1 Loop Invariant (15 Points)

The following algorithm is the combine step for merge sort. That is, let A be a sorted array of $n/2$ integers, and let B be a sorted array of $n/2$ integers. The algorithm outputs C which is an array of n elements consisting of the elements in $A \cup B$ in sorted order. For simplicity, assume n is even and that $A[n/2+1] = B[n/2+1] = \infty$.

Algorithm 1 MergeSortCombine(integer array A[1..n/2], integer array B[1..n/2])

```
a = b = 1
for all i = 1 to n do
    if A[a] < B[b] then
        m = A[a]
        a = a + 1
    else
        m = B[b]
        b = b + 1
    C[i] = m
return C
```

1. State a loop invariant for the 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.

At the begining of iteration i, the first $i-1$ smallest elements are in their correct sorted position.

Algorithm terminates at $i = n+1$ which implies that all n elements are sorted.

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

ve; Since $A + B$ are sorted, the smallest element is e
in position $A[a]$ or $B[b]$. We set m to be the
of these, and necesarily all other elements remaining a
than m,

Trivially true for $i=1$,

2 Recursion Tree (15 Points)

Let $T(n) = 4T(n/2) + n^3$ with $T(1) = 1$. Use a recursion tree to generate a guess of what $T(n)$ solves to.

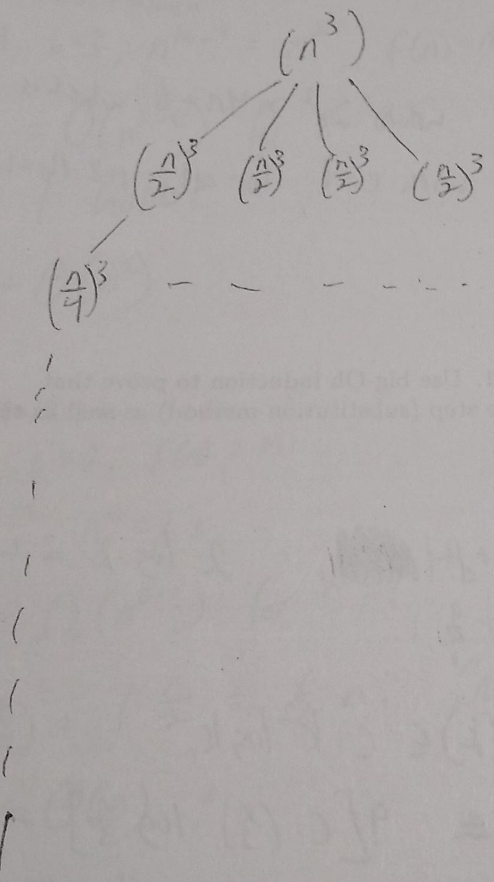
0 $\qquad n^3$

1 $\qquad 4\frac{n^3}{8} = \frac{n^3}{2}$

2 $\qquad 16\frac{n^3}{64} = \frac{n^3}{4}$

$i \qquad \frac{n^3}{2^i}$

$\log_2 n$



$$\sum_{i=0}^{\log_2 n} \frac{n^3}{2^i} = n^3 \sum_{i=0}^{\log_2 n} \left(\frac{1}{2}\right)^i < n^3 \sum_{i=0}^{\infty} \left(\frac{1}{2}\right)^i = n^3\left(\frac{1}{1-\frac{1}{2}}\right) = 2n^3$$

$$T(n) \in O(n^3)$$

3 Asymptotic Growth (12 Points)

1. Show that $3n^3 + 4n - 6 \in \Theta(n^3)$.

$3n^3 + 4n - 6 < 3n^3 + 4n^3 = 7n^3$. So $f(n) \le c \cdot g(n)$ for $c = 7$ and all $n \ge n_0 = 1$.

$n^3 \le 3n^3 - 2n^3 + 4n - 6$ when $2n^3 > 4n - 6$ which is true for all $n \ge 1$.

So $g(n) \le c \cdot f(n)$ with $c = 1$ for all $n \ge n_0 = 1$.

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

Base Case: $n = 2$

$T(2) \le 9T(1) + dn^2 = 9 + d4$, $2^2 \log 2 = 4$, so true for

$4c \ge 9 + d4 \Rightarrow c > d + \frac{9}{4}$.

Inductive Step: Assume $T(k) \le c \cdot k^2 \log k$,

$T(n) = 9T(\frac{n}{3}) + dn^2 \le 9[c \cdot (\frac{n}{3})^2 \cdot \log \frac{n}{3}] + dn^2$

$= cn^2[\log n - \log_3 3] + dn^2$

$= cn^2 \log n - cn^2 + dn^2$

$\le cn^2 \log n$ when $cn^2 > dn^2 \Rightarrow c > d$.

So we chose $c > d + \frac{9}{4}$ and it holds for all $n \ge n_0 = 2$.

4 Master Method (15 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. $T(n) = 9T(n/3) + n \log n$

$a=9,\ b=3,\ n^{\log_b a} = n^2,\ f(n) = n \log n$

$f(n) = O(n^{2-\varepsilon})$ for $\varepsilon = 0.5$.

Case 1 holds.

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

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

$a=1,\ b=2,\ f(n) = n$

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

$f(n) = \Omega(n^{0+\varepsilon})$ for $\varepsilon = 1$.

$a \cdot f(\frac{n}{b}) = 1 \cdot \frac{n}{2} = \frac{1}{2} \cdot n$, Case 3 holds.

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

3. $T(n) = 8T(n/2) + n^3$

$a=8,\ b=2,\ n^{\log_b a} = n^3,\ f(n) = n^3$

Case 2 holds.

$T(n) \in \Theta(n^3 \log n)$

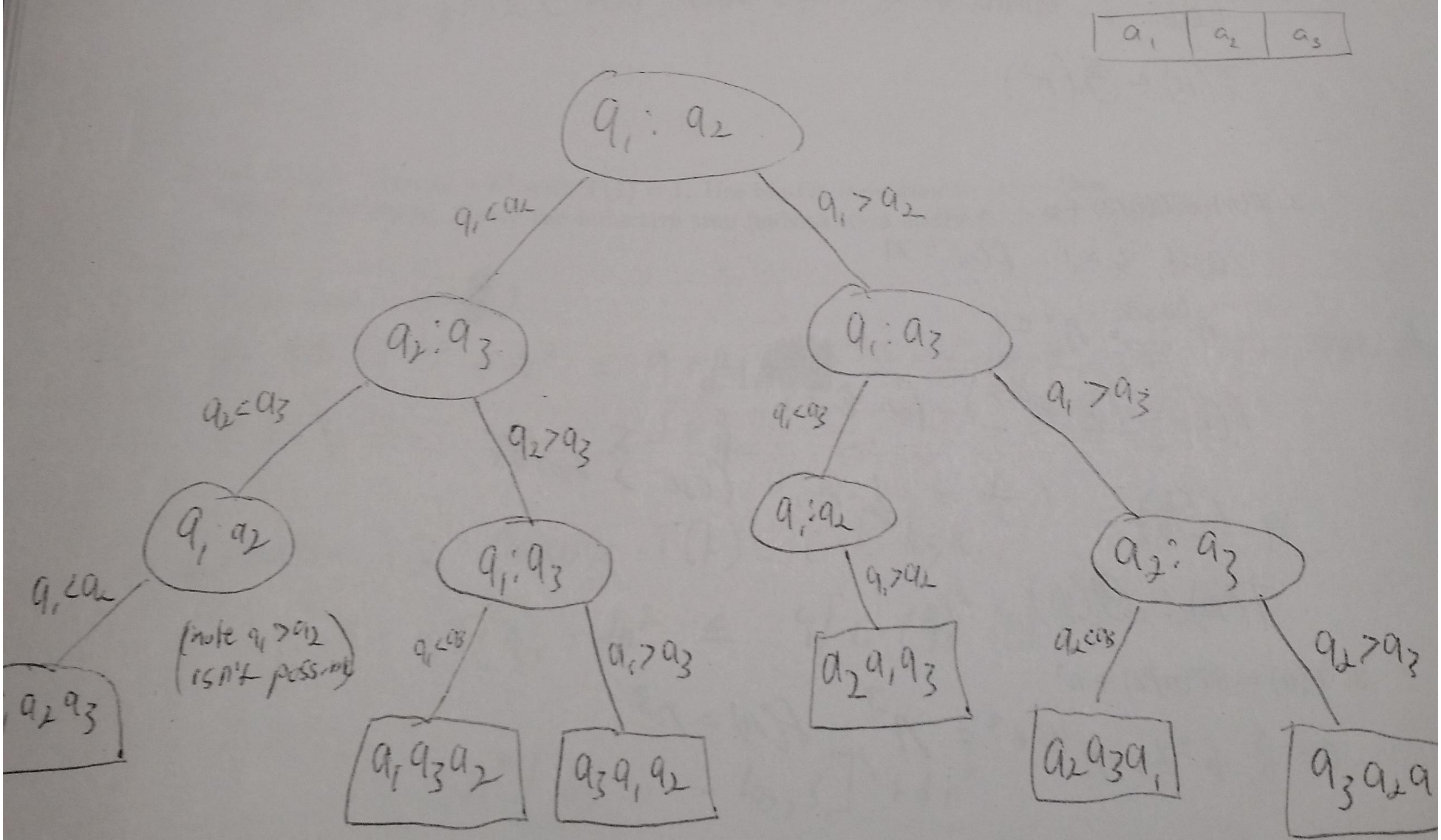
5 Decision Tree (15 Points)

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

Algorithm 2 BubbleSort(integer array A[1..n])

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

Draw the decision tree for BubbleSort on an input of size $n = 3$.



6 Divide and Conquer (25 Points)

Let A be an array of $n \geq 2$ integers (positive or negative). We are interested in computing indices l and r that minimizes the sum $\sum_{i=l}^{r} A[i]$. That is we want to compute the indices l and r such that the sum of integers between indicies l and r (inclusive) is minimized. Assume that A is indexed from 1 to n. For example, if $A = [1, 4, -2, -6, 7, -4, -5, 1, 3]$, then the optimal solution is $l = 3$ and $r = 7$ as the sum $\sum_{i=3}^{7} A[i] = -2 + -6 + 7 + -4 + -5 = -10$ and any other such sum will have a larger value. This problem can easily be solved in $O(n^2)$ time by checking all possibilities. This problem is about designing a divide and conquer algorithm for this problem with running time $o(n^2)$.

1. Suppose A is an array of size n where n is an even number, and let B denote the first $n/2$ elements of A and let C denote the last $n/2$ elements of A. Suppose we know an optimal solution for B and C. Give an $O(n)$ time algorithm that computes the optimal solution for A.

```
sum1 = 0; max1 = 0; index1;
sum2 = 0; max2 = 0; index2;
for(i = n/2, i>0, i--){
    sum1 += B[i]
    if(sum1 > max1){max1 = sum1; index1 = i}
}

for(i=1, i ≤ n/2, i++){
    sum2 += C[i]
    if(sum2 > max2){sum2 = max2; index2 = i;}
}
Return min(opt for B, opt for C, max1 + max2);
```

note this shou return the ind associated the max

2. What is the base case for this problem? (note two more parts to this question are on the next page)

When we have a single element.

Return the index.

3. Give the pseudocode for the algorithm. You can refer to your code above (i.e. you don't need to write it all over again, but do point out where it should go in the overall algorithm).

```
(int, int) minSubarray(A, p, q){
    if (p==q) return p;
    l1, r1 = minSubarray(A, p, ⌊(p+q)/2⌋);
    l2, r2 = minSubarray(A, ⌊(p+q)/2⌋+1, q);
    Let index1 + index2 be as defined in part 1.
    Return (l1, r1), (l2, r2), or (index1, index2) depending on
        which sum is smallest.
}
```

4. What is the runtime relation for this algorithm? Use the Master Theorem to determine what it evaluates to.

$T(n) = 2T(\frac{n}{2}) + \Theta(n)$

$a=2, \ b=2, \ n^{\log_b a} = n^1$

$f(n) = n^1$

Case 2

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