ECS 122A – Algorithm & Analysis
Homework 01

Question 1: Inductive Proof (20 points)

1. (5 points) Find the closed form of $\Sigma_{i=1}^{n} 2^i$.

   Answer: $\Sigma_{i=1}^{n} 2^i = 2^{n+1} - 2$

2. (15 points) Prove your closed-form formula using induction. (Show your work.)

   Answer:
   Base case: When $n = 1$,
   $$\Sigma_{i=1}^{n} 2^i = 2^1 = 2$$
   $$2^{n+1} - 2 = 2^2 - 2 = 2$$

   Inductive hypothesis: Let $k \geq 1$. Assume $\Sigma_{i=1}^{k} 2^i = 2^{k+1} - 2$.

   Induction step: Need to show $\Sigma_{i=1}^{k+1} 2^i = 2^{k+2} - 2$
   $$\begin{aligned} \Sigma_{i=1}^{k+1} 2^i &= \Sigma_{i=1}^{k} 2^i + 2^{k+1} \\ &= 2^{k+1} - 2 + 2^{k+1} \\ &= 2^{k+2} - 2 \end{aligned}$$

Question 2: Basic Code Analysis (15 points)

What is the asymptotic upper bound (tightest Big-O) of the following algorithm, assume n is the input and is a positive number? (Briefly explain your solution.)

```
i = n
while (i > 1) {
    j = i
    while (j < n) {
        k = 1
        while (k < n) {
            k = k * 2
        }
        j = j + 1
    }
    i = i / 2
}
```

Answer:

i	number of iterations of the second loop
n	$n - n$
$\frac{n}{2}$	$n - \frac{n}{2}$
$\frac{n}{4}$	$n - \frac{n}{4}$

Total number of iterations of the first two loops:

$$
\begin{aligned}
&(n - n) + (n - \frac{n}{2}) + (n - \frac{n}{4}) + ... + 1 \\
&= (n - \frac{n}{2^0}) + (n - \frac{n}{2^1}) + (n - \frac{n}{2^2}) + ... + (n - \frac{n}{2^{logn}}) \\
&= (n - \frac{n}{2^0}) + (n - \frac{n}{2^1}) + (n - \frac{n}{2^2}) + ... + (n - \frac{n}{2^{logn}}) \\
&= nlogn - nlogn\Sigma_{i=0}^{logn} \frac{1}{2^i} \\
&= O(nlogn)
\end{aligned}
$$

The third loop iterates n times every time.
The total runtime is $O(n(logn)^2) = O(nlog^2n)$.

Question 3: Proving Big-O By Definiton (15 points)

Prove that $T(n) = 2n^4 + 5n^3 + 3n^3logn + 2n + 5$ is $O(n^4)$ without using the Limit Lemma theorem. (Show your work.)

Answer:

$$
\begin{aligned}
&2n^4 \leq 2n^4 && \forall n.n \geq 0 \\
&5n^3 \leq 5n^4 && \forall n.n \geq 1 \\
&3n^3logn \leq 3n^4 && \forall n.n \geq 1 \\
&2n \leq 2n^4 && \forall n.n \geq 1 \\
&5 \leq 5n^4 && \forall n.n \geq 1
\end{aligned}
$$

Choose $c = 2 + 5 + 3 + 2 + 5 = 17$, $n_0 = 1$.
$T(n) = 2n^4 + 5n^3 + 3n^3logn + 2n + 5 \leq 17n^4$ for all $n \geq 1$.

Question 4: Limit Lemma Theorem (10 points)

Prove that $T(n) = 5n^6 + n^2 + 3$ is $O(logn + n^6 + n)$ using the Limit Lemma Theorem. (Show your work.)

Answer:

$$\begin{aligned}
&\lim_{n\to\infty} \frac{T(n)}{f(n)} \\
&= \lim_{n\to\infty} \frac{5n^6 + n^2 + 3}{logn + n^6 + n} \\
&= \lim_{n\to\infty} \frac{30n^5 + 2n}{\frac{1}{nln2} + 6n^5 + 1} \\
&= \lim_{n\to\infty} \frac{150n^4 + 2}{-\frac{2}{ln2}n^{-2} + 30n^4} \\
&= \lim_{n\to\infty} \frac{600n^3}{\frac{4}{ln2}n^{-3} + 120n^3} \\
&= \lim_{n\to\infty} \frac{600}{\frac{4}{ln2}n^{-6} + 120} \\
&= 5
\end{aligned}$$

By the Limit Lemma Theorem, $T(n)$ is $\Theta(logn + n^6 + n)$.
By the definition of Big-Θ, $T(n)$ is also $O(logn + n^6 + n)$

(You could also divide the nemerator and denominator by n^5 after differentiating once. Then you would not need to use the L'Hopital's Rule that many times.)

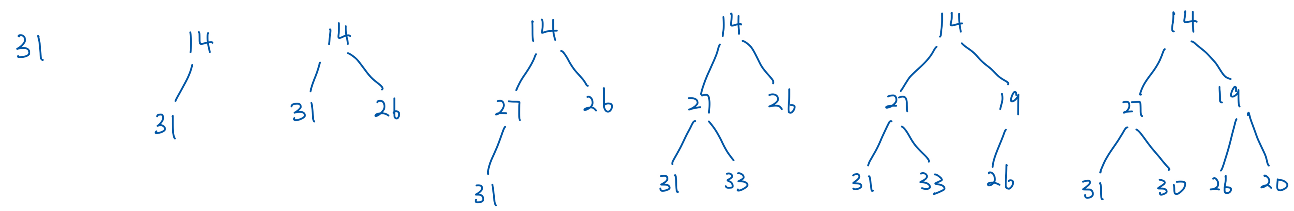
Question 5: MinHeap Review (40 points)

1. (15 points) Build a (binary) min heap by pushing the following numbers one by one in order:

$$31, 14, 26, 27, 33, 19, 20.$$

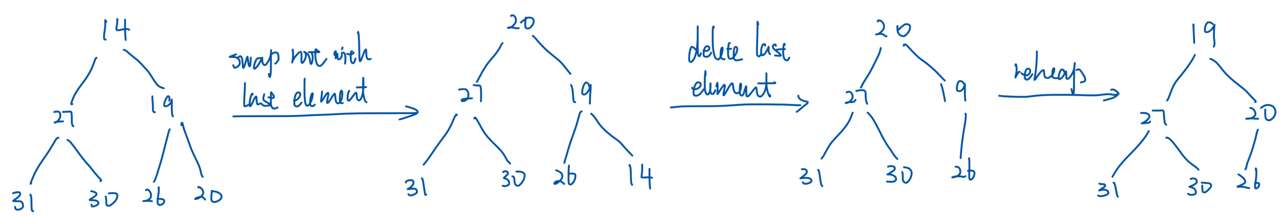
   Draw the min heap (as a binary tree) after pushing each number (no need to draw the intermediate steps of swapping).

   Answer:



2. (15 points) Perform the `pop` (or sometimes called `extractMin`) operation on the final resulting min heap in the above. (Show your work, including the intermediate steps of swapping.)

   Answer:



3. (10 points) Given an array `A` of numbers. Let `f(A)` be a function that pushes each element of `A` onto a min heap h, i.e., `f(A)` executes the following statements sequentially:

```
push(A[1])
push(A[2])
...
push(A[n])
```

   What is the asymptotic upper bound of `f(A)`? (Show your work.)

   Answer:
   Intuitively, each push takes $O(logn)$ time. So the total of n push takes $O(nlogn)$ time.
   More formally, the total runtime is

$$\begin{aligned} & log1 + log2 + ... + logn \\ & = log(1 * 2 * ... * n) \\ & = log(n!) \\ & = nlogn - nloge + O(logn) \\ & = O(nlogn) \end{aligned}$$