1 Compute

1. $\sum_{k=0}^{n}\sum_{i=-k}^{0}(\frac{1}{2})^i$

$$\begin{aligned}
\sum_{k=0}^{n}\sum_{i=0}^{k}(2)^i &= \sum_{k=0}^{n}2^{k+1}-1 \\
&= \sum_{k=0}^{n}2\cdot 2^k - \sum_{k=0}^{n}1 \\
&= 2\cdot(2^{n+1}-1)-(n+1) \\
&= 2^{n+2}-2-n-1 \\
&= 2^{n+2}-n-3
\end{aligned}$$

2. $log(\frac{log_5 log(32*2^{20})}{4^{log\sqrt{5}}})$

$$\begin{aligned}
log(\frac{log_5 log(32)+20log_5 2}{25}) &= log_2(\frac{log_5(5+20)}{5}) \\
&= log_2(\frac{2}{5}) \\
&= 1-log_2(5)
\end{aligned}$$

2. Use L'hopital's rule to determine the limit of

$$lim_{x\to\infty}\frac{xe^{lnx}-3(x+1)}{(6x+lnx)^2}$$

$$\begin{aligned}
lim_{x\to\infty}\frac{xe^{lnx}-3(x+1)}{(6x+lnx)^2} &= \frac{x^2-3x-3}{36x^2+12xlnx+(lnx)^2} \\
&= \frac{1}{36}
\end{aligned}$$

3. What is the growth of the below function

$$f(n) = 2^{log_2 n^3}+\sqrt{7\cdot n-3}+n^3log^2n+8logn^{\sqrt{n}}$$

1. $\Theta(\sqrt{n})$

2. $\Theta(n^3 log^2 n)$
3. $\Theta(n^3)$
4. $\Theta(n \cdot logn)$
5. Neither!

$$n^3 + \sqrt{7 \cdot n - 3} + n^3 log^2 n + 8logn^{\sqrt{n}}$$

(2) is the correct answer.

4. What is the growth of the below function:

$$f(n) \quad = \quad 2^{loglogn} + 5 \cdot logn + log^3 log^2 n + log^2 n$$

1. $\Theta(logn)$
2. $\Theta(logn^5)$
3. $\Theta(loglogn)$
4. $\Theta(log^2 n)$
5. $\Theta(log^3 logn)$

$$\begin{aligned} f(n) &= 2^{loglogn} + 5 \cdot logn + log^3 log^2 n + log^2 n \\ &= logn + 5logn + log^3 log^2 n + log^2 n \end{aligned}$$

$$\Theta(log^2 n)$$

5. Suppose a machine takes 10^{-9} seconds to execute a single algorithm step. When does the machine finish executing the below code when $n = 100$?

$$\begin{array}{lll} for(i = 0; & i \leq n; & i++) \\ & counting\_sort(a); & //a.length = n, max(a) = n^2 \end{array}$$

$\sum_{i=0}^{n+1} \Theta(n + n^2) = \Theta(n^3)$. $(10^2)^3 \cdot 10^{-9} = 10^{-3}$.

6. Assume we want to write code to calculate the subtraction of two numbers digit by digit. Provide the running time for your algorithm, assuming the inputs are two $n-$digits numbers.

First conver the second number to 10's complement then do the standard addition algorithm, this will be overall $\Theta(n)$.

7. Sort the below numbers using insertion sort [1,4,7,2,15,5,10]

1st Phase

1, $\boxed{4}$, 7, 2, 15, 5, 10

2nd Phase

1, 4, $\boxed{7}$, 2, 15, 5, 10

3rd Phase

1, 4, 7, $\boxed{2}$, 15, 5, 10

swap

4th Phase

1, 2, 4, 7, $\boxed{15}$, 5, 10

5th Phase

1, 2, 4, 7, 15, $\boxed{5}$, 10

6th Phase

1, 2, 4, 5, 7, 15, 10

7th Phase

1, 2, 4, 5, 7, 15, $\boxed{10}$

Final array

$\boxed{1, 2, 4, 5, 7, 10, 15}$

8. Prove that $f(n) = 3 \cdot logn^2 - loglogn + \sqrt{logn}\, is\, O(logn)$

$$\begin{aligned} 6logn - loglogn + \sqrt{logn} &\leq 6logn + \sqrt{logn} \\ &\leq 6logn + logn \\ &\leq 7logn, n \geq 2 \end{aligned}$$

9. Prove that if $f(n) = O(h(n)), g(n) = O(k(n))$ then $f(n) \cdot g(n) = O(h(n), k(n))$.

$$\begin{aligned} f(n) \cdot g(n) &\leq c_1 \cdot g(n) \cdot h(n) \\ &\leq c_1 \cdot c_2 \cdot h(n) \cdot k(n) \end{aligned}$$

$f(n) \cdot g(n) = O(h(n) \cdot k(n))$.

10. Compare the growth of $f(n) = \sqrt{2}^{logn^{14}}, g(n) = n^{1+2^{log5}}$

$$\begin{aligned} f(n) &= n^7 \\ g(n) &= n^6 \end{aligned}$$

$f(n) = \Omega(g(n)), f(n) = \omega(g(n))$

11. What is the growth of $n^2 + 2 \cdot n^2 + 3 \cdot n^2 + \ldots + n^4$?

$$n^2 \cdot \sum_{i=1}^{n^2} i = \Theta(n^6)$$

12. Prove $\forall k > 0, \epsilon > 0 \implies log^k(n) = o(n^\epsilon)$

We prove this with mathematical induction on k. Let $k = 1$ then

$$\begin{aligned} lim_{n\to\infty} \frac{log(n)}{n^\epsilon} &= lim_{n\to\infty} \frac{1}{ln(2) \cdot \epsilon \cdot n^\epsilon} \\ &= 0 \end{aligned}$$

hence we conclude $log(n) = o(n^\epsilon)$. Assume it holds for some integer k we show then that is must also hold for the $k+1$ integer.

$$\begin{aligned} lim_{n\to\infty} \frac{log^{k+1}(n)}{n^\epsilon} &= lim_{n\to\infty} \frac{(k+1)log^k(n)}{ln(2) \cdot \epsilon \cdot n^\epsilon} \\ &= 0\, (by\, IHOP) \end{aligned}$$

hence $k+1$ is also true. By the priniciple of mathematical induction the result holds for all $k \geq 1$.

13. $(log_2(n))^{2 \cdot log^3(n)} = \omega((\sqrt{n \cdot log_2(n)})!)$

$$\begin{aligned} lim_{n \to \infty} 2 \cdot log^3(n) \cdot log_2(log_2(n)) - \Theta(\sqrt{n \cdot log_2(n)} \cdot log(\sqrt{n \cdot log_2(n)}) &= \infty \\ 2^{\infty} &= \infty \end{aligned}$$

Hence we conclude $(log_2(n))^{2 \cdot log^3(n)} = \omega(\sqrt{n \cdot log_2(n)})$

14. Given a sorted integer array with the size of n, provide an algorithm with $O(log(n))$ running time that checks if there is an i for which $a[i] = i^2 + 1$.

```python
def binary_search(A):
    lo, hi = 0, len(A) - 1
    while lo <= hi:
        mid = (lo + hi) // 2
        value = mid * mid + 1
        if A[mid] < value:
            lo = mid + 1
        elif A[mid] > value:
            hi = mid - 1
        else:
            return True
    return False
```