College of Engineering & Applied Sciences

CSPB 2824

Discrete Structures

Algorithms And Number Properties Mastery Workbook

Taylor Larrechea

2024

Mastery Workbook 5

Algorithms And Number Properties Workbook

I have neither given nor received unauthorized assistance.

Taylor James Larrechea

Problems

Mastery Workbook Quiz Format

This week's mastery workbook was done in a quiz format and can be found in the following pages.

CSPB 2824 - Stade - Discrete Structures

Started on	Thursday, 28 September 2023, 10:54 PM
State	Finished
Completed on	Thursday, 28 September 2023, 10:56 PM
Time taken	1 min 33 secs
Grade	Not yet graded

Question **1**

Correct

Mark 5.00 out of 5.00

Take 15 index cards and write the numbers 1-15 on one side.

1. Shuffle the cards.
2. Sort the cards naively (that is without any particular method) in order from smallest to largest.
3. Note how many times you moved a card before it was completely sorted
4. Repeat 1-3 a total of 3 times

What is true about this process?

Select one or more:

- [x] a. I really did try this out - honest.
- [ ] b. There is only one way to sort the cards.
- [x] c. Informally trying out a problem without focusing on the outcome can help with understanding complex problems before trying to solve them.
- [ ] d. Repeatedly shuffling the cards until they randomly fall in order is an effective strategy.
- [ ] e. It is completely obvious what is the best way to sort the cards.
- [ ] f. The number of cards moved was the same each time.

Correct

Marks for this submission: 5.00/5.00.

Question **2**

Complete

Marked out of 20.00

Do the Old School Binary Sorting Activity from Moodle.

Add a single photo of your project in the response box.

Please "drag and drop" a photo into the response box below or click on the landscape icon to upload photo directly.

I have attached the files for this activity below. I want to preface that I did not have index cards or a hole punch, so I just cut up two pieces of 8.5 x 11 paper into 8 parts each. I then used a thumb tack to put holes in the card and cut out the slots with scissors. Overall the process worked and I was very amazed at how well this did with sorting.

IMG_0755.png

IMG_0756.png

IMG_0758.png

Question **3**

Correct

Mark 5.00 out of 5.00

How many cards can be sorting if each card has 5 holes instead of 4 ?

Select one:

- a. 32
- b. 64
- c. 62
- d. 31
- e. 5
- f. none, this only works with 4 holes

Correct

Marks for this submission: 5.00/5.00.

Question **4**

Correct

Mark 5.00 out of 5.00

If each card had 10 holes, then more than 1000 cards could be sorted.

How how many steps would it take to sort them? (using the same process)

Answer: 10

Correct

Marks for this submission: 5.00/5.00.

Question **5**

Complete

Marked out of 5.00

Look up Radix Sort. Cite your sources to answer.

- How is Radix sort related to the Old School Sorting Trick
- How efficient is Radix sort?
- What do sources say about its complexity?

How is Radix sort related to the Old School Sorting Trick?

Radix sort works by first sorting a set of numbers by the least significant digit (the right most digit. e.g in 102, the right most digit is 2). Once this has been completed, it moves to the next digit (to the left if the number has more than one digit) and sorts the numbers based on that digit. Once all of the digits have been iterated through, the numbers are sorted in ascending order (or descending depending on the algorithm's) implementation.

The old school sorting trick does something similar in that when you shake the cards of the right most value (1 in our case), all of the cards that fell are then placed on top of the ones that remain. We then sort the cards that fell in ascending order where the smallest number is on the bottom and repeat the same process with 2, then 4, and then 8. This of course can be expanded for other numbers of cards (32 in the case where our cards have 5 holes).

Radix sort and the Old School Sorting Trick are similar in that they "shake out" the values after each iteration. They then are sorted based on what digit we are currently working on.

How efficient is Radix sort?

According to https://www.simplilearn.com/tutorials/data-structure-tutorial/radix-sort#:~:text=Radix%20Sort's%20time%20complexity%20of,of%20elements%20with%20equal%20values., the Radix Sort algorithm is O(nd) where n is the size of the array or container of values and d is the number of digits of the largest value in the array. If we generalize this, we can say that the complexity of Radix Sort is on the order of O(n) where n is some integer value.

What do sources say about its complexity?

Like I stated before, sources say the complexity of the algorithm is dependent upon the number of elements in the array and the number of digits that are in the largest number in the array. Wikipedia also corroborates my previous statement and that with simplilearn as well: https://en.wikipedia.org/wiki/Radix_sort.

Question **6**

Correct

Mark 5.00 out of 5.00

Recall our Puzzle from Week 1.

Match the following - can you solve without a key?

Use:

V for big wedge

v for little wedge

< for side wedge

Discuss on Piazza - How is this different, and how is it the same as our standard base systems?

V<vv	72
V<<vvvvv	85
<vvv	13
VVV<<v	201
VVVVV	300
VVV<vvvvvvvv	199

Correct

Marks for this submission: 5.00/5.00.

Question **7**

Correct

Mark 5.00 out of 5.00

How might we write Babylonian numbers using our standard digits?

**Consider the "hundreds place" for 60s, "tens place" for 10s, "1s place" for 1s.*

Convert the following to Babylonian using standard digits using this method.

Discuss in Piazza:

- Why don't need to use letters like we do for base 16 ?
- Why is this not the 'standard' base 60? (what would that look like?)
- What is the largest number we can express with this method using "3 digits."

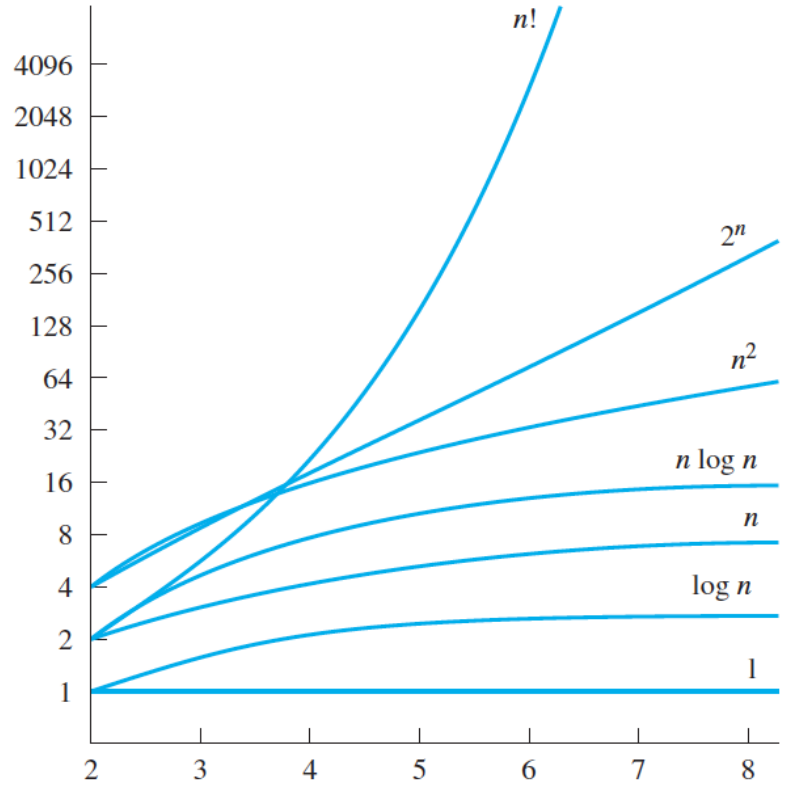
182_{10}	302
99_{10}	139
124_{10}	204
13_{10}	13
47_{10}	47
72_{10}	112

Correct

Marks for this submission: 5.00/5.00.

Information

For reference if needed.



Question **8**

Correct

Mark 10.00 out of 10.00

Look at the Big O chart above.

Which of the following are true?

Select one or more:

- [x] a. An algorithm that has complexity log n is considered better (runs faster) than one that has complexity n.
- [ ] b. An algorithm that has complexity n is considered better (runs faster) than one that has complexity log n.
- [x] c. An algorithm that has complexity log n is considered better (runs faster) than one that has complexity n!.
- [ ] d. An algorithm that has complexity n! is considered better (runs faster) than one that has complexity log n.

Correct

Marks for this submission: 10.00/10.00.

Question **9**

Correct

Mark 20.00 out of 20.00

You are asked to write a function

```
hex_to_decimal(lst)
```

wherein given a hexadecimal number represented as a list of numbers between 0 - 15 (using A,B,C,D,E for 10, 11, 12, 13, 14, 15), and convert it to an equivalent decimal number.

The table below shows how a hexadecimal number is represented as a list (and as a review of what hex digits stand for).

Hexadecimal numbers represented as a list

ABCDEF	**[10,11,12,13,14,15]**
A013F	[10, 0, 1, 3, 15]
B0B0FED	[11,0,11,0,15,14,13]
CD985011	[12,13,9,8,5,0, 1, 1]
9875431	[9,8,7,5,3,4,1]

Before you start coding, review the test cases below.
In particular, notice the format of the 'letter' hex digits in the input list for this function.

To Check MAKE SURE:
You have defined the function with the name given.
Do NOT have any print statements.
Do not include your own test case.

The autograder will run its own tests, if you include tests, the autograder won't work.

SPOILER ALERT: Here is an answer. You may use this one or try your own.

```
def hex_to_decimal(lst):
    dec = 0
    lst.reverse()
    hex = "0123456789ABCDEF"
    for i in range(len(lst)):
        value = hex.index(str(lst[i]))
        dec+= value*16**i
    return dec
```

For example:

Test	Result
`print(hex_to_decimal([]))`	`0`
`print(hex_to_decimal([1,0]))`	`16`
`print(hex_to_decimal([1,1,6]))`	`278`
`print(hex_to_decimal([1,"F",6]))`	`502`

Answer: (penalty regime: 10, 20, ... %)

```
def hex_to_decimal(lst):

    dec = 0

    lst.reverse()

    hex = "0123456789ABCDEF"
```

```
    for i in range(len(lst)):

        value = hex.index(str(lst[i]))

        dec += value*16**i

    return dec
```

	Test	Expected	Got	
✔	print(hex_to_decimal([]))	0	0	✔
✔	print(hex_to_decimal([1,0]))	16	16	✔
✔	print(hex_to_decimal([1,1,6]))	278	278	✔
✔	print(hex_to_decimal([1,"F",6]))	502	502	✔
✔	print(hex_to_decimal([1,"A","B","C",4,5]))	1752133	1752133	✔
✔	print(hex_to_decimal(["D","D","D","E","F"]))	908783	908783	✔
✔	print(hex_to_decimal([1,8,4,5,2,7,"F","E"]))	407185406	407185406	✔
✔	print(hex_to_decimal([3,4,5]))	837	837	✔
✔	print(hex_to_decimal(["D"]))	13	13	✔
✔	print(hex_to_decimal([3]))	3	3	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.

Question **10**

Correct

Mark 20.00 out of 20.00

Write a Python function `DecToBin()` that takes in a nonnegative integer `d` and returns a Python list of `0`'s and `1`'s corresponding to the binary representation of `d`.

To Check MAKE SURE:
You have defined the function with the name given.
Do NOT have any print statements.
Do not include your own test case.

The autograder will run it's own tests, if you include tests, the autograder won't work.

For example:

Test	Result
`print(DecToBin(0))`	`[0]`
`print(DecToBin(10))`	`[1, 0, 1, 0]`
`print(DecToBin(241))`	`[1, 1, 1, 1, 0, 0, 0, 1]`

Answer: (penalty regime: 0 %)

```
import math

def DecToBin(input):

    bin = []

    binVal = 0

    quotient = math.floor(input / 2)

    binVal = input - 2 * quotient

    bin.append(binVal)

    while (quotient >= 1):

        priorQuotient = quotient

        quotient = math.floor(priorQuotient / 2)

        binVal = priorQuotient - 2 * quotient
```

	Test	Expected	Got	
✔	`print(DecToBin(0))`	`[0]`	`[0]`	✔
✔	`print(DecToBin(10))`	`[1, 0, 1, 0]`	`[1, 0, 1, 0]`	✔
✔	`print(DecToBin(241))`	`[1, 1, 1, 1, 0, 0, 0, 1]`	`[1, 1, 1, 1, 0, 0, 0, 1]`	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.