بسمه تعالی

دانشگاه صنعتی امیرکبیر **دانشکده مهندسی کامپیوتر**

پاسخنامه تمرین سری اول درس طراحی الگوریتم‌ها

1. Let $T(n)$ denote the running time for insertion sort called on an array of size n. We can express $T(n)$ recursively as

$$T(n) = \begin{cases} \Theta(1) & \text{if } n \leq c \\ T(n-1) + I(n) & \text{otherwise} \end{cases}$$

where $I(n)$ denotes the amount of time it takes to insert $A[n]$ into the sorted array $A[1..n-1]$. Since we may have to shift as many as $n-1$ elements once we find the correct place to insert $A[n]$, we have $I(n) = \theta(n)$.

2. We can see that the while loop gets run at most $O(n)$ times, as the quantity $j-i$ starts at $n-1$ and decreases at each step. Also, since the body only consists of a constant amount of work, all of lines 2-15 takes only $O(n)$ time. So, the runtime is dominated by the time to perform the sort, which is $\Theta(n \lg(n))$. We will prove correctness by a mutual induction. Let $m_{i,j}$ be the proposition $A[i] + A[j] < S$ and $M_{i,j}$ be the proposition $A[i] + A[j] > S$. Note that because the array is sorted, $m_{i,j} \Rightarrow \forall k < j, m_{i,k}$, and $M_{i,j} \Rightarrow \forall k > i, M_{k,j}$.

Our program will obviously only output true in the case that there is a valid i and j. Now, suppose that our program output false, even though there were some i, j that was not considered for which $A[i] + A[j] = S$. If we have $i > j$, then swap the two, and the sum will not change, so, assume $i \leq j$. we now have two cases:

Case 1 $\exists k, (i, k)$ was considered and $j < k$. In this case, we take the smallest such k. The fact that this is nonzero meant that immediately after considering it, we considered (i+1,k) which means $m_{i,k}$ this means $m_{i,j}$

Case 2 $\exists k, (k, j)$ was considered and $k < i$. In this case, we take the largest such k. The fact that this is nonzero meant that immediately after considering it, we considered (k,j-1) which means $M_{k,j}$ this means $M_{i,j}$

Note that one of these two cases must be true since the set of considered points separates $\{(m, m') : m \leq m' < n\}$ into at most two regions. If you are in the region that contains $(1, 1)$(if nonempty) then you are in Case 1. If you are in the region that contains (n, n) (if non-empty) then you are in case 2.

```
1:  Use Merge Sort to sort the array A in time Θ(n lg(n))
2:  i = 1
3:  j = n
4:  while i < j do
5:      if A[j] + A[j] = S then
6:          return true
7:      end if
8:      if A[i] + A[j] < S then
9:          i = i + 1
10:     end if
11:     if A[i] + A[j] > S then
12:         j = j − 1
13:     end if
14: end while
15: return false
```

3. a. False. Counterexample: $n = O(n^2)$ but $n^2 \neq O(n)$.

b. False. Counterexample: $n + n^2 \neq \Theta(n)$.

c. True. Since $f(n) = O(g(n))$ there exist c and n_0 such that $n \geq n_0$ implies $f(n) \leq cg(n)$ and $f(n) \geq 1$. This means that $\log(f(n)) \leq \log(cg(n)) = \log(c) + \log(g(n))$. Note that the inequality is preserved after taking logs because $f(n) \geq 1$. Now we need to find d such that $\log(f(n)) \leq d\log(g(n))$. It will suffice to make $\log(c) + \log(g(n)) \leq d\log(g(n))$, which is achieved by taking $d = \log(c) + 1$, since $\log(g(n)) \geq 1$.

d. False. Counterexample: $2n = O(n)$ but $2^{2n} \neq 2^n$ as shown in exercise 3.1-4.

e. False. Counterexample: Let $f(n) = \frac{1}{n}$. Suppose that c is such that $\frac{1}{n} \leq c\frac{1}{n^2}$ for $n \geq n_0$. Choose k such that $kc \geq n_0$ and $k > 1$. Then this implies $\frac{1}{kc} \leq \frac{c}{k^2c^2} = \frac{1}{k^2c}$, a contradiction.

f. True. Since $f(n) = O(g(n))$ there exist c and n_0 such that $n \geq n_0$ implies $f(n) \leq cg(n)$. Thus $g(n) \geq \frac{1}{c}f(n)$, so $g(n) = \Omega(f(n))$.

g. False. Counterexample: Let $f(n) = 2^{2n}$. By exercise 3.1-4, $2^{2n} \neq O(2^n)$.

h. True. Let g be any function such that $g(n) = o(f(n))$. Since g is asymptotically positive let n_0 be such that $n \geq n_0$ implies $g(n) \geq 0$. Then $f(n) + g(n) \geq f(n)$ so $f(n) + o(f(n)) = \Omega(f(n))$. Next, choose n_1 such that $n \geq n_1$ implies $g(n) \leq f(n)$. Then $f(n) + g(n) \leq f(n) + f(n) = 2f(n)$ so $f(n) + o(f(n)) = O(f(n))$. By Theorem 3.1, this implies $f(n) + o(f(n)) = \Theta(f(n))$.

4. Assume $T(n) \leq c(n-a)\lg(n-a)$

$$
\begin{aligned}
T(n) &\leq 2c(\lfloor n/2 \rfloor + 17 - a)\lg(\lfloor n/2 \rfloor + 17 - a) + n \\
&\leq 2c(n/2 + 1 + 17 - a)\lg(n/2 + 1 + 17 - a) + n \\
&\leq c(n + 36 - 2a)\lg(\frac{n + 36 - 2a}{2}) + n \\
&\leq c(n + 36 - 2a)\lg(n + 36 - 2a) - c(n + 36 - 2a) + n \\
&\leq c(n + 36 - 2a)\lg(n + 36 - 2a) \quad if \quad c > 1 \\
&\leq c(n - a)\lg(n - a) \quad if \quad a \geq 36
\end{aligned}
$$

5. Determine an upper bound on $T(n) = 3T(\lfloor n/2 \rfloor) + n$ using a recursion tree. We have that each node of depth i is bounded by $n/2^i$ and therefore the contribution of each level is at most $(3/2)^i n$. The last level of depth $\lg n$ contributes $\Theta(3^{\lg n}) = \Theta(n^{\lg 3})$. Summing up we obtain:

$$
\begin{aligned}
T(n) &= 3T(\lfloor n/2 \rfloor) + n \\
&\leqslant n + (3/2)n + (3/2)^2 n + \cdots + (3/2)^{\lg n - 1} n + \Theta(n^{\lg 3}) \\
&= n \sum_{i=0}^{\lg n - 1} (3/2)^i + \Theta(n^{\lg 3}) \\
&= n \cdot \frac{(3/2)^{\lg n} - 1}{(3/2) - 1} + \Theta(n^{\lg 3}) \\
&= 2(n(3/2)^{\lg n} - n) + \Theta(n^{\lg 3}) \\
&= 2n \frac{3^{\lg n}}{2^{\lg n}} - 2n + \Theta(n^{\lg 3}) \\
&= 2 \cdot 3^{\lg n} - 2n + \Theta(n^{\lg 3}) \\
&= 2n^{\lg 3} - 2n + \Theta(n^{\lg 3}) \\
&= \Theta(n^{\lg 3})
\end{aligned}
$$

We can prove this by substitution by assumming that $T(\lfloor n/2 \rfloor) \leqslant c\lfloor n/2 \rfloor^{\lg 3} - c\lfloor n/2 \rfloor$. We obtain:

$$
\begin{aligned}
T(n) &= 3T(\lfloor n/2 \rfloor) + n \\
&\leqslant 3c\lfloor n/2 \rfloor^{\lg 3} - c\lfloor n/2 \rfloor + n \\
&\leqslant \frac{3cn^{\lg 3}}{2^{\lg 3}} - \frac{cn}{2} + n \\
&\leqslant cn^{\lg 3} - \frac{cn}{2} + n \\
&\leqslant cn^{\lg 3}
\end{aligned}
$$

Where the last inequality holds for $c \geqslant 2$.

6. Use the master method to find bounds for the following recursions. Note that $a = 4, b = 4$ and $n^{\log_2 4} = n^2$

- $T(n) = 4T(n/2) + n$. Since $n = O(n^{2-\epsilon})$ case 1 applies and we get $T(n) = \Theta(n^2)$.
- $T(n) = 4T(n/2) + n^2$. Since $n^2 = \Theta(n^2)$ we have $T(n) = \Theta(n^2 \lg n)$.
- $T(n) = 4T(n/2) + n^3$. Since $n^3 = \Omega(n^{2+\epsilon})$ and $4(n/2)^3 = 1/2n^3 \leqslant cn^3$ for some $c < 1$ we have that $T(n) = \Theta(n^3)$.