Homework Assignments #1

Due: 2020/04/08 12:00

Assessment policy:

- Give full points when correct, 1/n for solving each n subproblems. 0 for totally wrong or none, -1 for each error.
- There may be partial points for proofs if the direction is correct.

1. Correctness of bubble sort (11 pts)

Bubblesort is a popular, but inefficient, sorting algorithm. It works by repeatedly swapping adjacent elements that are out of order.

```
BUBBLESORT(A)
1  for i = 1 to A.length − 1
2      for j = A.length downto i + 1
3          if A[j] < A[j − 1]
4              exchange A[j] with A[j − 1]
```

To prove that `BUBBLESORT` is correct, we need to prove that it terminates and that $A'[1] \leq A'[2] \leq \cdots \leq A'[n]$, where $n = A.length$.

- (a) The loop invariant for the inner `for` loop (lines 2-4) is like follows. Prove that this loop invariant holds. *(3 pt)*

  loop invariant: At the start of each iteration the subarray $A[j..n]$ consists of the elements originally in $A[j..n]$ before entering the loop but possibly in a different order and the first element $A[j]$ is the smallest among them.

Initialization – when it starts, $j = A.length$, only $A[A.length]$ is (last element) in the subarray $A[j...n]$, which is the smallest one in the subarray.

Maintenance – during the loop, it compare $A[j]$ and $A[j-1]$, then, make $A[j-1]$ the smallest among them. After loop, the number of elements in subarray will increase by one and the subarray's first element is the smallest one.

Termination – when $i = j$, loop is over. In $A[j...n]$, $A[j]$ is the smallest one and $A[j...n]$'s elements consists of the elements originally in $A[j...n]$ before entering the loop.

- (b) Using the termination condition of the loop invariant proved in (a), state a loop invariant for the outer `for` loop (lines 1–4). Prove that this loop invariant holds. *(6 pt)*

loop invariant: At the start of each iteration the subarray $A[1 \cdots i-1]$ consist of the smallest elements of ~~the~~ $A[1 \cdots n]$ in sorted order. $A[i \cdots n]$ consists of the $n-i+1$ elements in $A[1 \cdots n]$, which is remaining by $A[1 \cdots i-1]$.

Initialization – when it starts, the subarray $A[1 \cdots i-1]$ is empty and this is the smallest elements of ~~the~~ the subarray.

Maintenance. – after the loop 2–4, by the condition of (a), $A[i]$ become the smallest one of the $A[i \cdots n]$. And then, in the next execution of loop 1, $A[1 \cdots i-1]$ is the smallest elements of $A[1 \cdots n]$ in sorted ~~array~~ ordered.

Termination – when $i = A.length$, loop is over, then, $A[1 \cdots n]$ will be sorted array.

- (c) What is the worst-case running time of bubblesort? How does it compare to the running time of insertion sort? *(2 pt)*

```
1 for - n
2 { for   - n-i
3
4 }
```

worst-case is $n(n-i) = n^2 - ni$

$= \theta(n^2)$

It is same with the Insertion Sort.

2. [Asymptotic growth rates] (10 pts)

List the sixteen functions below from asymptotically lowest order to highest order. If there are functions of the same order, indicate which.

n	2^n	$n \lg n$	$\ln n$
$n - n^3 + 7n^5$	$\lg n$	$\sqrt{n}$	e^n
$n^2 + \lg n$	n^2	2^{n-1}	$\lg \lg n$
n^3	$(\lg n)^2$	$n!$	$n^{1+\epsilon}$, where $0 < \epsilon < 1$

You can denote it as a subset notation, for instance like the follwing:

$O(n) \subset O(n^2) = O(3n^2 + 1) \subset O(n^3) \subset \cdots$

$o(\log\log n) \subset o(\log n) = o(\ln n) \subset o(\sqrt{n}) \subset o((\log n)^2) \subset o(n) \subset o(n \log n)$

$\subset o(n^{1+\epsilon} \text{ where } 0<\epsilon<1) \subset o(n^2) = o(n^2 + \log n) \subset o(n^3)$

$\subset o(n - n^3 + 7n^5) \subset o(2^n) = o(2^{n-1}) \subset o(e^n) \subset o(n!)$

-1.5

A [1 2 .... n]

3. Brute-force maximum subarray problem (10 pts)

The maximum subarray problem takes $\Theta(n^2)$ time to compare all possible pairs.

- **input**: Sequence of n numbers A
- **output**: A subsequence A' whose sum is the maximum among all subsequences of A

Write a pseudocode for the brute-force method `BRUTE-FORCE-FIND-MAXIMUM-SUBARRAY(A)` of solving the maximum-subarray problem. Your procedure should run in $\Theta(n^2)$ time.

```
Maximum subarray (A)
for i=1 to A.length
    sum is 0.
    for j=i to A.length
        sum = sum + A[j]
        if sum > max
            max is sum
            Maximum subarray is A[i...j]
```

max is not initialized -1

maximum subarray is not resolved here. -1

4. Apply substitution method (6 pts)

Prove the following solutions of recurrences with substitution method *(3 pts each)*

- (a) Show that the solution of $T(n) = T(n-1) + n$ is $O(n^2)$

$T(n) = T(n-1) + n$ $\quad O(n^2) = 0 \le f(n) \le c \cdot n^2, n \ge n_0$

$\le c \cdot (n-1)^2 + n$

$= c \cdot n^2 - 2cn + c + n$

$\le c \cdot n^2$

So $cn^2 - 2cn + c + n \le cn^2$

$(1-2c)n + c \le 0$

where $\therefore n \ge \frac{c}{2c-1}$, $n_0 = \frac{c}{2c-1}$

- (b) Show that the solution of $T(n) = T(\lceil n/2 \rceil) + 1$ is $O(\lg n)$

$T(n) = T(\lceil n/2 \rceil) + 1$
$\leq c \cdot \log^{n/2} + 1$
$= c \cdot \log^n - c + 1$
$\leq c \cdot \log^n$

$O(\log n) = 0 \leq f(n) \leq c \cdot \log n, \; n \geq n_0.$

So $c \cdot \log^n - c + 1 \leq c \cdot \log^n$

where $\therefore 1 \leq c$

* upper 일때 조건이 부합하는 상한선 제시

5. Apply recursion method (15 pts)

Use a recursion tree to determine a good asymptotic upper bound for the following recurrences. Use the substitution methods to verify your answer. *(3 pts each)*

- (a) $T(n) = 3T(\lfloor n/2 \rfloor) + n$

$\log n = \log_{10} n$, $\lg n = \log_2 n$

height $= \log^n + 1$

Cost at each level $= (3/2)^i n$

# of nodes at depth $i = 3^i$

$T(n)$
$T(n/2) \; T(n/2) \; T(n/2)$
$T(n/4) \; T(n/4) \; T(n/4)$
$T(1) \; T(1) \; \cdots \; T(1)$

all cost at specific level $= n + \left(\frac{3}{2}\right)n + \left(\frac{3}{2}\right)^2 n + \cdots + \left(\frac{3}{2}\right)^{\log n - 1} n + \theta(n^{\log^3})$

$= n \cdot \dfrac{\frac{3}{2}^{\log^n} - 1}{\frac{3}{2} - 1} + \theta(n^{\log^3})$

$= 2(n^{\log^3} - n) + \theta(n^{\log^3})$

$\Rightarrow O(n^{\log^3})$

$T(n) \leq c \cdot n^{\log^3} - dn$

$= 3\left[c\left(\frac{n}{2}\right)^{\log^3} - d \cdot \frac{n}{2}\right] + n = c \cdot n^{\log^3} + (1 - \frac{3}{2}d)n$

$(1 - \frac{3}{2}d)n \leq -dn$

where $\therefore d \geq 2$

- (b) $T(n) = T(n/2) + n^2$

$T(n)$
$T(\frac{n}{2})$
$T(\frac{n}{2^2})$
$\vdots$
$T(1)$

height $= \log n + 1$

Cost at each level $= (\frac{1}{4})^i \cdot n^2$

# of nodes at depth $i = 1$

all cost at specific level $= n^2 + (\frac{1}{4})n^2 + (\frac{1}{4})^2 n^2 + \cdots + (\frac{1}{4})^{\log n - 1} n^2 + \theta(1)$

$= n^2 \cdot \dfrac{1 - (\frac{1}{4})^{\log n}}{1 - \frac{1}{4}} + \theta(1)$

$< n^2 \cdot \dfrac{1}{1 - \frac{1}{4}} + \theta(1)$

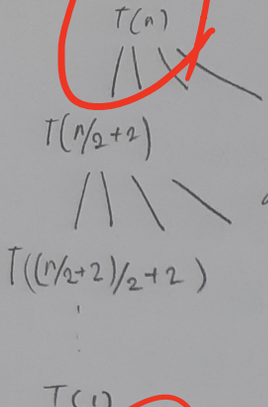
$= \theta(n^2)$

$T(n) \leq cn^2$
$\leq c\left(\frac{n}{2}\right)^2 + n^2$
$= cn^2/4 + n^2$
$= (c/4 + 1)n^2$
$\leq cn^2$

where $\therefore \frac{4}{3} \leq c$

$\frac{c}{4} + 1 \leq c$

- (c) $T(n) = 4T(n/2+2) + n$



height = $\log n + 1$

# of nodes at depth i = 4^i

cost at each level = $n \times 2^i + 2^{2i+2} - 2^{i+2}$

all cost at specific level = $\sum_{i=0}^{\log n - 1} \left[n \times 2^i + 2^{2i+2} - 2^{i+2} \right] + \theta(n^2)$

$= n(2^{\log n} - 1) + \frac{4}{3}(4^{\log n} - 1) - 4(2^{\log n} - 1) + \theta(n^2)$

$= \theta(n^2)$

$c > 0$

$T(n) \le cn^2 - dn$

$T(n) \le 4c(n/2+2)^2 + 4d(n/2+2) + n$

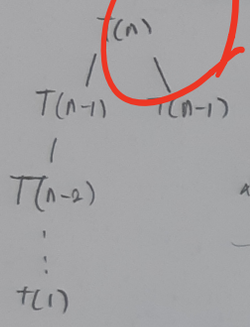
$= cn^2 + 8cn + 16c - 2dn - 8d + n$

$= cn^2 + (8c - 2d + 1)n + 16c - 8d$

$\le cn^2$

$\therefore 8c - 2d + 1 < 0$

- (d) $T(n) = 2T(n-1) + 1$



height = n

# of nodes at depth i = 2^i

Cost at each level = 2^i

all cost at specific level = $\sum_{i=0}^{n-1} 2^i = 2^n - 1 = \theta(2^n)$

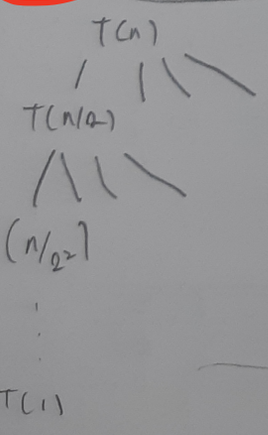
$T(n) \le c \cdot 2^n + n$

$= 2c2^{n-1} + n$

$= c \cdot 2^n + n$

6. Draw recursion method (4 pts)

Draw the recursion tree for $T(n) = 4T(\lfloor n/2 \rfloor) + cn$, where c is a constant, and provide a tight asymptotic bound by the substitution method.



height = $\log n + 1$

# of nodes at depth i = 4^i

Cost at each level = $c \cdot n \cdot 2^i$

all cost at specific level = $\sum_{i=0}^{\log n - 1} cn \cdot 2^i + \theta(n^2)$

$= cn(2^{\log n} - 1) + \theta(n^2)$

$= \theta(n^2)$

- $O(n^2)$ $T(n) \le dn^2 - cn$

$= 4 \cdot d\frac{n^2}{4} - 4c\frac{n}{2} + cn = dn^2 - cn$

- $\Omega(n^2)$ $T(n) \ge dn^2 - cn$

$= dn^2 - cn$