CS 3000: Algorithms & Data — Fall 2024

Due Tuesday Sep 17 at 11:59pm via Gradescope

Name:
Collaborators:

- Make sure to put your name on the first page. If you are using the LaTeX template we provided, then you can make sure it appears by filling in the `yourname` command.
- This homework is due Tuesday Sep 17 at 11:59pm via Gradescope. No late assignments will be accepted. Make sure to submit something before the deadline.
- Solutions must be typeset, preferrably in LaTeX. If you need to draw any diagrams, you may draw them by hand as long as they are embedded in the PDF. We recommend that you use the source file for this assignment to get started, which can be accessed from

this link:
https://www.overleaf.com/read/fpqrfpcbwkqp#711733

- We encourage you to work with your classmates on the homework problems, but also urge you to attempt all of the problems by yourself first. If you do collaborate, you must write all solutions by yourself, in your own words. Do not submit anything you cannot explain. Please list all your collaborators in your solution for each problem by filling in the `yourcollaborators` command.
- Finding solutions to homework problems on the web, or by asking students not enrolled in the class is strictly forbidden.

Problem 1. *(8 + 7 = 15 points) What does this code do?*

You encounter the following mysterious piece of code.

Algorithm 1: Mystery Function

```
Function F(n):
    If n = 0 :
        Return (2, 1)
    Else
        b ← 1
        For i from 1 to n
            b ← 2b
        (u, v) ← F(n − 1)
        Return (u + b, v · b)
```

(a) What are the results of $F(1)$, $F(2)$, $F(3)$, and $F(4)$?

Solution:

F(1) = (4,2), F(2) = (8,8), F(3) = (16,64), F(4) = (32,1024)

(b) What does the code do in general, when given input integer $n \geq 0$? Prove your assertion by induction on n.

Base Case: $F(0) = (2^{0+1}, 2^{0(0+1)/2}) = (2, 1)$

Inductive Step: Assume for $k \geq 0$, $F(k) = (2^{k+1}, 2^{k(k+1)/2})$

Then, $(u, v) = F(k)$

$F(k+1) = (u + b, v \times b)$

and $b = 2b_{F(k)}$

$F(k+1) = (u + b, v \times b)$ and $b = 2b$ *so*, $(u + 2^{k+1}, v \times 2^{k+1})$

Since $u = 2^{k+1}, v = 2^{k(k+1)/2}$
$= (2^{k+1} + 2^{k+1}, 2^{k(k+1)/2} \times 2^{k+1})$
$= (2^{k+2}, 2^{k(k+1)/2} \times 2^{2(k+1)/2})$
$= (2^{k+2}, 2^{((2+k)(k+1))/2})$
$= (2^{k+2}, 2^{(k+1)(k+2)/2})$

Problem 2. *(15 points) Making exact change*

You are provided with an infinite supply of coins of values 3 and 5. Your task is to prove, using induction, that for any integer $n \geq 8$ you can pay exactly n using the coins you have.

Hint: Use strong induction.

Solution:

Base Case:
n= 8, 5 + 3 = 8
n= 9, 3 + 3 + 3 = 8
n= 10, 5 + 5 = 10
n= 11, 5 + 3 + 3 = 11
n = 12, 3 + 3 + 3 + 3 = 12

Inductive Step: Assume i can be represented with 3 and 5 coins and $12 \leqslant i \leqslant k$, where k is some integer greater than 12.

Then, P(k+1) can be represented by a combination of 3 and 5 coins counting backwards in increments of 3 and/or 5 until a base case is reached, proving that a combo of 3 and 5 coins can compose a value of P(k+1).

Problem 3. *(3 + 12 = 15 points) Row-column ordered arrays*

We say that an $n \times n$ 2-D array A of integers is row-column ordered if it satisfies the following property: the entries of each row are sorted from left to right and the entries of each column are sorted from top to bottom. Thus, for any $1 \leq i \leq n$, we have

$$A[i,1] \leq A[i,2] \leq \ldots \leq A[i,n],$$

and for $1 \leq j \leq n$, we have

$$A[1,j] \leq A[2,j] \leq \ldots \leq A[n,j].$$

For each of the tasks below, present your algorithm in pseudocode.

(a) THIRD-MIN(A): returns the 3rd smallest element in A. Your algorithm should use at most 10 comparisons among the elements of A.

(b) CHANGE-KEY(A, i, j, k): changes the value of $A[i,j]$ to k and updates A so that it continues to be row-column ordered. Your algorithm must make at most $10n$ comparisons.

Solution:

a.

$third - min(A)$:

```
    i = 1
    j = 1
    n = 2
    while n != 0
        if A[i+1, j] < A[i, j + 1]
            i = i + 1
            n = n − 1
        else if A[i+1, j] > A[i, j + 1]
            j = j + 1
            n = n − 1
        else
            i = i + 1
```

b.

$Change - Key(A, i, j, k)$:

```
    A[i, j] = k
    For x = ((i x n) + j) - 1
        min_pos = x
        For k = (i x n) + j
            If(A[k ÷ n, k  mod n)] < A[min_pos ÷ n, min_pos  mod n)]
                min_pos = k
        Swap A[min_pos ÷ n, min_pos  mod n)] and (A[x ÷ n, x  mod n)]
```

Problem 4. *(2 + 10 + 3 = 15 points) Top k baby names*

You have a massive repository of baby names, extracted from a census report, which is a list of n names of all babies (including duplicates) within a region. You are asked to compute the k most popular baby names, in sorted order of popularity; here k is a positive integer less than number of distinct names in the list.

(a) State clearly the input and output for the given problem.

(b) Describe an algorithm, in pseudocode, to solve the problem. Your algorithm may use any of the algorithms we have covered in class.

(c) Analyze the worst-case running time of your algorithm.

Solution: Type your solution here.
a. input is n, the list of all babies (with duplicates) and k, the number of most popular baby names we want.

b.

```
algorithm(n, k):
    namesValues = []
    contains = false

    For name in babyNames:
        For pairs in nameValues:
            If pairs[0] is name:
                pairs[1]++
                contains = true

        If contains = false:
            namesValues.append(name, 1)
        contains = false

    MergeSort the namesValue by their frequency numbers (first index of each pair)

    nameValues = [:k]
    topNames = []
    For names in nameValues:
        topNames.append(names[0)]

    return topNames
```

c. $n^2 + (n \times \log_2 n)$ which is upper bounded by n^2, so $n^2$is the time complexity