
C. pivot
D. smallest

A B C D

8. ______ is an algorithm design strategy that uses a table to store solutions to overlapping subproblems, and fills in this table using nested loops.

A. top-down dynamic programming
B. divide and conquer
C. bottom-up dynamic programming
D. transform and conquer

A B C D

9. The worst-case run time of quickselect is ______.

A. $\Theta(\log n)$
B. $\Theta(n^2)$
C. $\Theta(n \log n)$
D. $\Theta(n)$

A B C D 10. The worst-case run time of quicksort is ______.

A. $\Theta(n \log n)$
B. $\Theta(n)$
C. $\Theta(\log n)$
D. $\Theta(n^2)$

A B C D 11. Why is rebalancing important for binary search trees?

A. Without rebalancing, the INSERT and SEARCH can take $\Theta(n)$ time.
B. Without rebalancing, the INSERT operation will not be able to create new nodes.
C. Without rebalancing, the INSERT and SEARCH can take $\Theta(n \log n)$ time.
D. Without rebalancing, the SEARCH operation returns incorrect answers.

A B C D 12. The QuickSelect algorithm is used for ______.

A. Selecting the kth smallest element in an array.
B. Selecting correct pivot to use within QuickSort.
C. Selecting the next element within selection sort.
D. Selecting which elements to merge within MergeSort.

A B C D 13. Which algorithm design strategy best describes the algorithm below?

```
MULTIPLYINTEGERS(a, b)
  p ← 0
  for i ← 1, ..., b do
    p ← p + a
  end for
  return p
```

A. brute force
B. decrease-and-conquer
C. divide-and-conquer
D. transform-and-conquer

A B C D 14. Which of the following features is *not* part of the definition of the word "algorithm"?

A. can be analyzed by writing a summation
B. complete in a finite amount of time
C. unambiguous instructions
D. correct answers for every input

A B C D

15. What are the inputs and outputs for Euclid's algorithm?

A. The inputs are two non-negative integers m, n; the output is the greatest common divisor of m and n.

B. The inputs are two integers a, n; the output is a^n.

C. The inputs are a set of three points x, y, and z; the output is a circle passing through those three points.

D. The inputs are two non-negative integers m, n; the output is the least common multiple of m and n.

A B C D

16. Barry is executing Kruskal's algorithm, using a disjoint set forest data structure to keep track of a partition of five nodes named a, b, c, d, and e. He has this parent array (in which the nodes are indexed in alphabetical order):

a	b	c	d	e
c	d	e	d	e

Then Barry performs the operation UNION(a, b). Because of this operation, node ______ gets a new parent.

A. b

B. a

C. c

D. e

A B C D

*17. A certain algorithm executes its basic operation $10n$ times in the best case and $30n$ times in the worst case. How many times does the algorithm execute its basic operation in the average case?

A. There is not enough information to determine the answer.

B. $20n$

C. $10n$

D. $30n$

A B C D

18. Which algorithm(s) for computing minimum spanning trees is/are greedy?

A. both Prim's and Kruskals

B. Prim's but not Kruskals

C. Kruskal's but not Prim's

D. neither Prim's nor Kruskals

A B C D

19. This recurrence is used for the dynamic programming solution to the knapsack problem:

$$V[i,j] = \begin{cases} \max(V[i-1,j], \underline{V[i-1,j-w_i]+v_i}) & \text{if } j \geq w_i \\ V[i-1,j] & \text{if } j < w_i \end{cases}$$

The underlined portion corresponds to the case in which ______.

A. we do not have room for item j

B. we take item i

C. we have room for item i, but choose not to take it

D. we take item j

A B C D E

20. In MERGESORT, what is the role of the MERGE function?

~~A~~. MERGESORT does not use the MERGE function.

~~B~~. It merges a single element into the part of the array that is already sorted.

~~C~~. It determines whether MERGESORT should recurse on the left side, the right side, or both.

D. It combines the two recursively-sorted subarrays into a single, fully-sorted array.

E. It arranges the elements around a pivot element.

A B C D

21. In a hash table, collisions occur when ______.

A. Two keys have different hash values.

B. Two keys share the same hash value.

C. At least one key causes a divide-by-zero error when evaluating the hash function.

D. Every time we insert into the table.

A B C D

22. Here are the last two rows of a complete dynamic programming table for the Knapsack problem with $W = 4$, $v_5 = 10$, and $w_5 = 2$.

	$j=0$	$j=1$	$j=2$	$j=3$	$j=4$
			...		
$i=4$	0	14	15	17	24
$i=5$	0	14	15	24	25

$25 - 10 = 15$

$4 - 2 = 2$

Does the final solution include item 5?

A. There is not enough information to tell.

B. No.

C. Maybe. There are two correct solutions, one with item 5 and one without it.

D. Yes.

A B C D

23. What is the primary difference between decrease-and-conquer and divide-and-conquer?

A. Decrease-and-conquer algorithms are generally inefficient, divide-and-conquer algorithms are generally very efficient.
B. Decrease-and-conquer algorithms are generally recursive, but divide-and-conquer algorithms are generally iterative.
C. Decrease-and-conquer algorithms generally solve only one smaller subproblem recursively, but divide-and-conquer algorithms generally solve two or more subproblems recursively.
D. There is no meaningful difference. These are two different names for exactly the same algorithm design strategy.

A B C D

24. Charlie is writing a program that stores an ordered collection of elements. Charlie's program frequently needs to access the elements directly via their numerical positions, but only rarely needs to insert or delete elements from the collection. What is the most appropriate data structure for Charlie to use?

A. linked list
B. queue
C. adjacency matrix
D. array

A B C D

25. One primary advantage of top-down dynamic programming over bottom-up dynamic programming is ______.

A. Only compute values that are relevant to the final answer.
B. Can optimize space by discarding entries that are no longer needed.
C. No overhead for 'Computed yet?' checks.
D. No overhead for function calls.

A B C D

26. The basic idea of dynamic programming is to express the overall solution using a ______ relating the overall solution to the solutions of subproblems.

A. linked list
B. binary search tree
C. recurrence
D. summation

A B C D

27. The worst-case run time of binary search is ______.

A. $\Theta(n \log n)$
B. $\Theta(n)$
C. $\Theta(n^2)$
D. $\Theta(\log n)$

A B C D

28. Which of the following is *not* one of the requirements for the individual decisions made in a greedy algorithm?

A. feasible
B. irrevocable
C. locally optimal
D. constant time

A B C D

29. Barry is using Kruskal's algorithm to compute the minimum spanning tree of a connected graph with five nodes named a, b, c, d, and e. He has this parent array (in which the nodes are indexed in alphabetical order):

a	b	c	d	e
c	d	e	d	e

Has Kruskal's algorithm finished its work?

A. Yes, because the first node, a, is not its own root.
B. There is not enough information to tell.
C. No, because the nodes do not all have the same root.
D. No, because the entries in the parent array are not all the same.

A B C D

30. The algorithm design technique based on a straightforward approach to solving a problem, usually directly based on the problem statement and definitions of the concepts involved, is called _______.

A. decrease-and-conquer
B. divide-and-conquer
C. brute force
D. dynamic programming

A B C D

31. What is the primary difference between divide-and-conquer and dynamic programming?

A. Divide-and-conquer algorithms are generally efficient; dynamic programming algorithms algorithms are generally very inefficient.
B. Divide-and-conquer algorithms can be analyzed using big-Θ notation; dynamic programming algorithms cannot.
C. In divide-and-conquer algorithms, the problem is divided into two or more independent subproblems; in dynamic programming algorithms, the subproblems can overlap.
D. There is no meaningful difference. These are two different names for exactly the same algorithm design strategy.

A B C D

32. This recurrence is used for the dynamic programming solution to the knapsack problem:

$$V[i,j] = \begin{cases} \max(\underline{V[i-1,j]}, V[i-1, j-w_i] + v_i) & \text{if } j \geq w_i \\ V[i-1,j] & \text{if } j < w_i \end{cases}$$

The underlined portion corresponds to the case in which ______.

A. we take item i

B. we take item j

C. we do not have room for item j

D. we have room for item i, but choose not to take it

A B C D

33. Where are keys stored in an AVL tree?

A. only at the leaves

B. only at the root node

C. only at the internal nodes

D. at all of the nodes

Problem 2 *(10 total points)*

[a] **Explain**, in **exactly one English sentence**, why we usually do not bother with **best-case** analysis. *(5 points)*

very rarely occurs & is not useful for analyzing how an algorithm will preform in real senaria

[b] **Explain**, in **exactly one English sentence**, we usually do not bother with **average-case** analysis. *(5 points)*

explains how the program will usually run but unhelpful for preparing an algorithm, ~~better to check how poorly our program can run~~ often requires addition input

Problem 3 *(12 total points)*

[a] In the algorithm below, the basic operation is . **Write** an expression, including one or more summations, for the exact count of basic operations executed by this algorithm, as a function of n. (Just write the summation. You do **not** need to simplify nor solve.) *(6 points)*

```
RADICAL(A[0,...,n-1])
  r ← 0
  for i ← 0,...,n-1 do
    for j ← i+1,...,n-1 do
      r ← r  A[i]
      r ← r  A[j]
    end for
  end for
  return r
```

$$T(n) = \sum_{i=0}^{n-1} \sum_{j=i+1}^{n-1} 2$$

[b] In the algorithm below, the basic operation is . **Write** a recurrence and base case for the **exact count** of basic operations executed by this algorithm, as a function of n. (Just write the recurrence. You do **not** need to simplify nor solve.) *(6 points)*

```
COWABUNGA(n)
  if n < 1 then
    return n
  else
    return COWABUNGA(n-1)  COWABUNGA(n-2)
  end if
```

CowaBunga(0) = 0

CowaBunga(n) = 1 + CowaBunga(n-1) + CowaBunga(n-2)

Problem 4 *(10 points)*

Consider a hypothetical algorithm that executes its basic operation

$$t(n) = 10n^2 + \log n + 1000$$ ← is a pollynomial we are only concerd w/the leading argument

times for every input of size n. **Prove, using limits**, that $t(n) \in \Theta(n^2)$.

$\cancel{L = \frac{10n^2}{n^2}}$ $L = \lim \frac{10n^2}{n^2}$

$= 10 \lim \frac{n^2}{n^2} = 10 = L$

$0 < L < 10$

$\therefore \in \Theta(n^2)$

Problem 5 *(10 points)*
Use the exhaustive search algorithm from the lecture to solve this instance of the Knapsack problem:

i	1	2	3
w_i	8	12	18
v_i	10	8	16

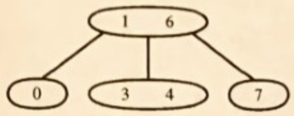
$W = 20$

Show your work.

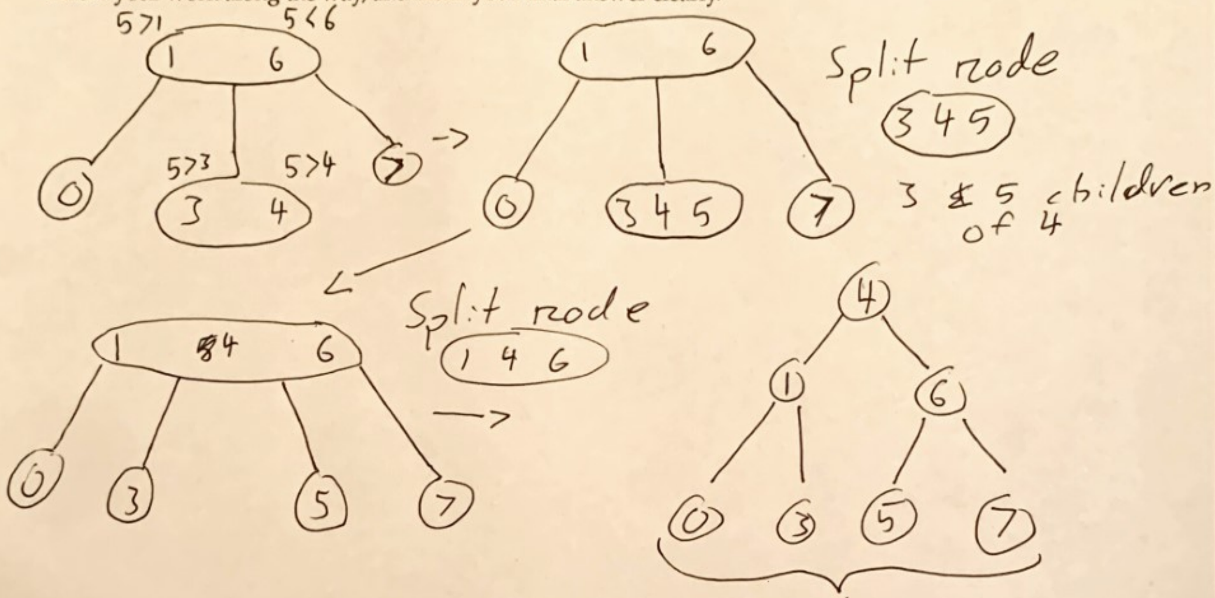
take	weight	value	
1 2 3	38	34	X
1 2	20	18	→ largest value within W
1 3	26	26	X
1	8	10	
2 3	30	24	X
2	12	8	
3	18	16	
None	0	0	

Problem 6 *(12 points)*

Draw the 2-3 tree that results from inserting a 5 the 2-3 tree below.



Show your work along the way, and mark your final answer clearly.



final 2-3 tree
after insert 5
all 2-Nodes w/ 2 children

Problem 7 *(12 total points)*

Slash starts wants to use QuickSelect to find the third smallest (that is, $k = 2$) element in his array.on the following array:

$A =$

71	39	62	12	79	76	22	10	89	60

[a] First, Slash chooses the first element, 71, as the pivot and partitions the array. **Show** an ordering of the array that is correctly partitioned on this pivot.[1] *(6 points)*

Wrong Answer

$A =$

0	1	2	3	4	5	6	7	8	9
~~39~~	~~62~~	~~12~~	~~22~~	~~10~~	~~60~~	~~71~~	~~79~~	~~76~~	~~89~~

↑ looking for element that stays in this index

[b] Next, Slash proceeds to the recursive part of QuickSelect. **How many** *top-level* recursive calls does QUICKSELECT make in this case? (Hint: The answer is 0, 1, or 2.) **How many elements** are in the subarray passed to each top-level recursive call? *(6 points)*

$k=2$

2 < 6 QuickSelect(A[0...5])   39 62 12 22 10 60

2 < 3 QuickSelect(A[0...2])   12 22 10 39 62 60

0 1 2 3

A[2] ← s = k

2 > 1 ~~QuickSelect(A[2])~~   12 22 10

10 12 22

Ignore improperly partitioned

↳ 22 third smallest element of A

2 recursive calls made @ top level

$A =$

60	39	62	12	22	10	71	76	89	79

10 ~~12 22 3~~9 60 62

39 12 22

[1] Hint: There are multiple correct answers. The specific ordering that Slash will get depends on how he implements PARTITION. Any correct partition is a correct answer for this problem.

Problem 8 *(12 total points)*

Recall MERGESORT, which works by dividing its input into two equal parts, recursively sorting each part, then merging the two sorted subarrays. What would happen if we split the array into three parts instead?

```
THREEWAYMERGESORT(A[0,...,n-1])
  if n > 1 then
    copy A[0,...,⌊n/3⌋-1] to a new array B
    copy A[⌊n/3⌋,...,⌊2n/3⌋-1] to a new array C
    copy A[⌊2n/3⌋,...,n-1] to a new array D
    THREEWAYMERGESORT(B)
    THREEWAYMERGESORT(C)
    THREEWAYMERGESORT(D)
    TRIPLEMERGE(B,C,D,A)
  end if
```

In this algorithm, TRIPLEMERGE is a generalization of the MERGE function used in normal MERGESORT. It takes three sorted subarrays and merges them back into a single sorted array. In the worst case, it makes $2n-3$ key comparisons. Throughout this question, you may assume that the input size is a power of 3.

[a] **Write** a recurrence for the worst-case run time of THREEWAYMERGESORT. *(5 points)*

$T(1) = 0$

$T(n) = 3T(n/3) + \Theta(n^1)$

[b] **Find** the order of growth for this recurrence, expressed using Θ notation. **Explain** your answer. *(5 points)*

$\underset{a}{}T(n) = \underset{a}{3}T(n/\underset{b}{3}) + \Theta(n^{\underset{d}{1}})$ using Master Theorem

$3 = 3^1 \therefore T(n) \in \Theta(n^d \log(n)) = \Theta(n \log n)$

3 recursive calls made each of size $n/3$
plus linear work to copy the arrays

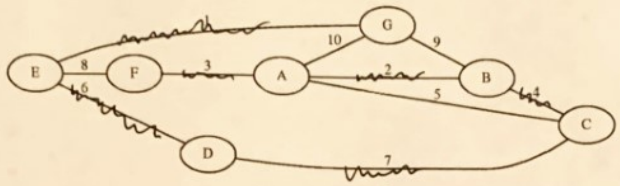
[c] Is this algorithm an improvement over the standard MERGESORT? Why or why not? *(2 points)*

No this is not an improvement because
our run time of $\Theta(n \log n)$ is equivalent to
standard mergesort's time efficiency of $\Theta(n \log n)$

Problem 9 *(12 total points)*

[a] Use *Kruskal's* algorithm to **compute** the Minimum Spanning Tree of the following weighted graph. **List**, in order, the edges considered by the algorithm and indicate whether or not they are included in the MST. **Mark** the edges of the completed MST on the graph. *(6 points)*

Heap of edges
✓ EF 1
✓ AB 2
✓ AF 3
✓ BC 4
× AC 5
✓ ED 6
✓ DC 7
× EF 8
~~AG 10~~
× BG 9
× AG 10

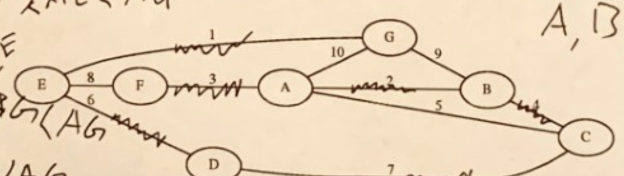


root

Node A B C D E F G
Parent F A B C G F E
D

[b] Use *Prim's* algorithm to **compute** the Minimum Spanning Tree of the following weighted graph, starting with node A. **List** the order in which the edges are selected by the algorithm. **Mark** the edges of the completed MST on the graph. *(6 points)*

Start A, AB<AF<AC<AG
Take AB



A, B, F, C, D, E

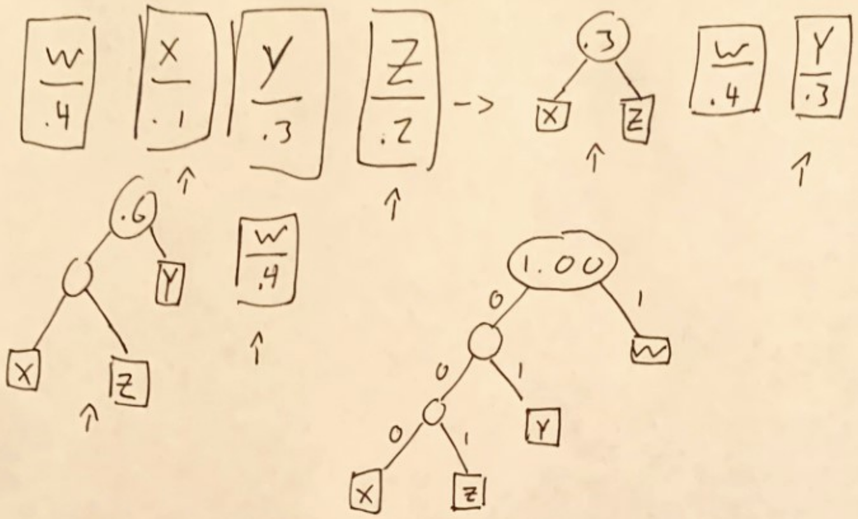
FE
AF<BC<AC<BG<AG
FE<
BC<AE<BG<AG
CD<FE<BG<AG
DE<FE<BG<AG
EG<BG<AG → No unseen nodes 6edges < 7verticies

Problem 10 *(14 total points)*

Probabilities for each character in a 4-letter alphabet appear below.

W	X	Y	Z
0.4	0.1	0.3	0.2

[a] **Construct** a Huffman tree for these characters. *(6 points)*



[b] Extract the code for each character from your tree. *(5 points)*

Character	Code	
W	1	00
X	000	01
Y	01	10
Z	001	11

$\sum$ 1,1,3,1,2,1,3

[c] How many bits does it take to represent the string WWXWYWZ using this code? How many bits does it take to represent the string WWXWYWZ using the best fixed-width representation for this alphabet? *(3 points)*

2 2 2 2 2 2 2

$\sum 1,1,3,1,2,1,3 = 12$ $\qquad$ $7(2) = 14$ <—

Num of bits w/ Huffman code

Num of bits w/ fixed width code

Problem 11 *(10 points)*
John has the following recursive algorithm:

```
SIMPLERECURSIVEALGORITHM(S, I, J)
  if j ≤ i then
    return 0
  else if S[i] + S[j] = 3 then
    return SIMPLERECURSIVEALGORITHM(S, i + 1, j − 1)
  else
    a ← SIMPLERECURSIVEALGORITHM(S, i + 1, j) + i
    b ← SIMPLERECURSIVEALGORITHM(S, i, j − 1) + j
    return max(a, b)
  end if
```

This algorithm works correctly but is too slow for John, so he attempts to implement the same idea as top-down dynamic programming to make it faster. He creates an appropriately-sized 2d array of integers called p, initializes each element of p to -1, and then uses p in this revised algorithm:

```
FANCYRECURSIVEALGORITHM(S, I, J)
  if i = j then
    p[i, j] ← 0
  else if S[i] + S[j] = 3 then
    p[i, j] ← FANCYRECURSIVEALGORITHM(S, i + 1, j − 1)
  else
    a ← FANCYRECURSIVEALGORITHM(S, i + 1, j) + i
    b ← FANCYRECURSIVEALGORITHM(S, i, j − 1) + j
    p[i, j] ← max(a, b)
  end if
  return p[i, j]
```

Unfortunately, John notices that this revised algorithm does not run any faster than the original.

Explain where John went wrong, by describing in a sentence or two why FANCYRECURSIVEALGORITHM is not a correct top-down dynamic programming algorithm. **Show**, either by clearly showing modifications to the pseudocode above or by writing new pseudocode, how to do it correctly.

This algorithm fails to run faster beause John does not check if p[i,j] == -1, by failing todo this there are still subproblems executed twice

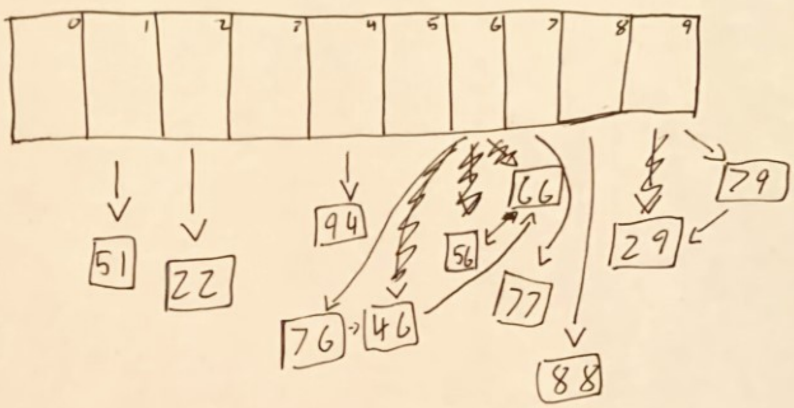
```
FancyRA(S, i, equals
  if p[i,j] -1 then
    if i = j then
      p[i,j] ← 0
    else if S[i]+S[j]=3 then
      p[i,j] ← FancyRA(S, i+1, j-1)
    else
      a ← FancyRA(S, i+1, j) + i
      b ← FancyRA(S, i, j-1) + i
      P[i,j] ← max(a, b)
    end if
  end if
  return p[i,j]
```

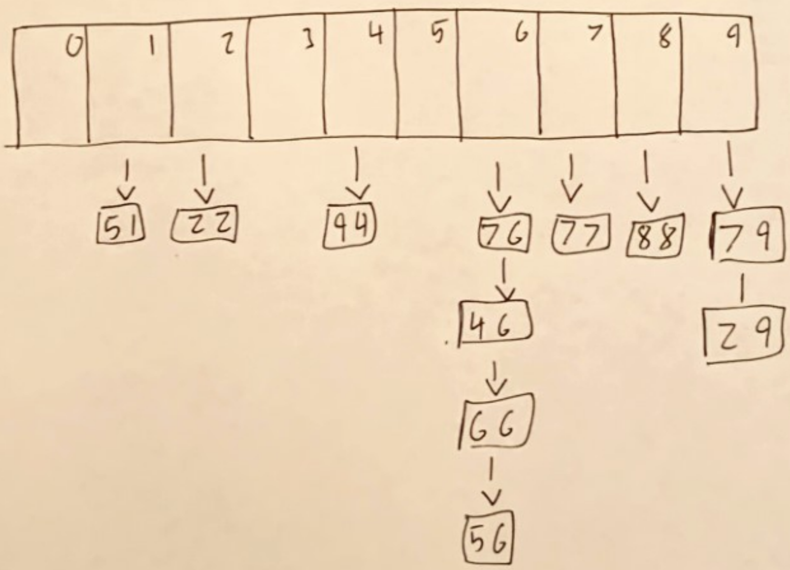
Problem 12 *(10 points)*

Demonstrate what happens when we insert the keys 56, 29, 51, 66, 77, 79, 94, 46, 88, 22, 76 into a hash table with 10 slots. Resolve collisions by linked-list chaining. Use $h(k) = k \bmod 10$ as the hash function. **Show** the completed hash table, along with enough work to demonstrate that you know what you're doing.

$h(k) = k \bmod 10$

56 $h(56) = 6$
29 $h(29) = 9$
51 $h(51) = 1$
66 $h(66) = 6$
77 $h(77) = 7$
79 $h(79) = 9$
94 $h(94) = 4$
46 $h(46) = 6$
88 $h(88) = 8$
22 $h(22) = 2$
76 $h(76) = 6$





Problem 13 *(10 points)*

In this class, we learned two algorithms for solving the knapsack problem: one based on a brute-force exhaustive search and one based on dynamic programming.

Write an instance of the Knapsack problem for which the brute force algorithm would be *significantly faster* than the dynamic programming algorithm.[2] Include all parts of your example instance, including the number of items, the weight and value of each item, and the available capacity. **Explain**, using a sentence or two, why your answer is correct.

Capacity 10

i	A	B	C
w_i	1	2	3
v_i	10	10	10

$i = 3$

Total $w = 6 < 10$

Total value $= 30$

Because we have a small number of items generating all permutations ~~[illegible]~~ can be done ~~generated~~ quickly. With no permutations exceeding the capacity we simply need to choose the one with the highest value

[2]Hint: The answer, "There are no such instances!" is not correct.

Evan Owre final note sheet

$Euclids(m, n) \to Euclids(n, m \bmod n)$
$Euclids(m, 0) \to m$
Adjency matrix & lists

Ch1

an algorithm is a sequence of unambiguous instructions for solving a problem

① understand prob.
Input: what look like
Output: correct ans

② Decide on
Design technic computational means
Data structs
Exact vs. approx

③ Design algo
④ Prove correctness
⑤ Analyze
⑥ Implement

Ch2

$O(n): t(n) \in O(g(n))$ if $t(n) \le c\,g(n)$

$\Theta(n): t(n) \in \Theta(g(n))$ if $c_1 g(n) \le t(n) \le c_2 g(n)$

$\Omega(n): t(n) \in \Omega(g(n))$ if $c\,g(n) \le t(n)$

$\Theta(1, \log n, n, n\log n, n^2, n^3, 2^n, n!)$

$L = \lim \frac{t}{g}$: $L = 0$ O, $L < \infty$ Θ, $= \infty$ Ω

$n^k - O(n^k)$, $\log_a n - \Theta(\log_b n)$

$t_1 + t_2 \in \Theta(\max(g_1, g_2))$

basic operation time & space eff
comparing keys
↳ best → best time eff
↳ worst → worst time eff
avg → avg time eff

Ch3 Brute force

Intpow(a, n)
$r \leftarrow 1$
$i \to n$
$r = r \cdot a$
ret r

SelectionSort
find smallest
put first
continue
$\Theta(n^2)$

StringMatch(P, T) indx in T w/ P
$0 \to m$
check for pattern
Shifting Θ ~
$\Theta(mn)$

Exhaustive Search
try all possibilities
choose shortest or best

Ch4 Decrease & conquer - smaller subprob → extend to solution

↳ const, const factor, var size

IntPow(a, n) $\Theta(\log n)$
n even - $a^n = a^{n/2} \cdot a^{n/2} = (a^{n/2})^2$
n odd - $a^n = a^{n/2} \cdot a^{n/2} \cdot a = (a^{n/2})^2 \cdot a$
$n = 1$ $a^n = a$

insertSort
sort n-1 find correct indx for last elem
worst $\Theta(n^2)$
best $\Theta(n)$

BS (sorted Array, K)
compare k to middle
cut in half

QuickSelect(A, k)
Partition → $s = k$ out s
recursive call on half failing

Partition(A)
out (A w/ pivot @ final place)

Ch5 Divide & conquer - several sub probs recursion, combine

master Theorem $T(n) = aT(n/b) + f(n)$ - $f(n) \in \Theta(n^d)$
$a < b^d$ n^d
$a = b^d$ $n^d \log n$
$a > b^d$ $n^{\log_b a}$

Karatsuba(a, b) $m = \lceil n/2 \rceil$
$a = 10^m a_1 + a_0$, $b = 10^m b_1 + b_0$
$ab = 10^{2m} c_2 + 10^m c_1 + c_0$
$c_2 = a_1 b_1$, $c_1 = (a_0 + a_1)(b_0 + b_1) + c_0$, $c_0 = a_0 b_0$

MergeSort $\Theta(n \log n)$
QuickSort $\Theta(n \log n)$
Portion continually puts element in correct pos, recursion on left & right

Ch6 Transform & conquer → change to easier to solve form

↳ simpler instance, diff rep, diff prob

Heaps "bubble" up/down parent $A[\lfloor (i-1)/2 \rfloor]$
$A[0]$ root $A[i] = A[i/2 + 1]$ $A[i/2 + 2]$

AVL tree, BST
↳ check BF = LCH - RCH
↳ rotate if necessary

2-3 tree
2-Node 1K 2C
3-Node 2K 3C
↳ No rotate builds upward

Ch7 Space-Time trade off

HaSH table - spread keys throught array size m using hash function h
memory efficent, hard to delete

h: Deterministic, Uniform, Efficient

chaining, collision occurs
"open" use LL to store mult key
Probing, h(k) occupied, try elsewhere
"closed"

Ch8 Dynamic Programing, similar to Divide & conquer

avoids redundant calls
memoization, store computed vals → Top down
DP table, store vals in table
Bottom up

→ Identify family of subprobs final
→ show how related w/ recurrence special case
→ Identify base case
→ form algo - fill table of recurrence vals
→ extract final sol

Ch9 Greedy algo - make the best choice each time

↳ Feasible, locally optimal, Irrevocable

Prims algo - Start w/ rand a node, connect closest node, Repeat V-1 times

Kruskals - add lightest edge that does not create cycle

Hoffman codes, use character probability
- variable width
- Pre-fix free
Out: pre-fix free binary representation minimizing length