CS 140 Algorithms   Problem Set 1   September 20, 2017

> *Due date:* Email your solutions or submit a written copy to the TA before 7:30PM, September 28.

Problem 1 Let $T(n)$ be the time complexity of an algorithm to solve a problem of size n. Assume $T(n)$ is $O(1)$ for any n less than 3. Solve the following recurrence relations for $T(n)$.

(a) $T(n) = 9T(n/3) + n^2$.

(b) $T(n) = 5T(n/3) + n$.

(c) $T(n) = 7T(\sqrt[8]{n}) + \log^2(n)$.

(d) $T(n) = T(n/2) + T(n/4) + \Theta(n)$.

(e) Consider the following functions. Within each group, sort the functions in asymptotically increasing order, showing strict orderings as necessary. For example, we may sort $n^3, n, 2n$ as $2n = O(n) = o(n^3)$. **Hint:** you may find Stirling's approximation helpful.

  (1) $\log(n), \sqrt{n}, (\log n)^2, \log\log n$.

  (2) $n^{4/3}, n\log n, n^2, \log n!$.

  (3) $n!, n^n, e^n, 2^{\log n^{\log\log n}}$.

Problem 2 Analyze Algorithm 1 and Algorithm 2, and compute the time complexity for each algorithm.

```
input : An integer n > 2
output: An integer s

s ← 2;
for i ← 3 to n do
    notPrime ← false;
    for j ← 2 to floor(√i) do
        if j|i then
            notPrime ← true;
            break;
        end
    end
    if notPrime == false then
        s ← s + i;
    end
end
```

Algorithm 1: An algorithm to sum up prime numbers

```
input  : A graph Graph and a starting vertex root of Graph
output: Goal vertex node
/* Assume all set and queue operations take O(1) time */
create empty set S;
create empty queue Q;
add root to S;
Q.enqueue(root);
while Q is not empty do
    current ← Q.dequeue();
    if current is the goal then
        node ← current
    end
    for each node n that is adjacent to current do
        if n is not in S then
            add n to S;
            Q.enqueue(n);
        end
    end
end
```

Algorithm 2: An algorithm to find the goal node

Problem 3 Analyze Algorithm 3 and Algorithm 4, and compute the time complexity for each algorithm.

```
input  : An integer n > 0
output: An integer sum
Function fakeFib(n):
    if n ≤ 2 then
        return 1;
    end
    return fakeFib(n − 1) + fakeFib(n − 2) * fakeFib(n − 2)
```

Algorithm 3: An algorithm to calculate the fakefib value

Problem 4 Use induction to prove the following inequalities.

(a) $\sum_{i=1}^{n} \frac{1}{i^2} < 2$.

(b) $n^{n+1} > (n+1)^n$ for integer $n > 2$.

Problem 5 Fix a positive number α, choose $x_1 > \sqrt{\alpha}$, and define $x_2, x_3, x_4, ...,$ by the formula

$$x_{n+1} = \frac{1}{2}(x_n + \frac{\alpha}{x_n})$$

Cont.

```
input  : An integer n > 0
output: An integer sum
Function dac(n):
    if n < 1 then
        return n;
    else if 3|n then
        tmp ← √dac(2n/3);
    for i ← 1 to n do
        tmp ← tmp + i;
    end
    return tmp+ dac(n/3) + dac(2n/3)
```

Algorithm 4: A new divide and conquer algorithm

(a) Prove that $\{x_n\}$ decreases monontonically and that $\lim_{n \to \infty} x_n = \sqrt{\alpha}$.

(b) Put $\varepsilon_n = x_n - \sqrt{\alpha}$, and show that

$$\varepsilon_{n+1} = \frac{\varepsilon_n^2}{2x_n} < \frac{\varepsilon_n^2}{2\sqrt{\alpha}}$$

so that, setting $\beta = 2\sqrt{\alpha}$,

$$\varepsilon_{n+1} < \beta \left(\frac{\varepsilon_1}{\beta} \right)^{2^n}$$

(c) This is a good algorithm for computing square roots, since the recursion formula is simple and the convergence is extremely rapid. For example, if $\alpha = 3$ and $x_1 = 2$, show that $\varepsilon/\beta < 1/10$ and that therefore

$$\varepsilon_5 < 4 \cdot 10^{-16}, \varepsilon_6 < 4 \cdot 10^{-32}$$

The End.