Merge Sort

Jonathan Windle

University of East Anglia

J.Windle@uea.ac.uk

June 4, 2017

Overview I

Divide and Conquer

Base case: Single element, return.

- **Divide:** Split the problem up into two or more non-overlapping sub-problems and solve there recursively.
- **Conquer:** Recombine the solutions of the sub-problems.

Merge Sort

1. Split array into two halves, left and right.
2. `mergeSort` left and right.
3. Merge the two sorted halves into a single sorted array.

```
array[] left, right, full
if low == high:
  return T[low]
mid = floor((low+high)/2)
// Divide and recursively solve....
left = mergeSort(T[low..mid])
right = mergeSort(T[mid+1...high])
// Conquer
full = merge(left,right)
return full
```

Summary

- Merge sort is a worst case and average case $O(nlogn)$
- Is stable
- Performs $O(nlogn)$ swaps in the worst case.
- Can be optimised to see if the merge is necessary, this makes it $O(n)$ in the best case.

The End