

CSE 221: Algorithms

1. Which search algorithm will you choose among Binary and Ternary Search and why?
Can we always use Binary or Ternary Search? If we can't, which algorithm can be used in that case?

2. Solve the following questions about time complexity.

- a. Calculate the time complexity of the following code snippet:

```
int p = 0;
for (i = n / 2; i > 1; i /= 6) {
    for (j = 2; j <= n; j *= 4) {
        for (k = 2; k <= j; k *= 3) {
            p = p + n / 2;
        }
    }
}
```

- b. Calculate the time complexity of the following code snippet:

```
int p = 0;
for (i = n; i > 1; i -= 1) {
    for (j = 2; j*j <= n; j += 1) {
        p = p + n;
        if (p > (n + i)) // ^ sign means XOR in code
        {
            break;
        }
    }
}
for (i = n / 2; i > 1; i /= 6) {
    for (j = 2; j <= i; j *= 4) {
        for (k = 1; k <= j; k *= 3) {
            p = p + n / 2;
        }
    }
}
```

3. If $f(n) = \log n + 5$; $g(n) = \log n^2$ then prove $f(n) = \Theta(g(n))$
4. Prove whether $f(n) = n^2 \log n^2$ is $O(\log n^3)$

5. Your friend gave you an integer (*key*) and asked you to find its integer square root without using built-in functions like `sqrt()` or exponentiation. For example, if the input is 25, the output should be 5 because $5^2 = 25$. If the square root is not an integer, return the floor value of the actual square root. For example, for an input of 10, the output should be 3.

Your friend's initial approach is to use linear search (time complexity $O(n)$) as shown below.

```
def LinearSearchToFindSquareRoot(key):  
    result = -1  
    for i in range(1, key, 1):  
        if i * i <= key:  
            result = i  
    return result
```

Now to find the square root he is working on a possible solution **P**, which will result in time complexity $O(\sqrt{n})$. As an algorithm student, you propose another algorithm **R** within time complexity $O(\log_2 N)$, which is much faster than $O(\sqrt{n})$.

- i. Present the **solution your friend** was trying to achieve (**Algorithm P**) with pseudocode/programmable code/step-by-step logical instructions.
 - ii. Present your proposed solution (**Algorithm R**) with pseudocode/programmable code/step-by-step logical instructions.
7. Write the steps to find the minimum element in a rotated sorted array using binary search.
- Sample Input: `arr = [4, 5, 6, 7, 0, 1, 2]`
Output: 0
Sample Input: `arr = [3, 4, 5, 1, 2]`
Output: 1