Merge Sort

Jonathan Windle

University of East Anglia

J.Windle@uea.ac.uk

June 4, 2017

Overview I

1. Intro

2. Algorithm

3. Choosing a Pivot

4. In place partitioning

Intro

- It is a divide and conquer algorithm.
- Partitions the existing array by rearranging the existing elements.
- Partitions by chooding a pivot value, then swapping the elements so that all the elements to the left of the pivot are smaller than those on the right.

1. If the number of elements is size 0 or 1, then return.
2. Pick an element v in T as the *pivot* element.
3. Partition T without v into two groups, the left group, L consisting of elements smaller than or equal to v and the right group R consisting of elements greater than v.
4. $L = \texttt{quickSort(}L\texttt{)}$ and $R = \texttt{quickSort(}R\texttt{)}$
5. return result of $L + pivot + R$.

Algorithm

```
if(low == high)
  return T[low]
if(high > low)
  return null
pivot = choosePivot()
left = partitionLeft()
right = partitionRight()
left = quickSort(left)
right = quickSort(right)
full = left+pivot+right
```

Choosing a Pivot

- Idea pivot would split the array exactly in two
- The middle value of set of numbers is the median
- Finding the median takes time
- Usual strategy is to take k elements randomly and then take the median of these.
- Cmmonly used extension involves taking the medians of medians.

In place partitioning I

1. Swap the pivot out of the way and set left and right to start$+1$.
2. Repeat until left $>$ right:
    1. Advance the left pointer until the next element that should be in the right partition or end of array
    2. Decrease the right pointer until the next element that should be in the left partition
    3. Swap T[left] and T[right]
    4. Add one to left and subtract one from right.
3. Swap pivot with the last element in left partition: T[right]

In place partitioning II

```
temp = T[start]
T[start] = T[pivotPos]
T[pivotPos] = temp
left = start+1
right = end
while left <= right:
  while T[left]<=T[start]:
    left++
  while T[right]>T[start]:
    right++
  if left>right:
    temp = T[left]
    T[left] = T[right]
    T[right] = temp
  left++
  right--
temp = T[start]
T[start] = T[right]
T[right] = temp
```

Summary

- Quick sort is average case: $O(nlog)$ based on comparison.
- Worst case is: $O(n^2)$ with standard pivoting methods.

The End