Prelab 12

We encourage you to work together on the Pre Lab. The Pre Lab is not graded but will help you prepare for your lab session. If you have any questions on the material of the Pre lab, first check the book and recitation slides. If you continue to have any doubt about it, please email your recitation TA or the course instructor.

This prelab has two parts:

- The first part explains the meaning of Big-O notation and the complexity of algorithms
- The second part explains how to use Big-O notation to analyze the complexity of the Algorithms.

Big-O Notation

There is always more than one way to solve a problem. But how can we know which solution is better?

In computer science we use the **Big-O** notation to describe the complexity and asymptotic behavior of a function. Basically, it tells the number of steps (or how fast) the program needs to be executed to accomplish an algorithm.

**How can we determine the complexity?**

There are three main steps to follow:

1. **Step 1:** we need to identify what kind of “input” we have. The size and type of data the algorithm will estimate.
2. **Step 2:** Evaluate the number of operations needed to process each piece of the input. This step will give a better idea how many steps are needed to accomplish the algorithm.
3. **Step 3:** To estimate the total running time under the Big-O notation, constants and small terms can be eliminated.

**How can we determine the running time?**

It depends of the number of statements needed:

1. **Sequence of statements**
   - statement 1
   - statement 2
   - .......
   - statement n
2. The total time is found by adding the time of each statement:
   - TOTAL time = time(statement 1) + time(statement 2) + ... + time(statement n)

In if-statments the total time is constants since only one operation is performed. In the loops, the total time will depended on the number of times the loop is executed.
Comparison of complexities

From Fastest to slowest growth rate:

- O(1) – constant time
- O(log_2 n) – logarithmic time
- O(n) – linear time
- O(n^2) – quadratic time
- O(2^n) – exponential time
- O(n!) – factorial time

Big-O Examples

The amount of steps needed to reverse a string depends on the length of the word. The following example shows how we can apply the three steps explained above in order to calculate the complexity of the program by the Big-O notation.

```python
# Example 1
# Step1
output = ""
word="CS177"

for i in range(0, len(word)):       # for- loop where i: 0,1,2,4   |
    n=len(word)       O(n)
    output = output + word[len(word)-i-1]
    #to assign each letter to "output" it goes until the length of the word

>>> print (output)                   # Constant time to print       |
    n=len(word)       O(1)
    >>> 771CS

# Step2:   O(n)+O(1)
#Step 3:
# O(n)+O(1) = O(n)
```

In Prelab9, we have learned how to encode and work with Matrices by using lists. Now we can analyze the three steps needed to calculate the complexity of the program and how long it takes to traverse a matrix and perform a matrix-addition.

```python
# Example 2

columns=3
rows=3

#Step1:
M1 = [[ 1, 2, 3], [ 4, 5, 6], [ 7, 8, 9]]
```
M2 = [ [-10, -9, -8], [-7, -6, -5], [-4, -3, -2] ]

M3 = [[0]*columns for i in range(rows)]  # One loop where i: 0,1,2
n=rows          O(n)
for x in range(rows):
    n=rows
    for y in range(columns):  # for every row - loop- y: 0,1,2
        n=columns
        # indexing each matrix
        time          O(n^2)
        M3[x][y] = M1[x][y]+M2[x][y]  # to assign values to M3 takes the
        # to traverse the whole matrix
    for x in range(rows):
        n=rows
        for y in range(columns):
            n=columns
            # indexing each matrix
            time          O(n^2)
            M1[x][y] = M3[x][y]+M2[x][y]  # to assign values to M3 takes the
            # to traverse the whole matrix
print(M3)    # It takes constant time.

Step2:  O(n)+ O(n^2)+0(1)
#Step 3:
# O(n)+ 0(n^2)+0(1) = 0(n^2)

From:
http://courses.cs.purdue.edu/ - Computer Science Courses

Permanent link:
http://courses.cs.purdue.edu/cs17700:fallprelab12

Last update: 2012/11/04 16:26