Due: Sep 29, 2016, in class.

Note: for simplicity, the base of “log” is always 2.

1. (10 × 2 = 20 points) Proof:

   (1) According to the definition of notation $\theta$, we need to find 2 positive constants $c_1$ and $c_2$ and another positive constant $n_0$ such that the inequality:

   $$c_1 n \log n \leq \sum_{i=1}^{n} \log i \leq c_2 n \log n$$

   hold for all $n \geq n_0$.

   First we prove left part of the above inequality.

   $$\sum_{i=1}^{n} \log i = \log 1 + \log 2 + \cdots + \log n$$

   $$= \log 1 + \cdots + \log \frac{n}{2} + \log \left(\frac{n}{2} + 1\right) + \cdots + \log n$$

   $$\geq \log \left(\frac{n}{2} + 1\right) + \cdots + \log n$$

   $$\geq \log \frac{n}{2} + \cdots + \log \frac{n}{2}$$

   $$= \frac{n}{2} \log \frac{n}{2}$$

   Hence we can choose $0 < c_1 \leq \frac{1}{2}$ and $n_0 = 1$ to make the left part hold.

   Then we prove the right part of above inequality. Notice that $i \leq n$, we have $\log i \leq \log n$ and hence

   $$\sum_{i=1}^{n} \log i \leq \sum_{i=1}^{n} \log n = n \log n.$$  This holds for any $c_2 \geq 1$ and $n_0 \geq 1$.

   Combine the above results we can see that $\sum_{i=1}^{n} \log i = \theta(n \log n)$.

   (2) Same as (1), we need to find 2 positive constants $c_1$ and $c_2$ and another positive constant $n_0$ such that the inequality:

   $$c_1 n^2 \leq \sum_{i=1}^{n} i \leq c_2 n^2$$

   hold for all $n \geq n_0$. 

   1
From the fact that $\sum_{i=1}^{n} i = \frac{n(n+1)}{2} = \frac{n^2+n}{2}$ and let $c_1 = \frac{1}{2}$, the left part of inequality becomes $n \geq 1$. Hence we find such $c_1 (= \frac{1}{2})$ and $n_0 (= 2)$ such that the left part of inequality holds.

For the right part, let $c_2 = 1$, the right part of inequality becomes $n \geq 1$. Hence we find such $c_2 (= 1)$ and $n_0 (= 1)$ such that the right part of inequality holds.

Combine the above results we have $\sum_{i=1}^{n} i = \Theta(n^2)$.

2. (10 $\times$ 2 = 20 points) Compare the growth rates of $T_1(n)$ and $T_2(n)$:

1. $T_1(n) = n^2$ and $T_2(n) = 6n \log n - \frac{1}{2}n$;

Solution:

Using limit test we have:

$$\lim_{n \to \infty} \frac{T_1(n)}{T_2(n)} = \lim_{n \to \infty} \frac{n^2}{6n \log n - \frac{1}{2}n} = \lim_{n \to \infty} \frac{n}{6 \log n - \frac{1}{2}}.$$ And both $n$ and $6 \log n - \frac{1}{2}$ tend to $\infty$ when $n \to \infty$. Now using L’hospital rule we have:

$$\lim_{n \to \infty} \frac{T_1(n)}{T_2(n)} = \lim_{n \to \infty} \frac{1}{6 \log n} = \infty.$$

According to limit test, we have $T_1(n) = \omega(T_2(n))$.

2. $T_1(n) = n^3$ and $T_2(n) = 1.5n^3 + (-1)^[n]n^3$.

Solution:

Using limit test we have:

$$\lim_{n \to \infty} \frac{T_1(n)}{T_2(n)} = \lim_{n \to \infty} \frac{n^3}{1.5n^3 + (-1)^[n]n^3} = \lim_{n \to \infty} \frac{1}{1.5 + (-1)^[n]}.$$ This shows that limit does not exists, hence limit test fails.

But we claim that $T_1(n) = \Theta(T_2(n))$ (or equivalently, $T_2(n) = \Theta(T_1(n))$). This is equivalent to find 2 positive constants $c_1$ and $c_2$ and another positive constant $n_0$ such that the inequality:

$$c_1n^3 \leq 1.5n^3 + (-1)^[n]n^3 \leq c_2n^3$$

hold for all $n \geq n_0$.

We can set $c_1 = 0.5$ and $c_2 = 2.5$ and $n_0 = 1$. Hence $T_1(n) = \Theta(T_2(n))$.

3. (20 points) Given an array $A = [6, 3, 5, 7, 0, 2]$, use insertion sort algorithm to sort $A$ in an increasing order. The algorithm scans $A$ from left to right by $size(A) = 6$ steps, so please list the changes of $A$ during these 6 steps.

Solution:
4. (20 points) Array \( A = [a_1, a_2, \cdots, a_n] \) is unsorted, design an \( O(n \log n) \)-time algorithm to report the number of inversions in \( A \). An inversion is a pair of numbers \( a_i \) and \( a_j \) such that \( i < j \) but \( a_i \geq a_j \) (hint: you may use divide-and-conquer idea).

Solution:

Using the ideas from merge-sort:

\[
\textbf{Algorithm 1} \ \text{Count-inversion (List)}
\]

\[\begin{array}{l}
\text{if length(List) == 1: then} \\
\quad \text{return (0, List)} \\
\text{else} \\
\quad \text{mid = length(List)/2} \\
\quad \text{(inv1, L1) = Count-inversion(List[0, mid])} \\
\quad \text{(inv2, L2) = Count-inversion(List[mid+1, len(List)-1])} \\
\quad \text{(inv, L) = merge(L1, L2)} \\
\end{array}\]

\[\begin{array}{c}
\text{end if} \\
\quad \boxed{\text{return (inv1+inv2+inv, L)}} \\
\end{array}\]
Algorithm 2 merge(List1, List2)

count = 0  \triangleright \text{Count the number of inversion}
k, i, j = 0  \triangleright \text{pointers point at the current position (new list, left list, right list)}
Listnew = zero(length(List1)+ length(List2))  \triangleright \text{An new list to store the sorted left and right list, initial with all zeros}

while k ≤ length(Listnew): do
    if List1[i] < List2[j]: then  \triangleright \text{Current element in left list is smaller}
        Listnew[k] = List1[i]
        i = i + 1
    else  \triangleright \text{Current element in right list is smaller}
        Listnew[k] = List2[j]
        count = count + (length(List1) - i)  \triangleright \text{Current element in right list is smaller all elements left in left list, which are inversions}
        j = j + 1
    end if
    if i == length(List1): then
        append the rest of List2 to Listnew
        break
    else  \triangleright \text{List2 left some elements unsorted}
        append the rest of List1 to Listnew
        break
    end if
    k = k + 1
end while
return (inv1+inv2+inv, L)
5. (20 points) Given an array \( A = [3, 10, 2, 7, 12, 4, 20] \), build the MaxHeap for \( A \). Please list the updated \( A \) after calling \textit{MaxHeapify} each time.

Solution:

\[
\begin{align*}
&[3, 10, 2, 7, 12, 4, 20] \\
&[3, 10, 20, 7, 12, 4, 2] \\
&[3, 12, 20, 7, 10, 4, 2] \\
&[20, 12, 3, 7, 10, 4, 2] \\
&[20, 12, 4, 7, 10, 3, 2]
\end{align*}
\]