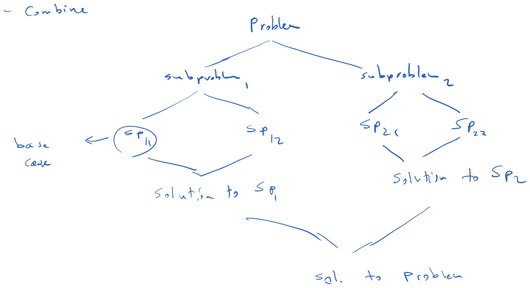
csi503-s21-lecture04

Tuesday, February 9, 2021

Divide-and-Conquer

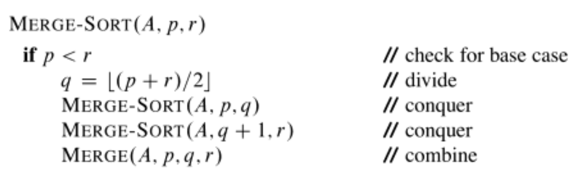
- divide → into subproblems
- Conquer : each subproblem (recursively)
  - base case, small enough, trivial to solve (brute force)
- Combine



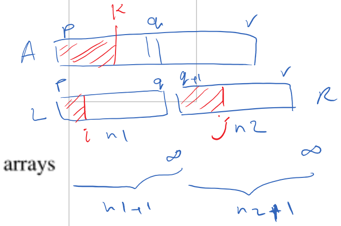
Example Merge-Sort:

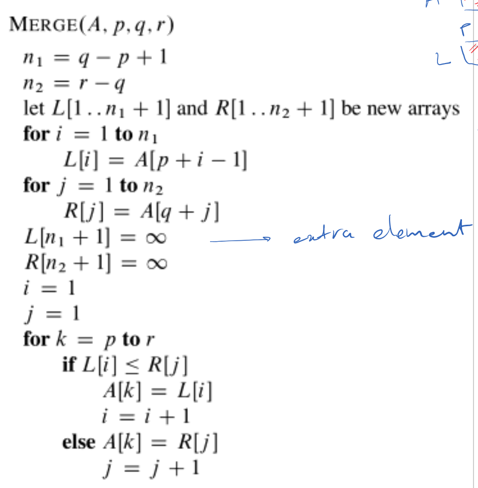
input: A[p..r]

- divide: find q half-way → A[p..q], A[q+1..r]
  sub problems
- base case: p==r (array has 1 element)
- Combine



```
MERGE-SORT(A, p, r)
  if $p < r$                          // check for base case
      $q = \lfloor (p+r)/2 \rfloor$               // divide
      MERGE-SORT(A, p, q)             // conquer
      MERGE-SORT(A, q+1, r)           // conquer
      MERGE(A, p, q, r)               // combine
```





```
MERGE(A, p, q, r)
  $n_1 = q - p + 1$
  $n_2 = r - q$
  let $L[1..n_1+1]$ and $R[1..n_2+1]$ be new arrays
  for $i = 1$ to $n_1$
      $L[i] = A[p+i-1]$
  for $j = 1$ to $n_2$
      $R[j] = A[q+j]$
  $L[n_1+1] = \infty$        → extra element (guard)
  $R[n_2+1] = \infty$
  $i = 1$
  $j = 1$
  for $k = p$ to $r$
      if $L[i] \le R[j]$
          $A[k] = L[i]$
          $i = i + 1$
      else $A[k] = R[j]$
          $j = j + 1$
```

loop invariant:

at start of each iteration A[p..k-1] contains $\underbrace{\text{smallest elements}}_{k-p}$ from L & R in sorted order

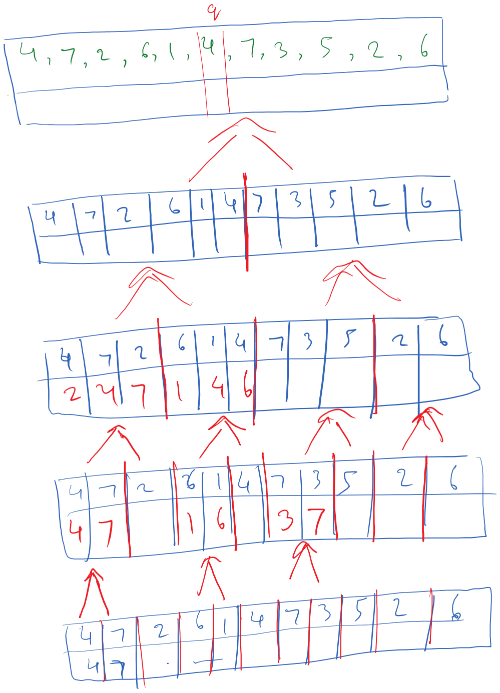
initialization: k = p ⇒ A[p..p-1] (empty array) is sorted

maintenance:

termination: k = r+1
i = n1+1
j = n2+1

Complexity of MERGE : $\Theta(n_1) + \Theta(n_2) + \Theta(n) = \Theta(n)$

example : 4, 7, 2, 6, 1, 4, 7, 3, 5, 2, 6



Merge-sort complexity $T(n)$

divide : $\Theta(1)$

conquer : 2 subproblems each $n/2 \longrightarrow 2T(n/2)$

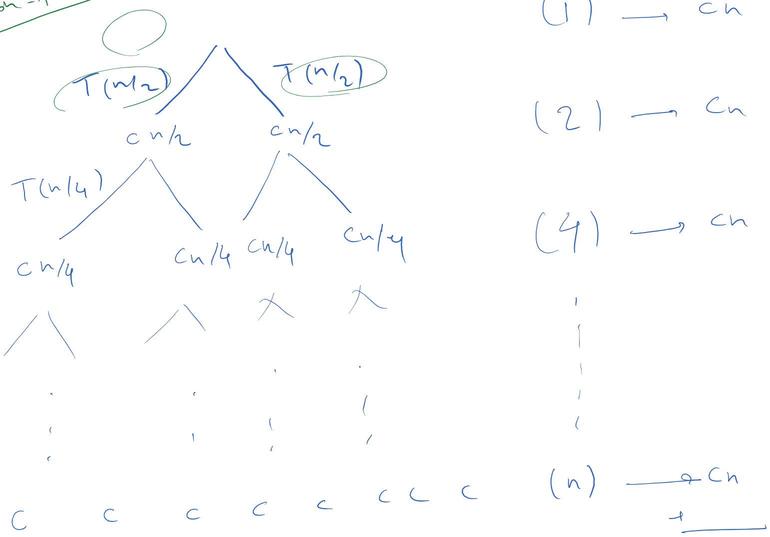
base-case : $\Theta(1)$

Combine : $\Theta(n)$

$$\Longrightarrow T(n) = \begin{cases} \Theta(1) & \text{if } n=1 \\ 2T(n/2) + \Theta(n) & n>1 \end{cases}$$ recurrence

$$T(n) = \begin{cases} c & n=1 \\ 2T(n/2) + cn & n>1 \end{cases}$$

recursion-tree



$cn \cdot (\lg n + 1) \Longrightarrow \Theta(n \lg n)$

substitution

$$T(n) = \begin{cases} & n=1 \\ 2T(n/2) + n & n>1 \end{cases}$$

guess: $n \lg n + n$

basis: $n=1 \longrightarrow 1 \cdot \lg 1 + 1 = 1$

induction step:

$T(n) = 2T(n/2) + n$ replace $T(n/2)$ based on guess

$= 2\left(n/2 \lg(n/2) + n/2\right) + n$

$= \left(n \lg(n/2) + n\right) + n$

$= \left(n \lg n - n \underbrace{\lg 2}_{1} + n\right) + n$

$= n \lg n - n + n + n$

$= \boxed{n \lg n + n}$

substitution ex. 2

$$T(n) = 2T(n/2) + \Theta(n)$$

upper-bound case: $T(n) \leq 2T(n/2) + cn$

guess: $T(n) \leq dn \lg n$

$T(n) \leq 2T(n/2) + cn$ replace guess for $n/2$

$\leq 2d\, n/2 \lg n/2 + cn$

$= dn \underbrace{\lg n/2}_{(\lg n - \lg 2)} + cn$

$= dn \lg n - dn + cn$

lower-bound case : $T(n) \geq 2T(n/2) + cn$

guess : $T(n) \geq dn \lg n$

replace $\longrightarrow$ $d \leq c$

substitution ex. 3

$$T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + 1$$

attempt 1 : guess : $T(n) \leq cn$

$$T(n) \leq \lfloor cn/2 \rfloor + \lceil cn/2 \rceil + 1$$

$$\leq cn + 1$$

$$\leq cn \quad ?$$

attempt 2 : guess : $T(n) \leq cn - d$

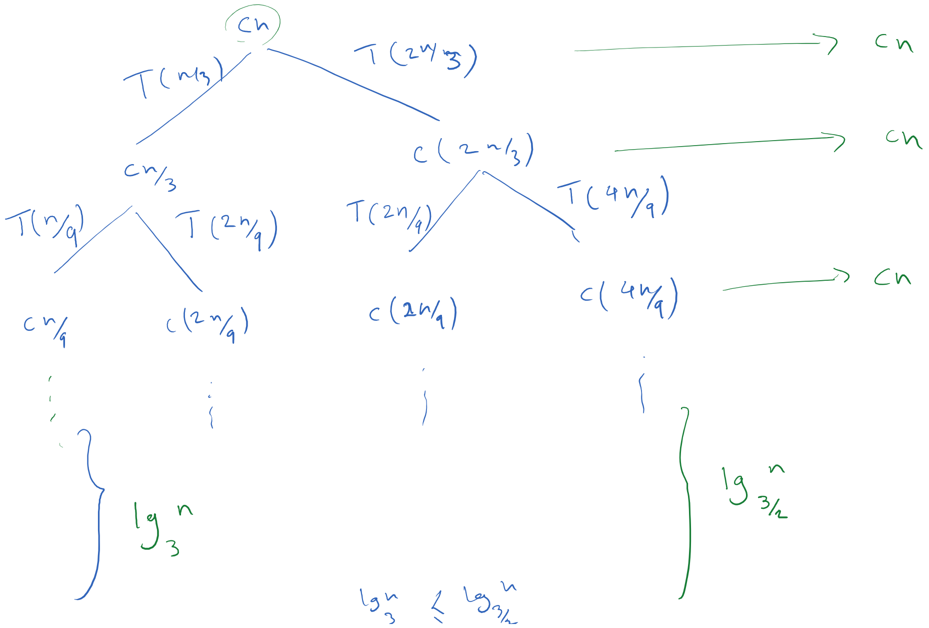
$$T(n) \leq c(n/2) - d + c(n/2) - d + 1$$

$$\leq cn - 2d + 1$$

$$\leq cn - d \longleftrightarrow -2d + 1 \leq -d \longleftrightarrow d \geq 1$$

example recursion-tree method

$$T(n) = T(n/3) + T(2n/3) + cn$$



$$\lg_3 n \leq \lg_{3/2} n$$

$) \quad {}_3 \qquad)$

$\lg_3 n \le \lg_{3/2} n$

upper-bound guess : $T(n) \le dn \lg_{3/2} n \longrightarrow O(n \lg n)$

lower-bound guess : $T(n) \ge dn \lg_3 n \longrightarrow \Omega(n \lg n)$

Now, replace and prove based on substitution. □

Master Method

$T(n) = aT(n/b) + f(n) \qquad a \ge 1, \; b > 1, \; \epsilon > 0$

case 1: $f(n) = O(n^{\lg_b a - \epsilon}) \longrightarrow T(n) = O(n^{\lg_b a})$

ex: $T(n) = 5T(n/2) + \Theta(n^2)$

$a = 5$
$b = 2$ $\longrightarrow n^{\lg_2 5} \ge n^2$

$\longrightarrow O(n^{\lg 5})$

case 2: $f(n) = \Theta(n^{\lg_b a}) \longrightarrow T(n) = \Theta(n^{\lg_b a} \cdot \lg n)$

case 3: $f(n) = \Omega(n^{\lg_b a + \epsilon})$, $af(n/b) \le cf(n)$ for some $c < 1$

$\Longrightarrow T(n) = \Theta(f(n))$

example:

$T(n) = T(2n/3) + 1$

$a = 1$

$b = \cancel{2/3}\ \ 3/2$

$\underbrace{n^{\lg_{3/2} 1}}_{n^0 = 1} \equiv$

$f(n) = 1$

$\longrightarrow T(n) = \Theta(1 \cdot \lg n)$

$= \Theta(\lg n)$

example : $T(n) = 2T(n/2) + n \lg n$

$a = 2$

$b = 2$

$n^{\lg_b a} = n^{\lg 2} = n$

$\longrightarrow T(n) = \Theta(n \lg^2 n)$

reg. constraint : $2 \, n/2 \lg n/2 \leq c \, n \lg n$

$\boxed{\text{not valid}}$

$f(n) = n \lg n$

$\cancel{n}(\lg n - \lg 2) \leq c \cancel{n} \lg n$

$\lg n - 1 \leq c \lg n$

$\frac{\lg n - 1}{\lg n} \leq c$

$1 - \frac{1}{\lg n} \leq c < 1$

cannot choose constant c

example, $T(n) = 5T(n/2) + \Theta(n^3)$

$a = 5$

$b = 2$

$\underbrace{n^{\lg_2 5}}_{n^{2\dots}} < n^3 \longrightarrow \Theta(n^3)$

$\boxed{\text{Case 3}}$