CISC4080 Computer Algorithms Homework (2)[1]

[1] Distinguish yourselves, folks!

Part A. Divide-and-Conquer Algorithm Complexity Analysis (20 points)

1. Give the time complexity with Θ bound for A divide-and conquer algorithm with the following recurrence formula

   1. $T(n) = T(n-1) + c^n$, where $c > 1$
   2. $T(n) = 16T(n/4) + n^3$
   3. $T(n) = 25T(n/36) + n^{3/2} \log n$

2. Show the best case of merge sort has complexity $\Theta(n)$ without using the master method.

Part B. Targeted Sorting (15 points)

- 1. Implement and test a method (function) to check if input array is sorted, which means the array entries are sorted in an ascending or descending order. It can have the following signature[2]

  ```
  bool isSorted(const int*, int &)
  ```

- 2. What are the worst and average time complexities of this method?

- 3. Combine this method with a quick-sort method implemented by yourself and test your sorting method with the best, worst and average cases for input arrays with $n = 10^4$ entries respectively[3].

[2] You don't need to follow this signature "exactly" in your implementation

[3] You may need to use some sample codes in your homework 1

Part C. A GCD Calculator (15 points)

Greatest common divisor

- If $a|b$ $c|b$,then we can say a is a common divisor of a and b . The largest factor of a and b is called the greatest common divisor of a and b: $GCD(a, b)$. For examples, $20 = GCD(20, 100)$, $2 = GCD(10, 8)$
- That is, $GCD(a, b)$ is the largest positive integer dividing both a and b. If $GCD(a,\ b) = 1$, we say a and b are relatively prime.

How to compute GCD of a and b?

- The typical algorithm to compute a GCD is called Euclidean algorithm invented more than 2000 years ago!
- **Ideas of the Euclidean algorithm**: *divident* is replaced by *divisor* and *divisor* is replaced by *remainder* recursively!
- It has the following algorithm description
  - To compute the GCD of a and b, we assume $a > b$ always. If not, just switch a and b .

```
Algorithm GCD(a,b)
Input: integer a, b (
Output: GCD(a,b)
% divide a by b:
   a = qb + r
   If r = 0
      return b
   else
      a = b;
      b = r;
      GCD(a, b)
   end
```

1. Implement Euclidean algorithm to write a GCD calculator and compute at least the following GCDs.

```
GCD(482, 1180)
GCD(8756, 23485)
GCD(87561, 23485)
GCD(1234567, 2008479)
GCD(578129810, 20092330)
```

- If you program in JAVA, remember to use long type for the dividend and divisor.
- The following codes and outputs may give you a little bit hint.
- You can write your own codes and may not need to follow these sample codes.

```
public static void main(String[] args) throws IOException{

   computeGCD myGCD=new computeGCD();

   System.out.println("Enter your Dividend:\n");

   long Dividend = myGCD.read_long();
   System.out.println("Enter your Divisor:\n");

   long Divisor = myGCD.read_long();
   myGCD=new computeGCD(Dividend, Divisor);

   long gcd = myGCD.gcd();

   System.out.println("Results: gcd(" + Dividend + ", " + Divisor + ") = " + gcd);
}
```

- Sample outputs

```
>java computeGCD
Enter your Dividend:
1234567
Enter your Divisor:
2008479
current divident-->1234567
current divident-->773912
current divident-->460655
current divident-->313257
current divident-->147398
current divident-->18461
current divident-->18171
current divident-->290
current divident-->191
current divident-->99
current divident-->92
current divident-->7
current divident-->1
current divident-->0
Results: gcd(1234567, 2008479) = 1
```

2. What's the time complexity of this algorithm?[4]

[4] Hint: you can count the number of recursion calls to get some idea!

What should you turn in?

1. A hardcopy of all your homework printout in class (Oct 29, 2013).

2. A folder contains all your homework assignments. If there is a programming assignment, you need to include workable source codes and related output in this folder. Please name your folder as first-name_last-name_CISC4080_homework_2. For example, John_Smith_CISC4080_homework_2 if your name is John Smith.

3. Send the zipped file (.zip instead of .rar) of your folder to xhan9@fordham.edu before 11:59 pm Oct 29, 2013.