Topic: Big-Oh Notation

Activity Guidelines

Group Size: 3

Method of Assigning Students: Count the number of students in the class, divide by 3, count off from 1 to the quotient, and group identical numbers.

Materials:
- Handout (one copy per group) with questions to be answered at the end of the session

Roles:
- Coordinator/Leader: Clarifies goals and objectives, allocates roles for each team member and divides the tasks within the group.
- Monitor/Evaluator: Person designed to evaluate the different ideas to approach the problem and make an accurate judgment of the most beneficial option.
- Implementer: Person in charge to transform discussions and ideas into a technical solution for the given problem.

Individual Accountability: Every member of the team is expected to be actively involved in the problem-solving activity and collaborate according to his or her assigned role.

Activity Summary

1. Every team is required to identify the following notations and provide an example for each of them.

   - $O(1) = “Constant Time”$
   - $O(\log n) = “Logarithmic Time”$
   - $O(n) = “Linear Time”$
   - $O(n^2) = “Quadratic Time”$
   - $O(2^n) = “Exponential Time”$
Elementary Data Structures
Peer Session

Big-Oh Notation

Basic Rules

1. Nested loops are multiplied together.
2. Sequential loops are added.
3. Only the largest term is kept, all others are dropped.
4. Constants are dropped.
5. Conditional checks are constant (i.e. 1).

O(1) - describes an algorithm that will always execute in the same time (or space) regardless of the size of the input data set.

```java
boolean IsFirstElementNull(String[] strings) {
    if(strings[0] == null)
        return true;
    return false;
}
```

O(N) - describes an algorithm whose performance will grow linearly and in direct proportion to the size of the input data set.

```java
boolean ContainsValue(String[] strings, String value) {
    for(int i = 0; i < strings.Length; i++) {
        if(strings[i] == value)
            return true;
    }
    return false;
}
```

O(N^2) - represents an algorithm whose performance is directly proportional to the square of the size of the input data set.

```java
boolean ContainsDuplicates(String[] strings) {
    for(int i = 0; i < strings.Length; i++){
        for(int j = 0; j < strings.Length; j++) {
            if(i == j)
                continue;
            if(strings[i] == strings[j])
                return true;
        }
    }
    return false;  }
```
O(2^N) - denotes an algorithm whose growth will double with each additional element in the input data set. The execution time of an O(2N) function will quickly become very large.

- Towers of Hanoi
- 8-queens