12

1 Loop Invariant (12 Points)

Consider the following sorting algorithm known as *bubble sort.*

Algorithm 1 BubbleSort(integer array A[1..n])

for all $i = 1$ to n **do**
  for all $j = 2$ to $n - i + 1$ **do**
    if $A[j] < A[j-1]$ **then**
      Swap $A[j]$ and $A[j-1]$.

1. State a loop invariant for the outer for loop that is true in each iteration of the loop, and in the terminating iteration implies that the algorithm is correct. Briefly argue why the invariant indeed implies the correctness of the algorithm.

2. Prove that your loop invariant is true by induction.

Loop invariant:
Before start of every iteration i the values from $A[n-i+1 \ldots\ldots n]$ is sorted.
the loop terminates when $i = n$; so the
values $A[n-n+1 \ldots n]$ that mean $A[1 \ldots n]$
is sorted. For each step i increases to 1 and th
numbers are getting sorted from the backwards.
this n steps sort n numbers.

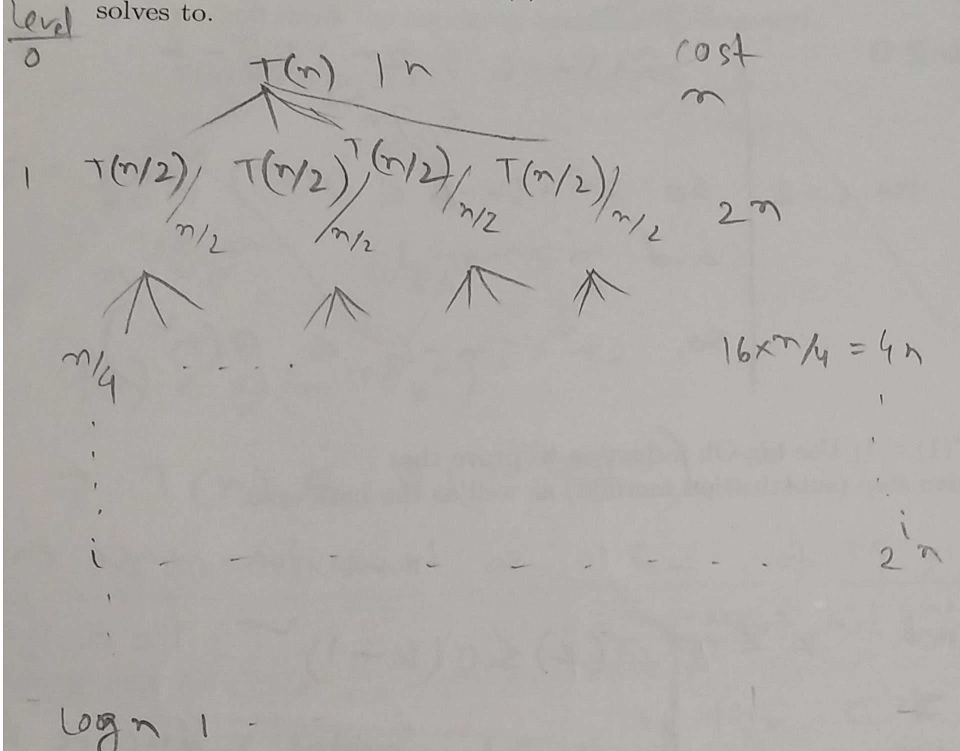
Proof by induction:
Base case: when $n=1$ then there is one element
trivially sorted.
Maintainance: Before the start of loop i ; all val
from $A[n-i+1 \ldots n]$ is sorted. Now, the in
loop compares the numbers from $j=1$ to $n-i$
and using swap it moves the largest ele
to $A[n-i]$ position. this largest element at A[n
$A[n-i+1 \ldots n]$ otherwise
smaller than all
be moved to that portion in earlier iteration

so now, $A[n-i \cdots n]$ is sorted.

Termination: when $i=n$, $A[n-n+1 \cdots n]$ that mean $A[1 \cdots n]$ is sorted. So the algo is correct.

2 Recursion Tree (12 Points)

Let $T(n) = 4T(n/2) + n$ with $T(1) = 1$. Use a recursion tree to generate a guess of what $T(n)$ solves to.



12

So $T(n) = \sum_{i=1}^{\log n} 2^i n = n \sum_{i=0}^{\log n} 2^i = n \cdot \frac{2^{\log n+1} - 1}{2-1} =$

12

3 Asymptotic Growth (12 Points)

1. Show that $2n^2 + 5n - 6 \in \Theta(n^2)$.

$2n^2+5n-6 \geq 2n^2$ if $5n-6 \geq 0$

$\rightarrow n \geq \frac{6}{5}$

so $2n^2+5n-6 \in \Omega(n^2)$ for $c=2$

and $n \geq n_0 = \frac{6}{5}$

again

$2n^2+5n-6 \leq 2n^2+5n^2-6$

$\leq 7n^2$

so $2n^2+5n-6 \in O(n^2)$ for $c=7$

and $n \geq n_0 = 1$

so, $2n^2+5n-6 \in \Theta(n^2)$

not sure which of is correct

2. Let $T(n) = 9T(n/3) + n$ with $T(1) = 1$. Use big-Oh induction to prove that $T(n) \in O(n^2)$. Prove the inductive step (substitution method) as well as the base case.

Base case: $T(1) = 1 \leq c \cdot 1^2$ for $c \geq 1$ so true for base case

Inductive step: for each $k < n$ $T(k) \leq c(k-1)^2$

now $T(n) = 9T(n/3) + n$

$\leq 9 \cdot c \cdot \left(\frac{n}{3}-1\right)^2 + n$

$= 9c\left(\frac{n^2}{9} - \frac{2n}{3} + 1\right) + n$

$= cn^2 - \cancel{6}nc + 9c + n$

$= c\,n^2 - 2nc + c - 4nc + 8c + n$

$= c(n-1)^2 - 4nc + 8c + n$

$\leq c(n-1)^2$ if $8c + n - 4nc \leq 0$

$\Rightarrow c(8-4n) \leq -n$

$\Rightarrow c \leq \frac{-n}{8-4n}$

$\Rightarrow c \leq \frac{n}{4n-8}$

$) = O(n^2)$ for $n \geq n_0 = 1$

$c \leq \frac{n}{4n-8}$

Alternate proof of inductive st

for $k < n$, $T(k) \leq ck^2$

now $T(n) = 9T(n/3) +$

$\leq 9c\left(\frac{n}{3}\right)^2$

$= cn^2 + n$

$\leq cn^2$

$= (c+$

so

9

4 Master Method (9 Points)

Solve the following recurrences using the master theorem. Justify your answers shortly (i.e. specify ϵ and check the regularity condition if necessary).

1. $T(n) = 16T(n/4) + n^2$

$a = 16$
$b = 4$

$n^{\log_b a} = n^{\log_4 16} = n^2$

$f(n) \in \Theta(n^{\log_b a})$

$\Rightarrow T(n) \in \Theta(n^2 \log n)$

2. $T(n) = T(n/4) + n$

$a = 1$
$b = 4$

$n^{\log_b a} = n^{\log_4 1} = n^0 = 1$

$f(n) \in \Omega(n^{\log_b a + \varepsilon})$

where $\varepsilon = 1$

Now $af(n/b) = 1 \cdot \frac{n}{4} \leq c \cdot n$

for $c \geq \frac{1}{4}$ so rule 3 applies

$\therefore T(n) \in \Theta(n)$

3. $T(n) = 9T(n/3) + n$

$a = 9$
$b = 3$

$\therefore n^{\log_b a} = n^{\log_3 9} = n^2$

$f(n) \in O(n^{\log_b a - \varepsilon})$

where $\varepsilon = 1$

So $T(n) \in \Theta(n^2)$

10

5 Counting Running Times (10 Points)

For both pieces of code, count the number of times "hello world" is printed.

```
for all i = 1; i < n; i+ = 3 do
    for all j = n; j > 1; j = j/4 do
        print("hello world")
```

$$\sum_{i=1}^{n/3} \sum_{j=1}^{\log_4 n} 1$$

Outer loop iterates n/3 times inner loop iterates $\log_4 n$ times.

So Total = $\frac{n}{3} \log_4 n$

```
for all i = 1; i < n; i + + do
    for all j = i; j < n; j + + do
        print("hello world")
```

outer loop runs n times and inner loop runs in geometric series [(n-i) times in each iteration)

$$\sum_{i=1}^{n-1} \sum_{j=i}^{n-1} 1 = 1 + 2 + 3 + \cdots n = \frac{n(n-1)}{2}$$

8

6 Randomized Analysis (8 Points)

Suppose someone offers to let you play a game. They will randomly (and independently) pick three numbers x_1, x_2, x_3 in the range 1-30. If x_1 is odd, you will be paid \$4 and if it is even you will pay \$2. If x_2 is in the range 1-10 you will be paid \$3, and if it is any other value you will pay \$1. If $x_3 = 1$ then you must pay \$15 (and will be paid nothing nothing otherwise). What is the expected value of this game from your perspective?

$$E[x] = \sum_{x \in X(S)} P(X=x) \cdot x$$

$$= \frac{1}{2} \times 4 + \frac{1}{2}(-2) + \frac{1}{3}(3) + \frac{2}{3}(-1) + \frac{1}{30}(-15)$$

$$= 2 - 1 + 1 - \frac{2}{3} - \frac{1}{2}$$

$$= \frac{12-4-3}{6} = \frac{5}{6}$$ ✓

So expectation positive. go for it.

16

7 Divide and Conquer (16 Points)

Let A be a *sorted* array of n distinct integers (positive or negative). Give a divide and conquer algorithm that finds an index i such that $A[i] = i$. If there is no such index, then your algorithm should return -1. Your algorithm should run in $O(\log n)$ time. Argue why your algorithm is correct. State the running time of your algorithm as a recurrence relation, and use the master method to show that the running time is indeed $O(\log n)$.

```
FIND-INDEX (A, p, q) {
    if (p≤q){
        mid = ⌊p+q/2⌋
        if (A[mid] == mid)
            return mid;
        else if (A[mid] > mid)
            return FIND-INDEX (A, p, mid-1)
        else if (A[mid] < mid)
            return FIND-INDEX (A, mid+1, q)
    } // end while
    return -1;
} // end code
```

This algorithm is co[rrect] because in each call taking the mid point array and if the valu[e] index is same then the index. if the value is than the index then all val[ues] that is greater than index look at the left half. And s[o] if the value is less than Then we look in the righ[t]

urrence relation:

$) = T(n/2) + 1$

$b = 1, \; n^{\log_b a} = n^{\log 1} = 1$

$) \in \Theta(n^{\log_b a})$

$(n) = \Theta(\log n)$

10

1 3 5 7 → lg n →
2 4 6 8 → lg n →

8 Order Statistics (16 Points)

Let A and B be two *sorted* arrays of n integers such that any integer appears in $A \cup B$ at most once. Note there are $2n$ distinct integers in $A \cup B$. Give an algorithm that given an integer x, returns the rank of x in $A \cup B$. If x is not in $A \cup B$, then return -1. Your algorithm should run as fast as possible. Argue why your algorithm is correct. What is its running time?

```
1. FIND-RANK(A, B, x)
2.     C = MERGE(A, B);          // O(n)
3.     for (i=1; i ≤ 2n; i++){   // O(2n)
4.     if (C[i] == x)
5.         return i;
6.     } // end for
7.     return -1;
8. }
```

```
MERGE (A, B){
  i=1 j=1, k=1
  while (i ≤ n and j ≤ n){
     if (A[i] ≤ B[i])
         C[k++] = A[i++]
     else C[k++] = B[j+
  }// end while
  while (i ≤ n)
     C[k++] = A[i+
  while (j ≤ n)
     C[k++] = B[
  return C
}
```

The algorithm is correct because merging two sorted array gives us a sorted array. Now we look the elements from 1 to 2n in the sorted array and will return the rank. The running time

is O(2n) or O(n)

Ok but can do in O(log n) time.