Design & Analysis of Algorithms

(Divide & Conquer)

Soharab Hossain Shaikh
***BML Munjal* University**

General Steps in D&C

The divide and conquer paradigm consists of the following steps:

- The **divide** step: Break the problem into $p \geq 1$ subproblems that are similar to the original problem but smaller in size.
- The **conquer** step. Solve the subproblems by performing p recursive calls.
- The **combine** step: Solutions to these recursive calls are combined to obtain the solution to the original problem.

Remark: If the size of the problem is small enough, say less than or equal to a specific threshold value, we solve the problem using a straightforward method.

Divide & Conquer (cont..)

- Performance is very sensitive to changes in the steps.
- The threshold value for the size of the problem or subproblem should be chosen carefully.
- In the divide step, the number of partitions should be selected appropriately so that we can achieve the asymptotically-minimum running time.
- The combine step is very crucial to the performance of virtually all divide-and-conquer algorithms. Therefore, this combine step should be as efficient as possible.
- The divide step is similar in almost all divide-and-conquer algorithms. In many divide-and-conquer algorithms, it takes $O(n)$ time or even only $O(1)$ time.

MinMax Problem without D&C

The MinMax Problem

- Consider the problem of finding both the minimum and maximum elements in an array of integers $A[1..n]$. Assume that n is a power of 2.
- A **Straightforward Approach**:

  Algorithm **STRAIGHTFORWARDMinMax**

  Input: An array $A[1..n]$ of n integers, n is a power of 2.

  Output: (min, max): the minimum and maximum integers in A.

  1. $min \leftarrow A[1]$; $max \leftarrow A[1]$
  2. **for** $i \leftarrow 2$ to n
     3. **if** $A[i] <$ min **then** $min \leftarrow A[i]$
     4. **if** $A[i] >$ max **then** $max \leftarrow A[i]$
  5. **end for**
  6. **return** (min, max)

- Returns a pair (***min, max***) where ***min*** is the minimum and ***max*** is the maximum.
- No use of the divide and conquer approach in this algorithm.

Running time : MinMax

The MinMax Problem (Cont.)

Running Time Analysis of the Straightforward MinMax Algorithm:

- The number of element comparisons is used as a measure for the complexity of this algorithm.
- The number of *element* comparisons performed by this method is $2n-2$.

Algorithm **STRAIGHTFORWARDMinMax**

Input: An array $A[1..n]$ of n integers, n is a power of 2.

Output: $(min,\ max)$: the minimum and maximum integers in A.

1. $min \leftarrow A[1]$; $max \leftarrow A[1]$
2. **for** $i \leftarrow 2$ to n
3. **if** $A[i] <$ min **then** $min \leftarrow A[i]$
4. **if** $A[i] >$ max **then** $max \leftarrow A[i]$
5. **end for**
6. **return** $(min,\ max)$

MinMax: Improvement

The MinMax Problem (Cont.)

Improvement to the Straightforward Algorithm:

- The comparison $A[i] > max$ is only necessary when the comparison $A[i] < min$ is false.

- Replace the two if statements in the straightforward algorithm with the following statement:

  If $A[i] < min$ **then** $min \leftarrow A[i]$

  Else if $A[i] > max$ **then** $max \leftarrow A[i]$

- The **best case** occurs when the elements are in decreasing order. The number of element comparisons equals $n-1$.

- The **maximum** number of comparisons occurs when the elements are ordered in increasing order. The number of element comparisons is $2(n-1)$.

MinMax: D&C Approach

The MinMax Problem (Cont.)

DivideAndConquerMinMax Algorithm :

- Using the divide and conquer strategy, we can find both the minimum and the maximum in less number of comparisons.
- The idea is as follows:
  - ➢ Divide the input array of size n into two smaller parts:

$$A[1], A[2], \ldots, A[\left\lfloor \frac{n}{2} \right\rfloor]$$

and

$$A[\left\lfloor \frac{n}{2} \right\rfloor + 1], A[\left\lfloor \frac{n}{2} \right\rfloor + 2], \ldots, A[n]$$

  - ➢ Find the minimum and maximum in each part recursively.
  - ➢ Return the minimum of the two minima and the maximum of the two maxima.

MinMax: D&C Approach (cont..)

The MinMax Problem (Cont.)

Algorithm **DivideAndConquerMinMax**

Input An array $A[1..n]$ of n integers, n is a power of 2.

Output (min, max): the minimum and maximum integers in A.

1. $minmax\,(1, n)$

Procedure $minmax(low; high)$

1. **if** $high - low = 1$ then
2. **if** $A[low] < A[high]$ **then return** $(A[low], A[high])$
3. **else return** $(A[high], A[low])$
4. **end if**
5. **else**
6. $mid \leftarrow \lfloor (low + high)/2 \rfloor$
7. $(\min_1, \max_1) \leftarrow minmax\,(low, mid)$
8. $(\min_2, \max_2) \leftarrow minmax\,(mid + 1, high)$
9. $min \leftarrow \text{minimum}\{\min_1, \min_2\}$
10. $max \leftarrow \text{maximum}\{\max_1, \max_2\}$
11. **return** (min, max)
12. **end if**

MinMax (D&C) : Complexity

$T(n) = 1 \quad \text{for } n=2$

$T(n) = 2T(n/2)+2 \quad \text{for } n>2$

$= 3n/2 - 2$

$< 2n - 2$

[How? Solve the recurrence with iterative method.]