Lab 3: Big-O and Mystery Functions

Due: February 27th at 12pm, noon

Overview

In this lab, you will answer the following math and algorithm questions and complete the mystery function exercise. You will submit a *hardcopy* of this lab. You may type your solution, if you wish. **This is an individual lab**. You can retrieve requisite code from `update35` for the mystery function portion of the lab.

Deliverables

Your submission should minimally include answers to all questions, written support of the mapping of mystery functions, and at-least two print out of graphs to support that argument.

Submission

You will submit this lab in *hardcopy* to my mail box in Science 239.

Short Answer Questions (10 points)

1. (**2pt**) Justify the following Big-O for the given functions:

   (a) $3n^4 - 2n^3 + 100$ is $O(n^4)$

   (b) $log_8(n) + 50$ is $O(lg(n))$

   (c) $\frac{1}{2}n - lg(n)$ is $O(n)$

   (d) $lg(n) + \sqrt{n}$ is $O(\sqrt{n})$ (**challenge**)

2. (**2pt**) Prove the following claims by induction:

   (a) $\forall n > 1, \quad n^3 > n^2$

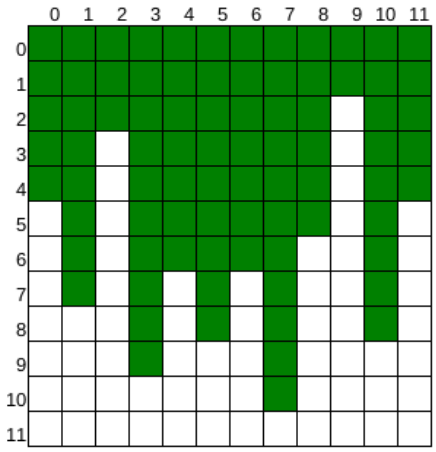
   (b) $\forall n > 1, \quad \sum_{i=1}^{n} i^3 = \frac{n^2(n+1)^2}{4}$

   (c) $\forall n > 1, \quad (n^3 + 2n) \mod 3 = 0$ (**challenge**)

3. (**2pt**) Using a loop-invariant and induction, prove that the following function verifies that consecutive integers have an absolute value difference of at least 2.

```
bool two_apart(int array[], int size){
  for(int i=0;i < size-1;i++){
    if(! ((array[i] - array[i+1]) >= 2 ||
          (array[i+1] - array[i]) >= 2)){
       return false;
    }
  }
  return true;
}
```

4. (**2pt**) The image below shows an n by n grid that is stored in memory as a two dimensional array. The element at row i column j can be indexed in constant time using `grid[i][j]`. Each cell in the grid contains either a 1 or a 0 (indicated in the figure with green and white blocks). In any column, all the one's appear before any of the zeros. Given such a grid, design an $O(n)$ algorithm for finding the column with the most ones (tallest green tower). Note there are $O(n^2)$ cells, so you cannot check every cell in the grid.



5. (**2pt**) Describe an algorithm for finding both the minimum and maximum (simultaneously) of n numbers using fewer than 3n/2 comparisons. The following pseudocode finds the max using n-1 comparisons.

```
def findMax(Array A, size n>0):
  max=A[0]
  for i in 1 to n   (inclusive)
    if(A[i] > max)
      max=A[i]
  return max
```

Mystery Functions (10 points)

In this part of the lab, you will use the provided program, `timer_func` to identify the mystery functions. You do this in two parts, each worth 5 points: First, identify the Big-O of each of the functions, and then use the `timer_func` and the associated graphs to identify which mystery functions matches the functions below.

Possible Functions

```
void ex1(long n){
  long a;
  for(long i=0;i<n;i++){
    a=i;
  }
}

void ex2(long n){
  long a;
  for(long i=0;i<n;i=(i+3)/2){
    a=i;
  }
}

void ex3(long n){
  long a;
  for(long i=0;i<n*n;i++){
    a=i;
  }
}

void ex4(long n){
  long a;
  for(long i=0;i<n;i++){
    for(long j=0;j<=i;j++){
        a=i;
    }
  }
}

void ex5(long n){
  long a;
  for(long i=0;i<n*n;i++){
    for(long j=0;j<=i;j++){
      a=i;
    }
  }
}


void ex6(long n){
  long a;
  long k=1;
  for(long i=0;i<n;i++){
    for(long j=0;j<=k;j++){
      a=i;
    }
    k=k*2;
  }
}
```

Using `function_timer`

Once you have identified the Big-O of the above functions, your task is to provide a mapping between the functions and the mystery functions used in `function_timer`. The `function_timer` program will run each of the functions, time their operations, and output a plottable format string whose associate plot (via `gnuplot` you can inspect. Here is the usage:

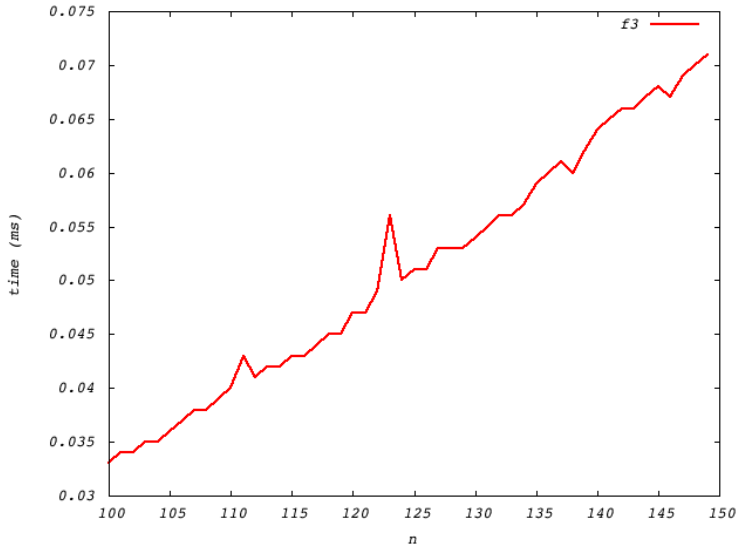
```
function_timer [OPTIONS] | gnuplot

function_timer will time mystery functions for some input parameter n
and output results in a plotable format for persistent gnuplot. To
properly use, results should be 'piped' to gnuplot plotting function
with the 'persist' flag, like in the above example.

OPTIONS:
        -h              print this help screen
        -n min_n        set the min value for n (dflt: 1)
        -m max_n        set the max value for n (dflt: 10)
        -[1-6]          Turn on mystery function number, e.g.
                        to run function 2 and 3: function_timer -2 -3
        -s out.png      Save the graph to a file out.png instead of graphing live
```

And here is a sample usage, and an output graph produced with the `-s` flag:

```
bash> ./function_timer -s out.png -n 100 -m 150 -3 | gnuplot
```



You can also include multiple mystery functions to compare by adding more numeric flags, e.g., to compare function 1 and 2, use flags `-1 and -2`. Note that you must 'pipe' the output of the program to the plotting program, `gnuplot`. The pipe, | , just sets the output of one program to the input of the other. for viewing the plots, you can either save the final result with `-s` flag, or watch live as the results come in.

In addition to determining the mapping between mystery functions and the functions above, **you must support your argument in writing and provide at least two graphs in your hard-copy submission.**