NPTEL MOOC,JAN-FEB 2015
Week 2, Module 6

DESIGN AND ANALYSIS OF ALGORITHMS

Merge sort: Analysis

MADHAVAN MUKUND, CHENNAI MATHEMATICAL INSTITUTE
http://www.cmi.ac.in/~madhavan

Merging sorted lists

Combine two sorted lists A and B into C

- If A is empty, copy B into C
- If B is empty, copy A into C
- Otherwise, compare first element of A and B and move the smaller of the two into C
- Repeat until all elements in A and B have been moved

Merging

```
function Merge(A,m,B,n,C)
   // Merge A[0..m-1], B[0..n-1] into C[0..m+n-1]

   i = 0; j = 0; k = 0;
   // Current positions in A,B,C respectively

   while (k < m+n)
      // Case 1: Move head of A into C
      if (j==n or A[i] <= B[j])
         C[k] = A[i]; i++; k++

      // Case 2: Move head of B into C
      if (i==m or A[i] > B[j])
         C[k] = B[j]; j++; k++
```

Analysis of Merge

How much time does Merge take?

- Merge A of size m, B of size n into C
- In each iteration, we add one element to C
  - At most 7 basic operations per iteration
  - Size of C is m+n
  - $m+n \lesssim 2 \max(m,n)$
- Hence $O(\max(m,n)) = O(n)$ if $m \approx n$

Merge Sort

To sort A[0..n-1] into B[0..n-1]

- If n is 1, nothing to be done
- Otherwise
  - Sort A[0..n/2-1] into L (left)
  - Sort A[n/2..n-1] into R (right)
  - Merge L and R into B

Analysis of Merge Sort …

- t(n): time taken by Merge Sort on input of size n
  - Assume, for simplicity, that $n = 2^k$
- t(n) = 2t(n/2) + n
  - Two subproblems of size n/2
  - Merging solutions requires time O(n/2+n/2) = O(n)
- Solve the recurrence by unwinding

Analysis of Merge Sort …

Analysis of Merge Sort …

- t(1) = 1

Analysis of Merge Sort …

- $t(1) = 1$
- $t(n) = 2t(n/2) + n$

Analysis of Merge Sort …

- $t(1) = 1$
- $t(n) = 2t(n/2) + n$

  $= 2\,[\,2t(n/4) + n/2\,] + n = 2^2\,t(n/2^2) + 2n$

Analysis of Merge Sort …

- $t(1) = 1$
- $t(n) = 2t(n/2) + n$

$$= 2 [2t(n/4) + n/2] + n = 2^2 t(n/2^2) + 2n$$

$$= 2^2 [2t(n/2^3) + n/2^2] + 2n = 2^3 t(n/2^3) + 3n$$

…

Analysis of Merge Sort ...

- $t(1) = 1$
- $t(n) = 2t(n/2) + n$

$= 2 [2t(n/4) + n/2] + n = 2^2 t(n/2^2) + 2n$

$= 2^2 [2t(n/2^3) + n/2^2] + 2n = 2^3 t(n/2^3) + 3n$

...

$= 2^j t(n/2^j) + jn$

Analysis of Merge Sort ...

- $t(1) = 1$
- $t(n) = 2t(n/2) + n$

  $= 2 [2t(n/4) + n/2] + n = 2^2 t(n/2^2) + 2n$

  $= 2^2 [2t(n/2^3) + n/2^2] + 2n = 2^3 t(n/2^3) + 3n$

  ...

  $= 2^j t(n/2^j) + jn$

- When $j = \log n$, $n/2^j = 1$, so $t(n/2^j) = 1$

Analysis of Merge Sort …

- $t(1) = 1$
- $t(n) = 2t(n/2) + n$

  $= 2\,[\,2t(n/4) + n/2\,] + n = 2^2\,t(n/2^2) + 2n$

  $= 2^2\,[\,2t(n/2^3) + n/2^2\,] + 2n = 2^3\,t(n/2^3) + 3n$

  …

  $= 2^j\,t(n/2^j) + jn$
- When $j = \log n$, $n/2^j = 1$, so $t(n/2^j) = 1$
  - $\log n$ means $\log_2 n$ unless otherwise specified!

Analysis of Merge Sort …

- $t(1) = 1$
- $t(n) = 2t(n/2) + n$

  $= 2\,[\,2t(n/4) + n/2\,] + n = 2^2\,t(n/2^2) + 2n$

  $= 2^2\,[\,2t(n/2^3) + n/2^2\,] + 2n = 2^3\,t(n/2^3) + 3n$

  …

  $= 2^j\,t(n/2^j) + jn$
- When $j = \log n$, $n/2^j = 1$, so $t(n/2^j) = 1$
  - $\log n$ means $\log_2 n$ unless otherwise specified!
- $t(n) = 2^j\,t(n/2^j) + jn = 2^{\log n} + (\log n)\,n = n + n \log n = O(n \log n)$

O(n log n) sorting

- Recall that O(n log n) is much more efficient than $O(n^2)$
- Assuming 10^8 operations per second, feasible input size goes from 10,000 to 10,000,000 (10 million or 1 crore)

Variations on merge

- Union of two sorted lists (discard duplicates)
  - If A[i] == B[j], copy A[i] to C[k] and increment i,j,k
- Intersection of two sorted lists
  - If A[i] < B[j], increment i
  - If B[j] < A[i], increment j
  - If A[i] == B[j], copy A[i] to C[k] and increment i,j,k
- Exercise: List difference: elements in A but not in B

Merge Sort: Shortcomings

- Merging A and B creates a new array C
  - No obvious way to efficiently merge in place
- Extra storage can be costly
- Inherently recursive
  - Recursive call and return are expensive

Alternative approach

- Extra space is required to merge
- Merging happens because elements in left half must move right and vice versa
- Can we divide so that everything to the left is smaller than everything to the right?
  - No need to merge!