PRT580 Assignment 2

cat.kutay

September 4 2020

1 Marking Rubric

Write your answers in overleaf and include the shareable overleaf link below **(replace the link below to this document with a link to your document)**, in case there is error in your formatting.

- You are required to submit your pdf. Export this from overleaf with correct formatting and symbol generation or your work will not be marked. For proofs it is recommended you provide a sample case first
- Half marks for working or explanation for a proof. The requirement is to include a brief and clear explanation of your work in your own words.
- Half marks for correct answer except where only explanation is required.
- Where a question has many parts, each part has equal value. Your result will be rounded up to nearest half mark

The original of this sheet is found https://www.overleaf.com/read/nrtwrqmgcptp
Please replace the link above with the link to your answer sheet in overleaf

2 Questions

1. (4 points) Proof of Correctness Week 4.8 and revised week 6 Workshop
   Given the following Algorithms derive their correctness (or not) by proof of the required invariant

   (a) Fibonnacci(n){
   $n \geq 0$ and $Fibonnacci(n) = ((1+\sqrt{5})/2)^n + (1-\sqrt{5})/2)^n)/\sqrt{5}$

```
    if n=0: return 0
    if n = 1: return 1

    return Fibonnacci(n-1) + Fibonnacci(n-2)
}
```

   (b) searchDictionary(word,node){

   $\forall c \in word : c \in Alphabet \cup \ '*'$.
   word $= c_1c_2..c_n \implies \forall\ c_i, c_{i+1}$ {if $c_1...c_i \in$ Dictionary and $c_{i+1} \notin$ Dictionary then word $\notin$ Dictionary}. Else word $\in$ Dictionary

```
word=word+"*"
for j in range (0, len(word))
    c=word[j:j+1]
    if c in node.children:
        index=0
        while c!=node.child[index]:
            index+=1
    else return false
    node=node_child[index]
return true
}
```

(c) public static int floor(int nums[], int key){
$\forall$ key $\in \mathbb{Z} \wedge$ i $\in$ range (0,nums.length) $\wedge$ nums[i] $\leq$ key and key $<$ nums[i+1], floor(nums,key)= i

```
    int listIndex=0;
    int r=nums.length-1;
    while(listIndex<=r){
        int approxIndex=listIndex+(r-listIndex)/2;
        if(nums[approxIndex]>=key)
            r=approxIndex-1;
        else
            listIndex=approxIndex+1;
    }
    return listIndex;
}
```

2. (3 points) Primes - Related to Proofs week 1 and 2.

   (a) Prove there are infinitely many primes

   (b) Using the Sieve of Eratosthenes, what is the worst case complexity for finding the prime factors of a number of magnitude N.

   (c) Prove that whenever a prime p does not divide the square of an integer, it also does not divide the original integer. i.e. p $\nmid x^2 \implies$ p $\nmid x$

3. (3 points) Partition - Coding exercise from Week 6
   Describe an efficient in-place algorithm called Partition-Even-Odd(A) that partitions an array A in even and odd numbers. The algorithm must terminate with A containing all its even elements preceding all its odd elements.
   For example, for input A = [7, 17, 74, 21, 7, 9, 26, 10],
   the result might be A = [74, 10, 26, 17, 7, 21, 9, 7].
   Partition-Even-Odd must be an in-place algorithm.
   What do you think this mean about your algorithm?

   (a) Write the pseudo-code for Partition-Even-Odd.

   (b) Characterize the complexity of Partition-Even-Odd. Briefly justify your answer.

   (c) Formalize the correctness of the partition problem as stated above, and prove that Partition-Even-Odd is correct using a loop-invariant (as repeated Week 6 workshop).

4. (1 point) Do these matrices represent equivalence relations. See Week 5 and repeated Workshop Week 6.

Explain your answer

(a) $\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$

(b) $\begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$

(c) $\begin{bmatrix} 0 & 1 & 1 \\ -1 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix}$

5. (3 points) Proof by Induction Week 4
Prove using weak or strong induction as needed. State which one you use

   (a) Prove that every positive integer n has a binary expansion. Namely, that there exists integers $c_i \in \{0, 1\}$ such that n=$\sum_{i=0}^{n} c_i 2^i$

   (b) Let $\{a_i\}$ be a sequence of sequence of real numbers such that $\forall$ i,j $a_{i+j} \leq a_i + a_j$
   Prove that $\sum_{i=0}^{n} a_i/i \geq a_n$

   (c) Prove by strong induction that if a country has n cities where any two cities are connected by a one-way road, there is a route that passes through every city.

6. (3 points) Closed forms - discussed since Week 4
Solve the following equation for the closed form of this expression:
$a_k = 4 \,.\, a_{k-1} + 5 \,.\, a_{k-2}$
That is find the nth term in terms of n, independent of previous terms except a_0 and a_1

7. (3 points) Equivalence Relations Week 5
Prove this statement from the definition of an equivalence relation:
Let A be a nonempty set and let R be an equivalence relation on the set A . Then,
$\forall$ a$\in$ A, a $\in$ [a].
$\forall$ a,b $\in$ A, a R b $\iff$ [a]=[b]
For each a,b $\in$ A, [a]=[b] $\vee$[a]$\cap$[b] = $\emptyset$
Also prove the collection C of all equivalence classes determined by R is a partition of the set A.

8. (4 points) Cryptography - extension from Week 6 Seminar
Consider a consequence of Euclid's Lemma:
$\forall$ a,b,c $\in Z$, gcd (a,b) =1 $\wedge$ a | bc $\rightarrow$ a | b
and Fermat's Little Theorem:
If p is any prime number and a is any integer such that p $\not| a \implies$ $a^{p=1} \equiv 1$ (mod p).
Show how these statements are used to show an RSA cypher will work. It is possible in RSA cryptography to encode a cypher with (pq, e) as public key, by using
C=M^e (mod pq)
and then decode the cypher with d=-e (mod (p-1)(q-1)), using M = C^d (mod pq)

9. (6 points) Algorithm - Based on work on graphs since week 1, mostly since week 3.
Consider the following Algorithms and explain why they are not sufficient to solve the problem
Simply a description or code for the standard algorithm will receive zero marks.

   (a) Find Shortest Path to Node
   To search for shortest path when the edges are different length you can use Breadth First or Depth First Search. The BFS and DFS are explained here

```
BFS
Traverse the graph breadth-wise as follows:
a. First move horizontally and visit all the nodes
of the current layer
b. Move to the next layer
DFS
Traverse the graph depth wise as follows:
a. Create a recursive function that takes the index of the node and a visited array.
b. Mark the current node as visited and print the node.
c. Traverse all the adjacent and unmarked nodes and call the recursive function
with the index of the adjacent node.\\
```

Provide examples of graphs where you will not find the shortest path by these methods. Consider other ways to approach this problem.

(b) Scheduling of jobs - similar to problem week 6 workshop

You are given jobs and need to schedule them. Each job is of one time unit duration but provide different profits for the company. Consider the sort as shown below Total Jobs TJ done and the Maximum Profit MP.

1 Sort the jobs based on decreasing order of profit.
2 Create two variables, TJ = 0, MP = 0.
3 Find the maximum deadline among all the jobs.
4 Initialise a set storing all the jobs in decreasing order of profit.
5 Iterate through the jobs and perform the following:
   i. If the set is empty or the deadline of the current job is less than the last element of the set, ignore the job.
   ii. Else, apply binary search dividing by deadline $\leq$ or$>$ i. Find the nearest Slot i, such that i $<$ deadline and add the profit.
   iii. Increment total jobs by 1 and remove the ith element from the set.
6 Return the maximum profit.

```
e.g. 'Item number', complete by, profit
arr =  [['a', 3, 35],  # Job Array
        ['b', 4, 30],
        ['c', 4, 25],
        ['d', 2, 20],
        ['e', 3, 15],
        ['f', 1, 12],
        ['g', 2, 5]]
Maximum profit sequence of jobs:
d c a b
Profit 110
```

Answer the following questions on this problem (Do not provide code)

(a) Do we need to do the initial sort of the jobs array? Why or why not.
(b) What will be the efficiency of this algorithm by calculating from the steps of the pseudo code? Explain how you calculated this.