CSCI 262
Data Structures

22– Divide and Conquer
Mergesort

Divide and Conquer

• Split problem into multiple smaller sub-problems
• Solve the sub-problems recursively
• Recombine solutions afterwards
• When splitting/recombination can be done efficiently, this approach is a winner

Example

Search for a value in a sorted list.
Obvious approach:

```c
// find element k in sorted list x containing n elements
search(x, k)
for i = 1 to n
if x[i] == k return i
return NOTFOUND
```

Complexity: O(N)

A Better Way

Search for a value in a sorted list.

```c
// find element k in sorted list x containing n elements
binary_search(x, k)
if x is empty return NOTFOUND
pivot = n/2 // look at element halfway through list
if x[pivot] == k return pivot // if found, return
elseif k < x[pivot] // else search left or right sublist
    return binary_search(x[1 : pivot-1], k)
else
    return binary_search(x[pivot+1 : n], k)
```

Example – Binary Search

Search for a value in a sorted list.
Example: search for 11 in the list 1-15

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
pivot
9 10 11 12 13 14 15
11
```

Analysis of Binary Search

Compare with pivot
Return or choose new pivot

```
O(1)
```

N elements

```
O(1)
```

O(N/2) elements

```
O(1)
```

```
O(1)
```

Worst case: element not found

```
O(N/4)
```

```
O(1)
```

```
1
```

Complexity: # of times we split the list in two before getting to length 1 = \log_2 N
Mergesort

- Divide and Conquer algorithm for sorting
  - Split input list in half
  - Sort the halves
  - Merge the sorted lists

```
mmerge_sort(x)
    n = length(x)
    if n == 1 return x
    left = merge_sort(x[1 : n/2])
    right = merge_sort(x[n/2 + 1 : n])
    return merge(left, right)
```

Mergesort Illustrated

```
// treat a, b as stacks
x = empty list
loop
  if a is empty
    append b to x, return x
  else if b is empty
    append a to x, return x
  else if top(a) < top(b)
    append pop(a) to x
  else append pop(b) to x
return x
```

Analysis of Mergesort

```
N elements
Split = O(1)
Merge = O(N)
2 x Split = O(1)
2 x Merge = O(N/2)
```

About Logarithms

- $\log_b b^k = k$
- For any $b$, $\log_2 x = \log_b x / \log_b 2$

This means the base doesn't matter in Big O — all bases are just a constant factor from each other.

- Because $\log_2$ comes up so often, it is often abbreviated to $\log$