CPSC 331 HW2_Written

Ali Akbari

TOTAL POINTS

14.75 / 15

QUESTION 1

1 Recursive Selection Sort 5 / 5

✓ **- 0 pts** Correct

- 2 pts the algorithm is not correct.

- 1 pts recurrence relation is not correct or not discussed properly.

- 0.75 pts substitution method is incorrect or is not discussed.

- 1 pts the execution time is not theta(n^2) or is not discussed.

- 1 pts the algorithm is partially correct

- 0.5 pts recurrence relation is partially correct or not determined precisely.

- 0.5 pts there is no stop criteria for the recursive function.

- 3 pts the algorithm is not provided.

- 0.5 pts part of code is missed.

- 5 pts Answer is not provided.

- 1.5 pts There is no recursion call for the sort algorithm or algorithm is partially correct.

the for loop should start from "lowVal" not "lowVal+1"

QUESTION 2

2 Catalan Numbers 4 / 4

✓ **- 0 pts** Correct

- 0.5 pts the complete tree is not drawn.

- 0.5 pts The relation ship among terms is not shown in the tree. It is not obvious where summation and where multiplication is happening.

- 1 pts The complete tree is not drawn and the resulting number is not calculated.

- 1 pts The tree is not expanded to show each term as a node of the tree.

- 0.25 pts The value of c3 is not calculated.

- 1 pts Half of the tree looks like erased and the tree looks incomplte.

- 1.5 pts No recursion tree has provided.

- 3 pts The tree provided does not look like a recursion tree and the computation is not shown or it looks like an incorrect one.

- 4 pts Answer is not provided.

- 0.5 pts C1 in not expanded.

QUESTION 3

3 Iterative Substitution 2.75 / 3

- 0 pts Correct

- 1 pts The execution time of the second part is not correctly calculated.

- 1.25 pts The execution time of the second part is not calculated and the total execution time is not discussed.

✓ **- 0.25 pts a tiny problem at the calculation of the second part of the execution time or first part.**

- 0.25 pts problem in summation of the two terms related to (2^(n-1))

- 0.75 pts Problem at reaching to T(1)

- 0.25 pts problem at calculating the closed form of the second part.

- 0.5 pts deformed closed form and the execution time is difficult to infer.

- 2 pts incorrect response while part of the calculation has been correct.

- 2 pts inability in reaching to a closed form.

- 3 pts incorrect or no answer is provided.

- 0.5 pts tiny problem at calculation of both part.

QUESTION 4

4 Induction 3 / 3

✓ **- 0 pts** Correct

- 0.75 pts base case is not discussed or is incorrect.

- **0.25 pts** base case is partially correct.
- **0.25 pts** The same notation is used for the number of summation and the fib number. It has reduced readability.(In the base part)
- **0.25 pts** base case does not include T(1) but it is discussed before.
- **1 pts** Induction case does not make sense or is incomplete.
- **0.5 pts** The same notation is used for the number of summation and the fib number. It has reduced readability.
- **0.75 pts** Base case is not discussed for T(0) and T(1).
- **3 pts** No Answer is provided or the answer is incorrect.
- **1.5 pts** The relation for the number of addition is not calculated correctly and as a result induction part is incorrect.
- **1.5 pts** induction step is not provided.

Question #1

Code :

```
public static void sort(double[] arr) {
            selectionSort(arr, 0, arr.length - 1);
      }

      public static void selectionSort(double[] arr, int lowVal, int highVal) {
            // if n = 1, lowVal = highVal = 0
            // Return array of size 1, cost = 1
            if (lowVal == highVal) {
                  Return arr;
            }
             if (lowVal < highVal) {
                  int minimumIndex = lowVal;
                  double minimum = arr[lowVal];
                  for (int i = lowVal + 1; i <= highVal; i++) {
                        if (arr[i] < minimum) {
                              minimum = arr[i];
                              minimumIndex = i;
                        }
                  }

                  // Swap the smallest number in array
                  arr[minimumIndex] = arr[lowVal];
                  arr[lowVal] = minimum;

                  // Sort the remaining array
                  // Low is incremented meaning f(n-1)
                  selectionSort(arr, lowVal + 1, highVal);
             }
      }
```

Based on the code if n = 1 then the array is returned in the first if statement.
From the code, we see that the number of comparisons is 1 per loop and the loop runs for the array length, which is n. And with the recursion part, we see that we are sending n - 1 as Low increments to the function itself.
So, from this we get
Let T(n) be the total number of comparisons
T(1) = 1
T(n) = T(n-1) + n   n comparisons and T(n-1) recursion calls

Recurrence Relationship = T(n) = T(n-1) + n

By substitution we the following:
T(1) = 1
T(2) = T(2-1) + 2 = T(1) + 2 = 1 + 2 = 3
T(3) = T(3-1) + 3 = T(2) + 3 = 1 + 2 + 3 = 6
T(4) = T(4-1) + 4 = T(3) + 4 = 1 + 2 + 3 + 4 = 10

Transforming it to summation we see that
$\sum_{i=1}^{n}$ 1+2+3+4...n
By using the arithmetic formula we get:
$T(n) = \frac{n(n+1)}{2} = \frac{n^2+n}{2}$
Thus by referring to HW1,
$T(n) = \Theta(n^2)$

Question #2

$C_0 = 1;$

$C_{n+1} = \sum_{i=0}^{n} = C_i C_{n-i}\ , \ \ n \geq 0.$

$C_2 = ?$

$C_{1+1} = \sum_{i=0}^{1} = C_i C_{1-i} = C_0 C_1 + C_1 C_0 = 1 * 1 + 1 * 1 = 2$

$C_3 = ?$

$C_{2+1} = \sum_{i=0}^{2} = C_i C_{2-i} = C_0 C_2 + C_1 C_1 + C_2 C_0 = 1 * 2 + 1 * 1 + 2 * 1 = 5$

C_n	*Output*
C_0	1
C_1	1
C_2	2
C_3	5

1 Recursive Selection Sort 5 / 5

✓ **- 0 pts** **Correct**

- 2 pts the algorithm is not correct.

- 1 pts recurrence relation is not correct or not discussed properly.

- 0.75 pts substitution method is incorrect or is not discussed.

- 1 pts the execution time is not theta(n^2) or is not discussed.

- 1 pts the algorithm is partially correct

- 0.5 pts recurrence relation is partially correct or not determined precisely.

- 0.5 pts there is no stop criteria for the recursive function.

- 3 pts the algorithm is not provided.

- 0.5 pts part of code is missed.

- 5 pts Answer is not provided.

- 1.5 pts There is no recursion call for the sort algorithm or algorithm is partially correct.

the for loop should start from "lowVal" not "lowVal+1"

By substitution we the following:
T(1) = 1
T(2) = T(2-1) + 2 = T(1) + 2 = 1 + 2 = 3
T(3) = T(3-1) + 3 = T(2) + 3 = 1 + 2 + 3 = 6
T(4) = T(4-1) + 4 = T(3) + 4 = 1 + 2 + 3 + 4 = 10

Transforming it to summation we see that
$\sum_{i=1}^{n}$ 1+2+3+4...n
By using the arithmetic formula we get:
$T(n) = \frac{n(n+1)}{2} = \frac{n^2+n}{2}$
Thus by referring to HW1,
$T(n) = \Theta(n^2)$

Question #2

$C_0 = 1;$

$C_{n+1} = \sum_{i=0}^{n} = C_i C_{n-i}, \; n \geq 0.$

$C_2 = ?$

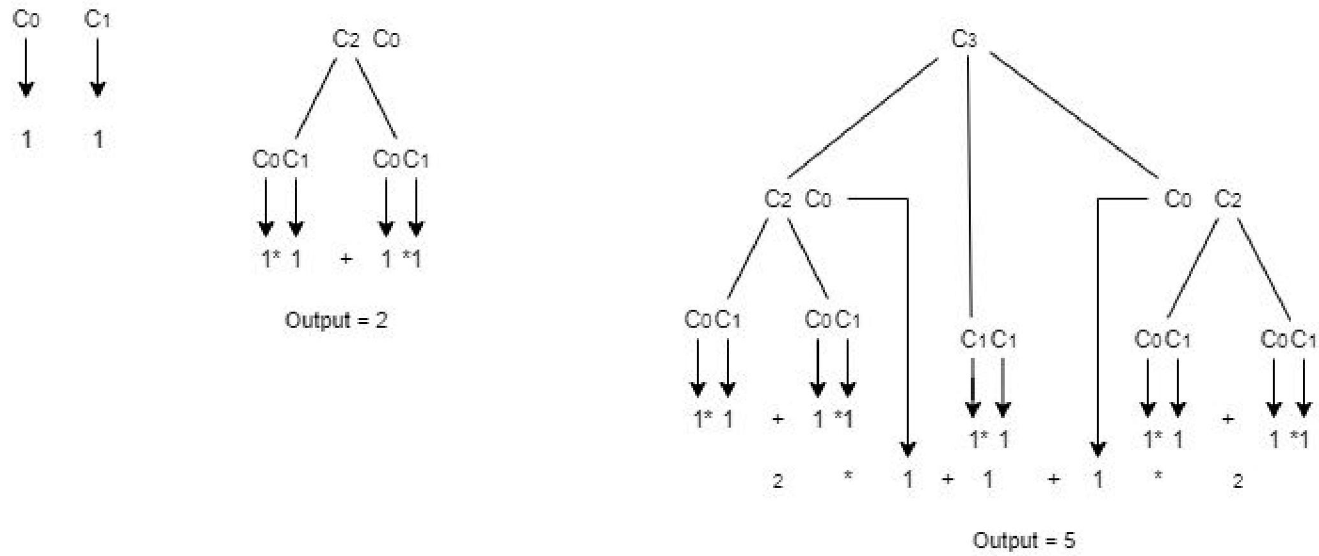
$C_{1+1} = \sum_{i=0}^{1} = C_i C_{1-i} = C_0 C_1 + C_1 C_0 = 1 * 1 + 1 * 1 = 2$

$C_3 = ?$

$C_{2+1} = \sum_{i=0}^{2} = C_i C_{2-i} = C_0 C_2 + C_1 C_1 + C_2 C_0 = 1 * 2 + 1 * 1 + 2 * 1 = 5$

C_n	*Output*
C_0	1
C_1	1
C_2	2
C_3	5

Recursion Call Trees of Catalan number up to C_3.



Question #3

$T(n) = 2T(n-1) + 1,\ (for\ n > 1),$

$T(1) = a,\ where\ a > 0.$

By iterative subsitution :

$T(n) \quad = 2T(n-1) + 1$ **Iteration # 1**

$\quad\quad = 2(2T(n-2)+1) + 1$

$T(n) \quad = 2^2T(n-2) + 2 + 1$ **Iteration # 2**

$\quad\quad = 2^2(2T(n-3)+1) + 2 + 1$

$T(n) \quad = 2^3T(n-3) + 2^2 + 2 + 1$ **Iteration # 3**

.

.

.

.

If this is continued for k iterations then we could generalize it in terms of k.

$T(n) \quad = 2^kT(n-k) + 2^{k-1} + 2^{k-2} + 2^{k-3} .... + 2^0$

Since we know that n - k is decreasing untill atleast 1 form our base case $T(1) = a$, assume n - k = 1. Which means n = k + 1 and k = n - 1.

So,

2 Catalan Numbers **4 / 4**

✓ **- 0 pts** **Correct**

- 0.5 pts the complete tree is not drawn.

- 0.5 pts The relation ship among terms is not shown in the tree. It is not obvious where summation and where multiplication is happening.

- 1 pts The complete tree is not drawn and the resulting number is not calculated.

- 1 pts The tree is not expanded to show each term as a node of the tree.

- 0.25 pts The value of c3 is not calculated.

- 1 pts Half of the tree looks like erased and the tree looks incomplte.

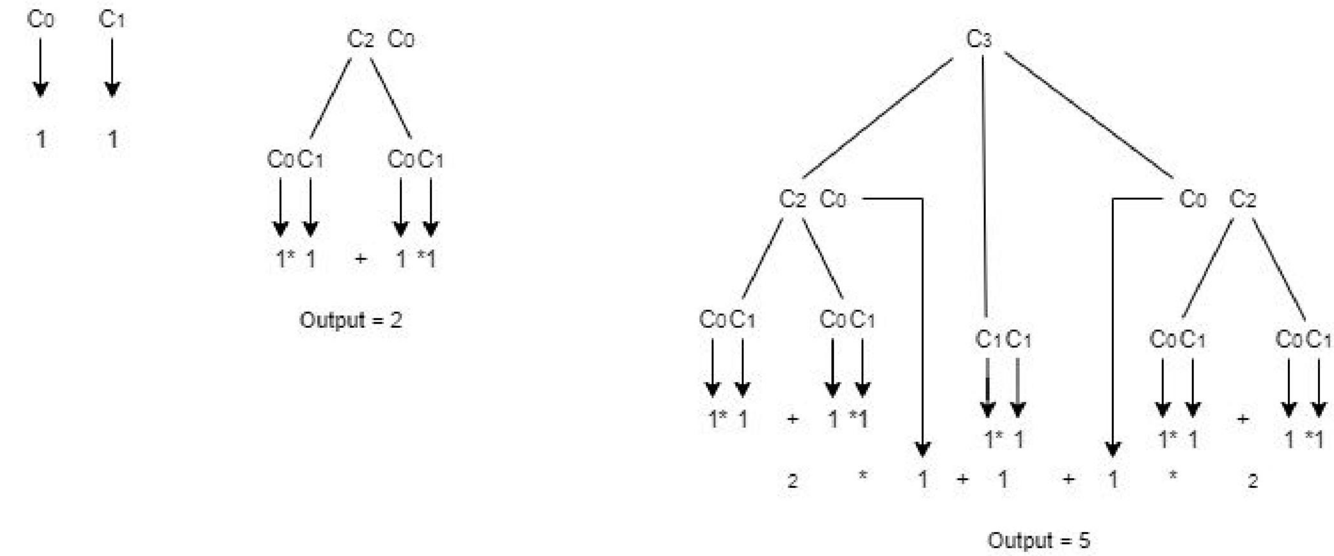
- 1.5 pts No recursion tree has provided.

- 3 pts The tree provided does not look like a recursion tree and the computation is not shown or it looks like an incorrect one.

- 4 pts Answer is not provided.

- 0.5 pts C1 in not expanded.

Recursion Call Trees of Catalan number up to C_3.



Question #3

$T(n) = 2T(n-1) + 1, \ (for\ n > 1),$

$T(1) = a, \ where\ a > 0.$

By iterative subsitution :

$T(n) \quad = 2T(n-1) + 1$ **Iteration # 1**

$\qquad\quad = 2(2T(n-2)+1) + 1$

$T(n) \quad = 2^2 T(n-2) + 2 + 1$ **Iteration # 2**

$\qquad\quad = 2^2(2T(n-3)+1) + 2 + 1$

$T(n) \quad = 2^3 T(n-3) + 2^2 + 2 + 1$ **Iteration # 3**

.

.

.

.

If this is continued for k iterations then we could generalize it in terms of k.

$T(n) \quad = 2^k T(n-k) + 2^{k-1} + 2^{k-2} + 2^{k-3} .... + 2^0$

Since we know that n - k is decreasing untill atleast 1 form our base case $T(1) = a$, assume n - k = 1. Which means n = k + 1 and k = n - 1.

So,

$$T(n) = 2^{n-1}T(1) + 2^{n-2} + 2^{n-3} + 2^{n-4} .... + 2^0$$
$$= 2^{n-1}a + 2^{n-2} + 2^{n-3} + 2^{n-4} .... + 2^0$$

If we rearrange for the numbers around we get:

$= 2^{n-1}a + 2^0 + 2^1 + 2^2 .... + 2^{n-2} \Rightarrow 2^{n-1}a$ + 1 + 2 + 4….+ 2^(n-2)

The higshlighted part can be simplified to $\sum_{i=0}^{n-2} 2^i = 2^{n-2} - 1$,

$$= 2^{n-1}a + 2^{n-2} - 1$$
$$= 2^{n-1}(a + \tfrac{1}{2} - \tfrac{1}{2^{n-1}})$$

and if n approaches infinity the limit makes the right side a constant leaving us with $2^{n-1} * c.$

So,

$$T(n) = \Theta(2^n)$$

Question #4

Fibonacci Sequence : 0,1,1,2,3,5,8,13...

$f(0) = 0$

$f(1) = 1$

$f(2) = F(0) + F(1) = 1$

$f(3) = F(1) + F(2) = 2$

$f(4) = F(2) + F(3) = 3$

$f(5) = F(3) + F(4) = 5$

Generalized:

$F(n) = F(n-2) + F(n-1)$

Given:

$F(n) = F_{n+1} - 1$

Prove by strong induction that the total number of additions performed is the given equation.

The total number of summation done is, the number of summation taken for each function (f(n-2), f(n-1)) plus the summation of the two functions to make f(n).

From this, we can derive the following equation for summation:

$F(n) = F(n-2) + F(n-1) + 1$

Basis:

From the sequence, we are given the following, where there is no summation done for.

$F(0) = 0$

$F(1) = 0$

$n = 2$

3 Iterative Substitution 2.75 / 3

- **- 0 pts** Correct
- **- 1 pts** The execution time of the second part is not correctly calculated.
- **- 1.25 pts** The execution time of the second part is not calculated and the total execution time is not discussed.
- ✓ **- 0.25 pts a tiny problem at the calculation of the second part of the execution time or first part.**
- **- 0.25 pts** problem in summation of the two terms related to (2^(n-1))
- **- 0.75 pts** Problem at reaching to T(1)
- **- 0.25 pts** problem at calculating the closed form of the second part.
- **- 0.5 pts** deformed closed form and the execution time is difficult to infer.
- **- 2 pts** incorrect response while part of the calculation has been correct.
- **- 2 pts** inability in reaching to a closed form.
- **- 3 pts** incorrect or no answer is provided.
- **- 0.5 pts** tiny problem at calculation of both part.

$$T(n) = 2^{n-1}T(1) + 2^{n-2} + 2^{n-3} + 2^{n-4} .... + 2^0$$
$$= 2^{n-1}a + 2^{n-2} + 2^{n-3} + 2^{n-4} .... + 2^0$$

If we rearrange for the numbers around we get:

$= 2^{n-1}a + 2^0 + 2^1 + 2^2 .... + 2^{n-2} \Rightarrow 2^{n-1}a$ + 1 + 2 + 4….+ 2^(n-2)

The higshlighted part can be simplified to $\sum_{i=0}^{n-2} 2^i = 2^{n-2} - 1$,

$= 2^{n-1}a + 2^{n-2} - 1$

$= 2^{n-1}(a + \frac{1}{2} - \frac{1}{2^{n-1}})$

and if n approaches infinity the limit makes the right side a constant leaving us with $2^{n-1} * c$.

So,

$T(n) = \Theta(2^n)$

Question #4

Fibonacci Sequence : 0,1,1,2,3,5,8,13...

$f(0) = 0$

$f(1) = 1$

$f(2) = F(0) + F(1) = 1$

$f(3) = F(1) + F(2) = 2$

$f(4) = F(2) + F(3) = 3$

$f(5) = F(3) + F(4) = 5$

Generalized:

$F(n) = F(n-2) + F(n-1)$

Given:

$F(n) = F_{n+1} - 1$

Prove by strong induction that the total number of additions performed is the given equation.

The total number of summation done is, the number of summation taken for each function (f(n-2), f(n-1)) plus the summation of the two functions to make f(n).

From this, we can derive the following equation for summation:

$F(n) = F(n-2) + F(n-1) + 1$

Basis:

From the sequence, we are given the following, where there is no summation done for.

$F(0) = 0$

$F(1) = 0$

$n = 2$

By using our derived formula the total number of summations done for f(2) is,
Left-hand side:
$F(2) = F(0) + F(1) + 1 = 0 + 0 + 1 = 1$ $\quad 1\ \textit{Summation for added the functions}$
And by using the given formula we get:
Right-hand side:
$F(2) = F_{2+1} - 1 = 2 - 1 = 1$
LH = RH

$n = 3$
By using our derived formula the total number of summations done for f(3) is,
Left-hand side:
$F(3) = F(1) + F(2) + 1 = 0 + 1 + 1 = 2$ $\quad 1\ \textit{Summation for added the functions}$
And by using the given formula we get:
Right-hand side:
$F(3) = F_{3+1} - 1 = 3 - 1 = 2$
LH = RH

$n = 4$
By using our derived formula the total number of summations done for f(2) is,
Left-hand side:
$F(4) = F(2) + F(3) + 1 = 1 + 2 + 1 = 4$ $\quad 1\ \textit{Summation for added the functions}$
And by using the given formula we get:
Right-hand side:
$F(4) = F_{4+1} - 1 = 5 - 1 = 4$
LH = RH
So the base case holds.

Induction Steps:
Inductive hypothesis, I.H: Assume the derived function is equal to the given function and it holds for all $f(n)$ *where* $n \leq k$.
Now prove that f(k+1) holds.

Right-hand side:
$F(k+1) = F_{k+2} - 1$

Left-hand side:
$F(k+1) = F(k+1-2) + F(k+1-1) + 1$
$F(k+1) = F(k-1) + F(k) + 1$
From our I.H we assume $F(k)$ is equal to the given function.
$F(k+1) = F(k-1) + F_{k+1} - 1 + 1$

Also since $F(k-1)$ falls into our I.H we can represent it in terms of the given function.
i.e = $F(k-1) = F_k - 1$ So,
$F(k+1) = F_k - 1 + F_{k+1} - 1 + 1$
By simplifying we get:
$F(k+1) = F_k - 1 + F_{k+1}$
$F(k+1) = F_{k+2} - 1$
LHS=RHS
Thus, the given function $F_{k+1} - 1$ computes the total number of additions for the nth Fibonacci number.

$Code\ for\ q\#1\ is\ implemented\ on\ an\ online\ version,\ forgot\ link\ source\ unkown.$

4 Induction 3 / 3

✓ **- 0 pts** Correct

- 0.75 pts base case is not discussed or is incorrect.

- 0.25 pts base case is partially correct.

- 0.25 pts The same notation is used for the number of summation and the fib number. It has reduced readability.(In the base part)

- 0.25 pts base case does not include T(1) but it is discussed before.

- 1 pts Induction case does not make sense or is incomplete.

- 0.5 pts The same notation is used for the number of summation and the fib number. It has reduced readability.

- 0.75 pts Base case is not discussed for T(0) and T(1).

- 3 pts No Answer is provided or the answer is incorrect.

- 1.5 pts The relation for the number of addition is not calculated correctly and as a result induction part is incorrect.

- 1.5 pts induction step is not provided.