Data Structures and Algorithms

COSC 336 Assignment 4 - solutions

Exercise 1. For each of the following functions, give a $\Theta(t(n))$ estimation with the simplest possible $t(n)$ (for example $3n^2 + 5n \log n = \Theta(n^2)$).

1. $13n^2 - 2n + 56$

   Solution: Polynomial of degree 2, so it $\Theta(n^2)$. □

2. $2.5 \log n + 2$

   Solution: After dropping the constants, we get $\Theta(\log n)$. □

3. $n(12 + \log n)$

   Solution: $\Theta(n \log n$. □

4. $1 + 2 + 3 + \ldots + 2n$

   Solution: The sum is an arithmetic series. We apply the formula and we get $2n(2n+1)/2 = \Theta(n^2)$. □

5. $1 + 2 + 3 + \ldots + n^2$

   Solution: The same as above, we get that the sum is $n^2(n^2 + 1)/2 = \Theta(n^4)$. □

6. $\log(n^3) + 10$

   Solution: This is $\Theta(\log n)$, because $\log(n^3) = 3 \log n$. □

7. $\log(n^3) + n \log n$

   Solution: This is $\Theta(n \log n)$, because $\log(n^3) = 3 \log n$ and by the Max Rule, we can drop it. □

8. $n \log(n^3) + n \log n$

   Solution: This is $\Theta(n \log n)$, because $\log(n^3) = 3 \log n$ and therefore the two terms in the sum are $\Theta()$ of each other. □

9. $2^{2 \log n} + 5n + 1$

Solution: This is $\Theta(n^2)$ because $2^{2\log n} = 2^{\log(n^2)} = n^2$. $\square$

Exercise 2.

1. Evaluate the following postfix arithmetic expression: $10\ 3\ 4\ -\ 5\ *\ /$

   Solution: In infix notation, the expression is $10/((3-4)*5)$, which is equal to -2. We can also use the algorithm using a stack that evaluates an expression in postfix notation. $\square$

2. Convert the following infix arithmetic expression to postfix notation: $(((2+3)*5)-15)$

   Solution: $2\ 3 + 5 * 15-$ $\square$

Exercise 3.

Consider the following algorithms A and B for the problem of computing $2^n \pmod{317}$ (This is the modular exponentiation problem that we have discussed in class. Algorithm A is the one that we have seen in class, and algorithm B is a variant of it.

```
Algorithm A.

mod_exp_A(n) {
   if (n== 0) return 1;
   else {
          t = mod_exp_A(n/2);
          if (n is even) return t*t (mod 317);
          if (n is odd) return t*t*2 (mod 317);
}}

Algorithm B.

mod_exp_B(n) {
   if (n== 0) return 1;
   else {
          if (n is even) return   mod_exp_B(n/2) * mod_exp_B(n/2) (mod 317);
          if (n is odd) return  mod_exp_B(n/2) * mod_exp_B(n/2) *2 (mod 317);
}}
```

1. Write the recurrence for the runtime $T_A(n)$ of algorithm A, and solve the recurrence to find a $\Theta(\cdot)$ estimation of $T_A(n)$.

   Solution: $T_A(n) = T_A(n/2) + c$ (where c is some constant). Using the Master Theorem, we solve the recurrence and obtain $T_A(n) = \Theta(\log n)$. Note: it is the same recurrence as the one for Binary Search. $\square$

2. Write the recurrence for the runtime $T_B(n)$ of algorithm B, and solve the recurrence to find a $\Theta(\cdot)$ estimation of $T_B(n)$.

   Solution: The algorithm B is calling itself recursively *two* times. So the recurrence is $T_B(n) = 2T_B(n/2) + c$. We can use the Master Theorem: we compare $n^{\log_2 2}$ vs. c, the Winner is $n^{\log_2 2} = n$, the ratio Winner/Loser is a polynomial in n, and therefore $T_B(n) = \Theta(n)$ □

3. Which algorithm is faster? (Note: There is a huge difference between T_A and $T_B(n)$.)

   Solution: Obviously, A is much faster (because $\log n = o(n)$). □

Exercise 4. Give a $\Theta(\cdot)$ evaluation for the runtime of the following code:

```
i= 1; x=0;
while(i <= n) {
      j=1;
      while (j <= i) { x=x+1; j= 2*j; }
      i= 2*i;
}
```

Hint: You can assume that n is a power two. Then i from the outer loop takes successivley the values: $1, 2, 2^2, 2^3, ..., 2^{\log n}$.)

Solution: We assume $n = 2^k$, so $k = \log n$. The inner loop has runtime $\Theta(\log i)$. Moving to the outer loop, we see that i takes in order the values: $1, 2, 2^2, 2^3, \ldots, 2^k$. So the total runtime is $\log 1 + \log 2 + \log 2^2 + \log 2^3 + \ldots \log 2^k = 1+2+3+\ldots+k = k(k+1)/2 = \Theta(k^2) = \Theta(\log^2 n)$. □