QUEENS COLLEGE Department of Computer Science
CSCI 313 Practice problems on O, Omega, Theta and the Master theorem

Instructor: Krishna Mahavadi

These are some of the solutions to practice problems. Not all problems have solutions here.

Solutions to some older problems might not make use of generics. Generics are now required in this course.

Problem 1 Give useful Θ estimates for the following functions $t(n)$.

(a) $t(n) = 5log_2(n^2) + (log_2(n))^2 + log_4(n) + (log_2(100))^3$.

Answer:

Simple Θ estimates for the terms are $log(n)$, $(log(n))^2$, $log(n)$ and 1. The second of these grows fastest so our estimate is $t(n) = \Theta((log(n))^2)$.

(b) $t(n)$ satisfies $t(n) = 2t(n/2) + n$.

Answer:

The recursive and nonrecursive costs are both $\Theta(n)$ so the Master Theorem gives $t(n) = \Theta(nlog(n))$.

(c) $t(n)$ satisfies $t(n) = 4t(n/3) + n$.

The recursive cost is $\Theta(n^{log_3(4)})$ which grows faster than the nonrecursive cost of $\Theta(n)$. Hence the Master Theorem gives $t(n) = \Theta(n^{log_3(4)})$.

(d) $t(n)$ is the running time of the following function:

```
public static void shuffle(int []x, int a, int b, int n) {
   for (int i = 0; i < n; i+=2) {
      int temp = x[a + i];
      x[a + i] = x[b + i];
      x[b + i] = temp;
   }
}
```

Answer:

The function performs $n/2$ iterations of a loop and each iteration has constant running time. Hence $t(n) = \Theta(n)$.

(e) $t(n)$ is the running time of the following function that calls shuffle from (d):

```
public static void multiShuffle(int []x, int a, int n) {
   if (n == 0) return;
   multiShuffle(x, a, n/2);
   multiShuffle(x, a + n/4, n/2);
   multiShuffle(x, a + n/2, n/2);
   shuffle(x, a, a + n/2, n/2);
}
```

Answer:

Here, the recursive cost is $\Theta(n^{log_2(3)})$ which grows faster than the nonrecursive cost of $\Theta(n)$ (given by (d)). Hence the Master Theorem gives $t(n) = \Theta(n^{log_2(3)})$.

Problem 2 Give useful O-estimates of the run times of the following methods:

(a) The method $addHead$ for a singly linked list that has size n.

Answer: $O(1)$

(b) An efficient method to calculate the power x^n (consider the run time as a function of n, the time should be considered as being proportional to the total number of additions, subtractions, multiplications, and divisions performed).

Answer: $O(log(n))$

(c) An efficient method to sort an array of n numbers into order.

Answer: $O(nlog(n))$

For (d) and (e), consider the following recursive function, in which A represents an integer constant:

```
int f(int n) {
   if (n <= 0) return 1;
   int ans = f(n/2) * 2;
   for (int i = 1; i<= n; i++)
      for (int j = 1; j <= n; j++)
         ans += i / j;
   for (int k = 1; k < A; k++)
         ans -= f(n/2 - k);
   return ans;
}
```

(d) In the case where $A = 3$ estimate the run time of $f(n)$.

Answer: $O(n^2)$

(e) In the case where $A = 4$ estimate the run time of $f(n)$.

Answer: $O(n^2 log(n))$