Analysis – Recursive Algorithms: Part 1

Overview

- □ Asymptotic Notations Review
- Recurrence relation
 - Factorial
 - Binary Search
 - ☐ Merge Sort
- Solution of recurrence

Asymptotic Notations - Review

- \Box O (Oh), Ω (Omega), θ (Theta)
- Definitions (sets)
- ☐ Insertion Sort Analysis
 - Worst Case
 - Best Case

Asymptotic Notations - Exercises

- 1. Is $3n^2 100n + 6$ is $O(n^3)$?
- 2. Is $3n^2 100n + 6 is \Omega(n^3)$?
- 2. Which algorithm do you prefer?
 - a. $\theta(n^2)$ or $\theta(n)$
 - **b.** $\theta(n)$ or $\theta(\lg n)$

Factorial - Recursive function

```
int factorial (int n) {
// returns the factorial of n, given n>=0
   if (n<=1)
      return 1;
   else
      return n * factorial (n-1);
}</pre>
```

Running Time T(n) = ?

Factorial – Running Time

$$T(n) = T(n-1) + c$$
 if $n>1$
= d if $n <= 1$

Asymptotic Running Time?

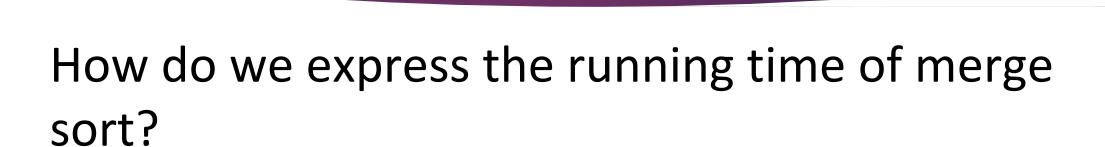
How do we express the running time of binary search?

Binary Search – Algorithm

```
BinarySearch (A, m, n, k)
  if (m>n) return -1;  //Base Case
  mid=(m+n)/2
  if A[mid]=k return mid;  //Base Case
  else if k < A[mid]
    BinarySearch(A, m, mid-1, k)
  else if k > A[mid]
    BinarySearch(A, mid+1, n, k)
```

Binary Search – Running Time

$$T(n) = T(n/2) + c$$
 if $n>1$
= d if $n <= 1$



Merge Sort - Recursive Algorithm

```
MERGE-SORT(A, p, r)

1 if p < r

2 q = \lfloor (p+r)/2 \rfloor

3 MERGE-SORT(A, p, q)

4 MERGE-SORT(A, q+1, r)

5 MERGE(A, p, q, r)
```

Ref: CLRS Book

Merge Sort – Running Time

$$T(n) = 2T(n/2) + cn \quad if n > 1$$
$$= c \quad if n = 1$$

Running Time - Recurrence equation

- > Running time of recursive algorithms described by a recurrence equation or recurrence
- ightharpoonup T(n) in terms of running times of smaller subproblems
- ➤ Solve the recurrence using mathematical tools to get bounds on the running time

Solving recurrence - Factorial

$$T(n) = T(n-1) + c$$
 if $n>1$
= d if $n <= 1$

Solving recurrence - Factorial

$$T(n) = c + T(n-1)$$
 if $n > 1$

Solving recurrence - Factorial

$$T(n) = c + T(n-1)$$
 if $n>1$
 $T(n-1) = c + T(n-2)$ if $n>2$

$$T(n) = c + c + T(n-2)$$
 if $n>2$
= $2c + T(n-2)$ if $n>2$

Factorial – Running Time

$$T(n) = 2c + T(n-2) \quad if n > 2$$

$$T(n) = 3c + T(n-3)$$
 if $n>3$

In general?

Factorial – Running Time

$$T(n) = 2c + T(n-2) \quad if n > 2$$

$$T(n) = 3c + T(n-3)$$
 if $n>3$

In general,

$$T(n) = ic + T(n-i)$$
 if $n > i$

when
$$i = n-1$$
,

$$T(n) = (n-1)c + T(1) = (n-1)c + d = cn - c + d$$

$$T(n)$$
 is $\Theta(n)$

Solving Recurrence – Iteration method

- > Expand (iterate) the recurrence
- \triangleright Express as a summation of terms dependent only on n
- **Recursion Tree** Visualize the iteration of recurrence

Divide and Conquer – Recurrence

$$T(n) = \Theta(1)$$
 if $n < = c$
= $a T(n/b) + D(n) + C(n)$ otherwise

- \triangleright Number of subproblems a
- \triangleright Each subproblem size is 1/b the size of the original
- ightharpoonup D(n) time to divide the problem into subproblems
- ightharpoonup C(n) time to combine the solutions

Divide and Conquer – Recurrence

$$T(n) = d$$
 if $n <= 1$
= $2 T(n/2) + c$ otherwise

Solve using iteration method

Reference

T H Cormen, C E Leiserson, R L Rivest, C Stein *Introduction to Algorithms*, 3rd ed., PHI, 2010