Mergesort and Quicksort

Lecture 07
CS 312

Mergesort

- Split the array in 1/2
- Sort each 1/2
- Merge the sorted halves
- Use insert sort when sub-arrays are small

Mergesort

procedure merge (U[1..m+1], V[1..n+1], T[1..m+n])
  i, j, := 1
  U[m+1], V[n+1] := sentinel
  for k := 1 to m+n do
    if U[i] < V[j]
      then T[k] := U[i]; i := i + 1
      else T[k] := V[j]; j := j + 1

Mergesort

procedure mergesort (T[1..n])
  if n is small enough then insertsort (T)
  else
    array U[1..floor(n/2)], V[1..ceil(n/2)]
    U[1..floor(n/2)] := T[1..floor(n/2)]
    V[1..ceil(n/2)] := T[1+floor(n/2)..ceil(n/2)]
    mergesort (U[1..floor(n/2)])
    mergesort (V[1..ceil(n/2)])
    merge (U, V, T)

Quicksort

- Invented by C.A.R. Hoare
- Pick a pivot element.
- Move everything smaller than the pivot to the left
- And everything else to the right
- Sort each side, then concatenate the results.

Quicksort

procedure Quicksort (T[i..j])
  if j - i is small enough, then insertsort (T[i..j])
  else
    pivot (T[i..j])
    quicksort (T[i..l])
    quicksort (T[l+1..j])
Picking a good pivot

• This is the crux of an implementation
• The median is the best choice.
  – sort the array into nondecreasing order, then the median is the middle element.
  – Why is the median the best choice?
  – Why isn’t the median used in practice?

C.A.R. Hoare

• Emeritus professor at Oxford
• Invented Hoare logic, Z notation and CSP
• Now works at... Microsoft Research
• Knighted March 7, 2000

Homework

• 7.14: Suppose each 1/2 of T is sorted. How do you merge these 2 halves without using another array?
• 7.17: Pivotbis sorts the array in 3 parts around pivot p: less than, equal to and greater than. Pivotbis is used in 7.5, but not explained.

Problem 7.14 Hints

Each 1/2 of T is sorted.

3 6 8 9 1 5 7 10

Your task is to combine each 1/2 using a fixed amount of new storage

Problem 7.17

Input:

2 4 1 3 8 2 5 5 4 8 6

pivot = 4

Output:

1 2 3 4 4 5 6 8 8 8

Complexity Analysis of DC

\[
t(n) = \begin{cases} 
\Theta(n^k) & \text{if } l < b^k \\
\Theta(n^k \log n) & \text{if } l = b^k \\
\Theta(n^{k \log_2 l}) & \text{if } l > b^k 
\end{cases}
\]

\(l = \text{number of subinstances}\)
\(n = \text{original instance size}\)
\(n/b = \text{size of subinstances}\)
\(k = \text{polynomial order of } g(n) = O(n^\ell)\)

where \(g(n) = \text{cost of doing the divide and recombination}\)