Divide and Conquer dan Recurrence

Wijayanti N Khotimah, M.Sc.

Latar Belakang

- ✓ Pada insertion sort, problem sorting diselesaikan dengan pendekatan incremental.
- ✓ Alternatif desain lagoritma yang lain adalah dengan pendekatan “devide and conquer”

Divide and Conquer

- **Divide**
  - Membagi permasalahan menjadi beberapa sub-permasalahan.
- **Conquer**
  - Sub-permasalahan diselesaikan secara rekursif
  - ***Base case:*** Jika sub-permasalahan cukup kecil, selesaikan dengan brute force.
- **Combine**
  - Gabungkan penyelesaian/solusi sub-sub permasalahan menjadi penyelesaian/solusi permasalahan awal.

Divide and Conquer pada MergeSort

- Cara kerja merge sort pada array *A[p . . r]:*
  - **Divide**
    - ✓Membagi array A[p..r] menjadi dua subarray A[p . . q] dan A[q + 1 . . r], dengan nilai q adalah nilai tengah antara p dan r.
  - **Conquer**
    - ✓Secara rekursif urutkan dua subarray A[p . . q] dan A[q + 1 . . r].
  - **Combine**
    - ✓Gabungkan dua subarray yang sudah terurut untuk menghasilkan sebuah array terurut. Nantinya akan dibuat sebuah prosedur MERGE(A, p, q, r).
  - Proses rekursi berhenti ketika subarray hanya memiliki 1 elemen, karena pasti sudah terurut.

Kerangka Umum Algoritma Merge Sort

```
MERGE-SORT(A, p, r)
if p < r
    Q = floor((p + r)/2)
    MERGE-SORT(A, p, q)
    MERGE-SORT(A, q + 1, r)
    MERGE(A, p, q, r)
```

Proses mana yang menjadi key operation?

Key: proses mana yang membutuhkan waktu paling banyak?

Jawab: Proses untuk mengkombinasikan dua array secara urut (berada dalam fungsi MERGE)

Pseudocode Merge

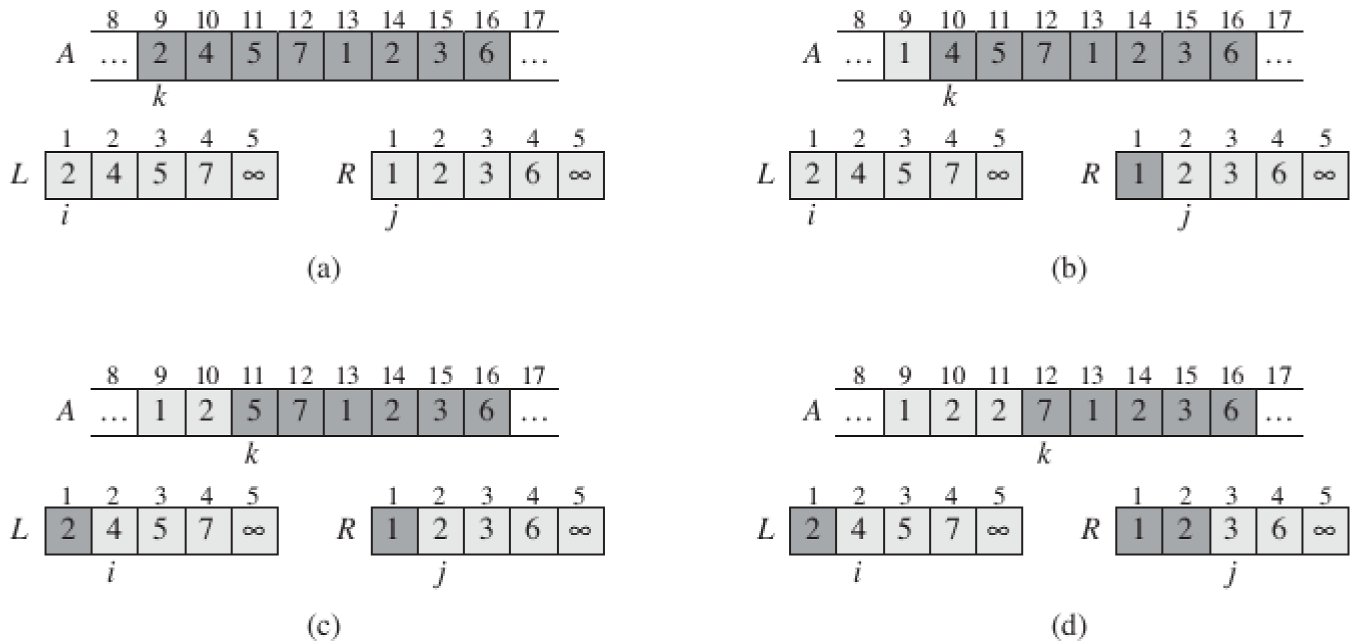
```
MERGE(A, p, q, r)
n_1 = q - p + 1
n_2 = r - q
let L[1 .. n_1 + 1] and R[1 .. n_2 + 1] be new arrays
for i = 1 to n_1
    L[i] = A[p + i - 1]
for j = 1 to n_2
    R[j] = A[q + j]
L[n_1 + 1] = ∞
R[n_2 + 1] = ∞
i = 1
j = 1
for k = p to r
    if L[i] ≤ R[j]
        A[k] = L[i]
        i = i + 1
    else A[k] = R[j]
        j = j + 1
```

- ✓ Proses merge, seperti proses penggabungan 2 buah kumpulan kartu yang sudah urut.
- ✓ Kita tidak perlu membandingkan seluruh kartu tetapi hanya membandingkan kartu terkecil pada masing-masing kelompok.
- ✓ Kompleksitas proses MERGE adalah Θ (n)
- ✓ n=r-p+1

Ilustrasi Proses Merge

Setelah baris ke-9 selesai diproses

Hasil proses pada baris 13-17



Analisa Divide and Conquer

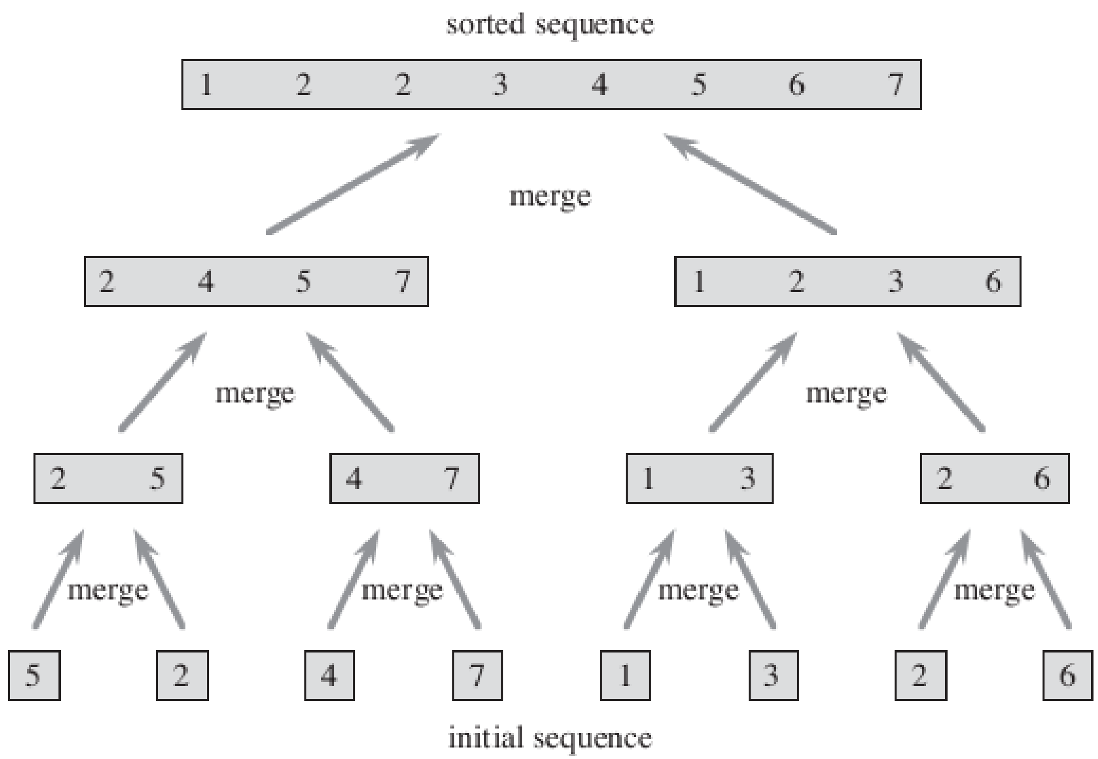
- Ketika suatu algoritma mengandung rekursif (memanggil dirinya sendiri), kita bisa mendeskripsikan running timenya dengan ***recurrence equation*** atau ***recurrence***.
- *Recurrence equation* mendeskripsikan seluruh running time dari suatu problem dengan ukuran *n* dalam bentuk *running time* dari input yang lebih kecil.
- Rekuren adalah fungsi yang didefinisikan dalam bentuk:
  - Satu atau lebih *base case*, dan
  - Dalam bentuk dirinya sendiri dengan argumen yang lebih kecil.

Recurrence dalam Divide and Conquer

- Running time untuk problem dengan ukuran *n* adalah *T(n)*.
- Jika problemnya sudah sangat kecil, *n*<*c* untuk *c* konstan, maka running timenya tetap yaitu *Θ(1)*.
- Misal pembagian problem menghasilkan ***a*** buah sub problem dengan masing-masing sub problem berukuran *1/b* ukuran aslinya, maka running time untuk setiap sub problemnya adalah ***T(n/b)*** sehingga butuh waktu ***aT(n/b)*** untuk menyelesaikan seluruh sub problem tersebut.
- Jika misal running time untuk membagi problem ke sub problem adalah ***D(n)*** dan running time untuk mengkombinasikan solusi dari seluruh sub problem adalah ***C(n)*** maka didapatkan recurrence dari divide and conquer:

$$T(n) = \begin{cases} \Theta(1) & \text{if } n \le c\ , \\ aT(n/b) + D(n) + C(n) & \text{otherwise}\ . \end{cases}$$

Ilustrasi Proses Merge dalam Mergeshort



Recurrence dari Mergsort

- Promblem terkecil berukuran 1 sehingga didapatkan running timenya ketika n=1 adalah Θ(1)
- Setiap problem dibagi menhadi 2 sub problem dengan masing-masing sub problem berukuran (n/2) dalam hal ini a=2 dan b=2
- Waktu untuk membagi problem menjadi sub problem adalah konstan yaitu D(n)= Θ(1)
- Waktu untuk mengkombinasikan solusi (proses merge) adalah C(n)=Θ(n) → **sudah dibahas sebelumya.**
- Sehingga recurrence untuk merge sort:

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 \\ 2T\left({}^{n}/_{2}\right) + \Theta(1) + \Theta(n) & \text{if } n > 1 \end{cases}$$

Disederhanakan

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

Contoh Lain dari Recurrence

$$s(n) = \begin{cases} 0 & n = 0 \\ c + s(n-1) & n > 0 \end{cases}$$

$$s(n) = \begin{cases} 0 & n = 0 \\ n + s(n-1) & n > 0 \end{cases}$$

$$T(n) = \begin{cases} c & n = 1 \\ 2T\left(\frac{n}{2}\right) + c & n > 1 \end{cases}$$

$$T(n) = \begin{cases} c & n = 1 \\ aT\left(\frac{n}{b}\right) + cn & n > 1 \end{cases}$$