Incremental & Recursive Approaches to Sorting

Lecture 2

Insertion Sort

```
ISERTION-SORT()
For j ← 2 to length[A]
    do key ← A[j]
        » Insert A[j] into the sorted sequence A[1…j-1]
        i ← j – 1
        while i > 0 and A[i] > key
            do A[i+1] ← A[i]
                i ← i – 1
        A[i+1] ← key
```

Insertion Sort

ISERION-SORT()	cost	times
1. **For** $j \leftarrow 2$ to *length*[*A*]	c_1	n
2. **do** $key \leftarrow A[j]$	c_2	$n-1$
3. » Insert $A[j]$ into the sorted sequence $A[1 \ldots j-1]$	0	$n-1$
4. $i \leftarrow j - 1$	c_4	$n-1$
5. **while** $i > 0$ and $A[i] > key$	c_5	$\sum_{j=2}^{n} t_j$
6. **do** $A[i+1] \leftarrow A[i]$	c_6	$\sum_{j=2}^{n} (tj - 1)$
7. $i \leftarrow i - 1$	c_7	$\sum_{j=2}^{n} (tj - 1)$
8. $A[i+1] \leftarrow key$	c_8	$n-1$

Insertion Sort Running Time

$$T(n) = c_1 n + c_2(n-1) + c_4(n-1) + c_5 \sum_{j=2}^{n} tj + c_6 \sum_{j=2}^{n} (t_j - 1)$$

$$+ c_7 \sum_{j=2}^{n} (t_j - 1) + c_8(n-1)$$

Best Case: $t_j=1$

$$T(n) = an + b$$

Worst Case: $t_j=j$

$$T(n) = an^2 + bn + c$$

The divide-and-conquer approach

- ✓ **Divide** the problem into number of subproblem.
- ✓ **Conquer** the subproblems by solving them ***recursively***.
- ✓ **Combine** the solutions to subproblems into the solution for the original problem.

Merge Sort

- ✓ **Divide:** divide the n-element sequences to be sorted into two subsequences of $n/2$ elements each.
- ✓ **Conquer:** sort the two subsequences ***recursively*** using merge sort.
- ✓ **Combine:** merge the two sorted subsequences to produce the sorted answer.

Merging

```
MERGE(A,p,q,r)
    n_1 ← q – p + 1
    n_2 ← r – q
    Create array L[1..n_1+1] and R[1..n_2+1]
    For i ← 1 to n_1
        do L[i] ← A[p + i - 1]
    For j ← 1 to n_2
        do R[j] ← A[q + j]
    L[n_1+1] = ∞
    L[n_2+1] = ∞
    i ← 1
    j ← 1
    for k ← p to r
        do if L[i] ≤ R[j]
            then A[k] ← L[i]
                 i ← i + 1
            else A[k] ← R[j]
                 j ← j + 1
```

Merge Sort

MERGE-SORT(A,p,r)

1. If $p < r$
2. then $q \leftarrow \lfloor (p+r)/2 \rfloor$
3. MERGE-SORT(A,p,q)
4. MERGE-SORT(A,$q+1$,r)
5. MERAGE(A,p,q,r)

Analysis of Divide-and-Conquer

- **Divide** the problem into number of a subproblem each of which is $1/b$ size of the original problem $D(n)$.
- **Combine** the solutions to subproblems into the solution for the original problem $C(n)$.

$$T(n) = \begin{cases} \Theta(1) & \text{if } n \le c \\ aT(n/b) + D(n) + C(n) & \text{otherwise} \end{cases}$$

Analysis of Merge-Sort

- **Divide** the problem into number of *2* subproblems each of which is *1/2* size of the original problem

  $D(n) = \Theta(1)$.

- **Conquer** recursively solve two subproblems, each of size $n/2$, which contributes $2T(n/2)$ to the running time.

- **Combine** $C(n) = \Theta(n)$.

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 \\ 2T(n/2) + \Theta(n) & \text{if } n > 1 \end{cases}$$