Lect Note (1)

IIIT BHUBANESWAR

Divide and Conquer Algorithms: Merge Sort
January, 2018

Instructor: Ajaya Kumar Dash.

```
MERGESORT (A, p, r)
1  if p < r
2      q = ⌊(p+r)/2⌋
3      MERGESORT(A, p, q)
4      MERGESORT(A, q+1, r)
5      MERGE(A, p, q, r)
```

divide → q = ⌊(p+r)/2⌋
conquer → MERGESORT(A, p, q); MERGESORT(A, q+1, r)
combine → MERGE(A, p, q, r)

```
MERGE (A, p, q, r)
1   n1 = q − p + 1                                   → 1 time step
2   n2 = r − q                                       → 1 time step
3   Create two new temporary arrays L1[1 ... (n1 + 1)] and L2[1 ... (n2 + 1)]   → 1 time step
4   L1[1 ... n1] := A[p ... q]                       → ≈ n/2 - time step
5   L2[1 ... n2] := A[(q + 1) ... r]                 → ≈ n/2 - time step
6   L1[n1 + 1] = ∞                                   → 1 time step
7   L2[n2 + 1] = ∞                                   → 1 time step
8   i = 1                                            → 1 - time step
9   j = 1                                            → 1 - time step
10  for k = p to r                                   → (n) times
11      if L1[i] ≤ L2[j]                             } (3-time steps)
12          A[k] = L1[i]                             }
13          i = i + 1                                }
14      else                                         (or)
15          A[k] = L2[j]                             } (3 time steps)
16          j = j + 1                                }
```

Total time for merge procedure

$T_{merge}(n) = 1+1+1+\frac{n}{2}+\frac{n}{2}+1+1+1+1 + n(3)$

$= 4n + \text{Const}$

$\Rightarrow T_{merge}(n) = \Theta(n)$

Complexity Analysis of Mergesort (worst case)

$T(n)$ = Time for divide + Time for conquer + Time for merge

$= 1 + T(\frac{n}{2}) + T(\frac{n}{2}) + \Theta(n)$

$= 2T(\frac{n}{2}) + cn + 1$

$T(1) = 1$ ⟶ Base condition

using substitution method,

$$T(n) = 2T\left(\frac{n}{2}\right) + cn + 1$$

$$= 2\left[2T\left(\frac{n}{2^2}\right) + c\left(\frac{n}{2}\right) + 1\right] + cn + 1$$

$$= 2^2 T\left(\frac{n}{2^2}\right) + (cn + 2) + (cn + 1)$$

$$= 2^2\left[2T\left(\frac{n}{2^3}\right) + c\left(\frac{n}{2^2}\right) + 1\right] + (cn + 2) + (cn + 1)$$

$$= 2^3 T\left(\frac{n}{2^3}\right) + (cn + 2^2) + (cn + 2) + (cn + 1)$$

$$= 2^3 T\left(\frac{n}{2^3}\right) + 3cn + 2^2 + 2 + 1$$

$$\vdots$$

$$= 2^k T\left(\frac{n}{2^k}\right) + kcn + 2^{k-1} + \cdots + 2^2 + 2 + 1$$

put $\frac{n}{2^k} = 1 \Rightarrow 2^k = n \Rightarrow k = \log_2 n$

$$\Rightarrow T(n) = n\,T(1) + (\log_2 n)\,cn + \frac{2^k - 1}{2 - 1}$$

$$\Rightarrow T(n) = n\,T(1) + cn\log_2 n + (n - 1)$$

$$\Rightarrow T(n) = n + cn\log_2 n + (n - 1)$$

$$= 2n + cn\log_2 n - 1$$

$$\leq (c + 2)\, n\log_2 n \quad (c')$$

$\Rightarrow T(n)$ is $O(n\log_2 n)$

for $n \geq 1$; $2n - 1 > 0$

$\Rightarrow T(n) = cn\log_2 n + (2n - 1)$

$\geq cn\log_2 n$

$\Rightarrow T(n)$ is $\Omega(n\log_2 n)$

$\Rightarrow$ $T(n)$ is $\Theta(n\log_2 n)$