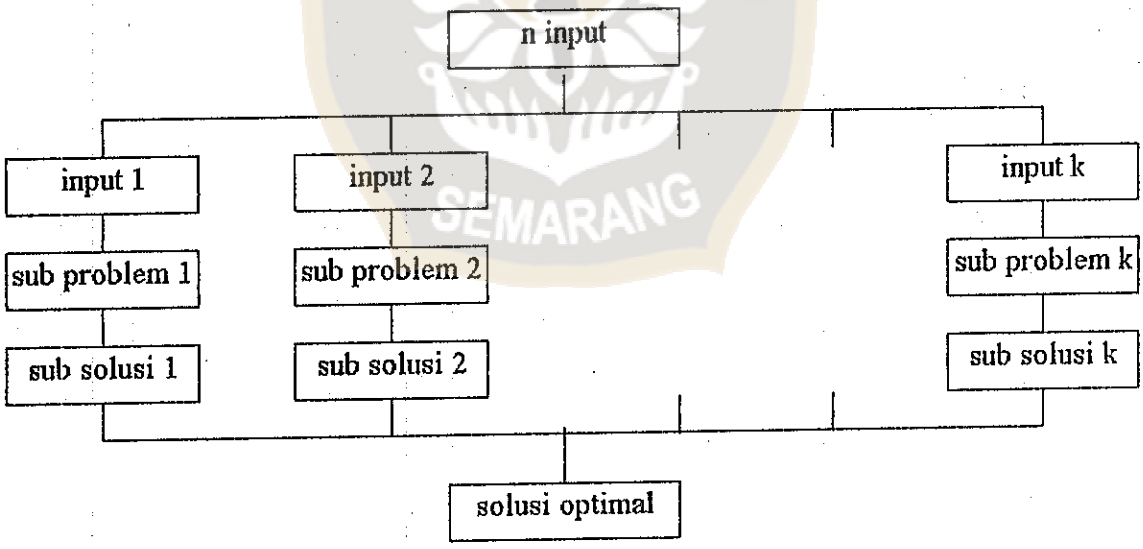
BAB III

ANALISA ALGORITMA MERGESORT

MENGGUNAKAN TEKNIK DIVIDE AND CONQUER (DANDC)

3.1 Teknik Divide and Conquer (DANDC)

Prinsip dasar dari metode ini adalah dengan membagi n input menjadi k sub set input yang berbeda ($1 < k \leq n$) dengan ukuran yang sama. Dari k sub set input yang berbeda akan terdapat k sub problem. Setiap sub problem mempunyai solusi, sehingga akan diperoleh k sub solusi. Selanjutnya dari k sub solusi dikombinasi sehingga diperoleh solusi yang optimal. Bentuk umum teknik divide and conquer adalah sebagai berikut :



gambar 3.1
Bentuk Umum metode DANDC

Jika sub problem ukurannya masih relatif besar (ukuran > 1), maka teknik DANDC dapat digunakan lagi secara rekursif sedemikian sehingga setiap sub problem mempunyai satu elemen. Adapun algoritma teknik DANDC adalah sebagai berikut :

```
Procedure A(D : Data ; Var R : Result ) ;
Var     D1, ..., Dk : Data ;
        R1, ..., Rk : Result ;
Begin
 If simple (D) then R := A0(D)
                Else
                 Begin
                   D1, ..., Dk := partition(D) ;
                   R1 := A(D1) ; ... Rk := A(Dk) ;
                   R := Combine (R1, ..., Rk) ;
                 End
End.
```

Keterangan

- Simple (D) adalah fungsi bernilai boole yang menentukan apakah data (D) cukup kecil sedemikian sehingga solusinya dapat dihitung tanpa pemecahan. Jika demikian maka R := A_0(D) yang akan diproses atau dieksekusi. Pada keadaan lain, D_1, ..., D_k := partition(D) yang akan dieksekusi.
- Fungsi partition(D) mempartisi data (D) menjadi D_1, ..., D_k. Untuk setiap i, A(D_i) merupakan hasil R_i- Fungsi Combine(R_1, ..., R_k) merupakan fungsi yang menentukan solusi umum atau solusi yang diharapkan dari persoalan semula dengan memanfaatkan sub solusi R_1, ..., R_k yang hasilnya adalah R.

Suatu kelas dari metode DANDC memenuhi syarat - syarat berikut :

1. Untuk masalah dengan ukuran n = 1, nilai pemecahan masalah atau running timenya adalah c, dimana c adalah konstanta positif.
2. Masalah - masalah dengan ukuran $n = b^m$, untuk $b > 1$, $m > 0$ dibagi menjadi a sub masalah yang berbeda dengan ukuran n/b.
3. Untuk masalah dengan ukuran $n > 1$, nilai pemecahan masalah atau running timenya adalah hasil penjumlahan dari nilai pembagian masalah menjadi sub masalah dengan nilai penggabungan solusi - solusi dari sub masalah menghasilkan suatu solusi dari masalah semula (h(n)).

Berdasarkan kondisi - kondisi di atas, maka dihasilkan suatu persamaan rekurensi sebagai berikut :

$$T(n) = \begin{cases} c & , \text{untuk } n = 1 \\ a\,T(n/b) + h(n) & , \text{untuk } n > 1 \end{cases} \qquad (3.1)$$

Teorema 3.1.1

Solusi persamaan $T(n) = a\,T(n/b) + \theta(n^k)$, untuk $a \geq 1$ dan $b > 1$ adalah

$T(n) = O(n^k \log n)$ jika $a = b^k$

Bukti

Misalkan $n = b^m$, maka $n/b = b^{m-1}$

$$n^k = (b^m)^k = b^{mk} = b^{km} = (b^k)^m$$

Asumsikan $T(1) = 1$. Dengan menggunakan definisi 2.6.1.3 maka persamaan $T(n) = a\,T(n/b) + \theta(n^k)$ dapat ditulis

$T(n) = a\,T(n/b) + c\,n^k$. Misalkan $c = 1$, maka didapatkan

$T(n) = a\,T(n/b) + n^k$

$$T(b^m) = a\,T(b^{m-1}) + (b^k)^m \qquad (3.2)$$

Jika setiap sisi dibagi dengan a^m, maka didapatkan

$$\frac{T(b^m)}{a^m} = \frac{T(b^{m-1})}{a^{m-1}} + \left[\frac{b^k}{a}\right]^m \qquad (3.3)$$

sehingga persamaan tersebut dapat diaplikasikan untuk nilai m yang lain sebagai berikut :

$$\frac{T(b^{m-1})}{a^{m-1}} = \frac{T(b^{m-2})}{a^{m-2}} + \left[\frac{b^k}{a}\right]^{m-1} \qquad (3.4)$$

$$\frac{T(b^{m-2})}{a^{m-2}} = \frac{T(b^{m-3})}{a^{m-3}} + \left[\frac{b^k}{a}\right]^{m-2} \qquad (3.5)$$

$$\vdots$$

$$\frac{T(b^1)}{a^1} = \frac{T(b^0)}{a^0} + \left[\frac{b^k}{a}\right]^1 \qquad (3.6)$$

Dengan menggunakan persamaan (3.2) sampai (3.6), maka dihasilkan :

$$\frac{T(b^m)}{a^m} = 1 + \sum_{i=1}^{m} \left[\frac{b^k}{a} \right]^i$$

$$= \sum_{i=0}^{m} \left[\frac{b^k}{a} \right]^i \qquad (3.7)$$

Dengan demikian

$$T(n) = a^m \sum_{i=0}^{m} \left[\frac{b^k}{a} \right]^i \text{, untuk } n = b^m \qquad (3.8)$$

Jika $a = b^k$, maka setiap suku dalam jumlahan adalah 1. Dengan demikian persamaan (3.8) dapat ditulis

$T(n) = b^{km} (1 + m)$

$T(n) = n^k (1 + \log_b n)$

$= n^k + n^k \log_b n$

$= O(n^k \log_b n)$

$= O(n^k \log n)$

Jadi terbukti bahwa solusi persamaan $T(n) = aT(n/b) + \theta(n^k)$, untuk $a \geq 1$ dan $b > 1$ adalah $T(n) = O(n^k \log n)$, jika $a = b^k$

■

3.2 Algoritma Mergesort Menggunakan Teknik DANDC

Misalkan terdapat suatu deret dengan n elemen yang ditempatkan dalam suatu array A. Deret tersebut akan diurutkan secara ascending. Dengan menggunakan algoritma mergesort dengan teknik DANDC deret tersebut dibagi menjadi dua bagian yaitu A(1), A(2), ..., A(n/2) dan A(n/2 + 1), A(n/2 + 2), ..., A(n). Jika pembagian deret tersebut hasilnya masih terlalu besar (ukuran setiap bagiannya > 1), maka masing-masing bagian dibagi lagi menjadi dua bagian yang lebih kecil, sedemikian sehingga setiap bagiannya mempunyai satu elemen. Selanjutnya setiap pasang bagian dikombinasi sehingga akan dihasilkan urutan yang tunggal dari n elemen semula.

Algoritma Mergesort dengan teknik DANDC terdiri dari dua prosedur yaitu Prosedur MERGESORT dan Prosedur MERGE. Algoritmanya adalah sebagai berikut :

Masukan : s_i, ..., s_j, i, dan j
Keluaran : s_i, ..., s_j teratur dalam urutan menaik.

```
1.  Procedure MERGESORT (s, i, j, p)
2.  // Kasus dasar i = j //
3.  If  i = j  then
4.          p := i
5.  // Bagilah deret dan urutkan //
6.  m := (i + j) div 2
7.  Call  MERGESORT (s, i, m, q)
8.  Call  MERGESORT (s, m + 1, j, r)
9.  // Gabungkan //
10. Call  MERGE (s, i, m, j, c)
11. End MERGESORT.
```

Masukan : Dua deret menaik s_i, ..., s_m dan s_{m+1}, ..., s_j dengan indeks i, m, j.

Keluaran : Deret s_i, ..., s_j dalam urutan yang menaik.

```
1.  Procedure MERGE (s, i, m, j, c)
2.  // p adalah posisi di deret s_i, ..., s_m. //
3.  // q adalah posisi di deret s_{m+1}, ..., s_j. //
4.  // r adalah posisi di deret c_i, ...,s_j. //
5.  p := i
6.  q := m + 1
7.  r := i
8.  // Salin nilai yang lebih kecil dari s_p dan s_q. //
9.  While p ≤ m dan q ≤ j  do
10. Begin
11.   If s_p < s_q  then
12.         Begin
13.           c_r := s_p
14.           p := p + 1
15.         End
16.   Else
17          Begin
18.           c_r :=  s_q
19.           q := q + 1
20.         End
21.   r := r + 1
22. End
23  // Salin sisa dari deret pertama //
24. While p ≤ m  do
```

```
25.  Begin
26.     c_r := s_p
27.     p := p + 1
28.     r := r + 1
29.  End
30.  // Salin sisa dari deret kedua //
31.  While q ≤ j  do
32.  Begin
33.     c_r := s_q
34.     q := q + 1
35.     r := r + 1
36.  End
37.  // Salinlah c ke dalam s //
38.  For k := i to j  do
39.           s_k := c_k
40.  End MERGE.
```

Teorema 3.2.1

Dua daftar yang telah urut dengan panjang m dan n dapat digabung (dimerge) menjadi satu daftar yang urut memerlukan kurang dari atau sama dengan m + n - 1 perbandingan.

Bukti

Misalkan list 1 dan list 2 adalah dua daftar dengan panjang m dan n, dan keduanya telah terurut secara naik (ascending). Dengan menggunakan algoritma MERGE, masukan - masukan pada bagian awal dari daftar list 1 dan list 2 dibandingkan dan

nilai yang lebih kecil dari kedua masukan tersebut diletakkan pada suatu list yang kosong. Proses ini diulang sampai salah satu dari kedua daftar kosong, dan selanjutnya daftar yang lain (tidak kosong), masukan - masukannya diletakkan pada urutan yang terakhir sampai daftar tersebut kosong. Karena setiap perbandingan dari elemen - elemen list 1 dan elemen - elemen list 2 menghasilkan suatu elemen yang diletakkan pada suatu list yang lain, maka terdapat kurang dari m + n perbandingan. Karena tidak ada perbandingan ketika salah satu list kosong, maka paling banyak terdapat m + n - 1 perbandingan. Dengan demikian algoritma MERGE akan menggabungkan dua daftar yang telah terurut menjadi satu daftar terurut memerlukan kurang dari atau sama dengan m + n - 1 perbandingan diantara elemen - elemen dari kedua list tersebut.

■

3.3 Analisa Algoritma Mergesort Menggunakan Teknik DANDC

Algoritma mergesort terdiri dari dua prosedur yaitu Prosedur MERGESORT dan Prosedur MERGE. Analisa algoritma mergesort menggunakan teknik DANDC dilakukan hanya pada Prosedur MERGESORT. Berdasarkan aturan umum analisa algoritma dan analisa struktur kontrol maka analisa algoritma mergesort dapat diuraikan sebagai berikut :

1. Pada baris ke-1 merupakan nama dari prosedur. Karena hanya berupa nama prosedur, maka baris tersebut tidak mempengaruhi analisa.

2. Pada baris ke-2 terdapat suatu komentar yang menyatakan suatu pengujian apakah i = j. Karena hanya berupa komentar, maka baris tersebut tidak berpengaruh terhadap analisa.
3. Pada baris ke-3 terdapat statemen kondisi if, sehingga running timenya adalah harga pada kondisi statemen yang dieksekusi. Jika statemen kontrol terpenuhi, maka running timenya adalah running time statemen atau blok statemen sesudah then (prosedur akan mengeksekusi baris ke-4) dan jika statemen kontrol tidak terpenuhi, maka running timenya adalah running time pada pengujian statemen kontrol yaitu O(1).
4. Pada baris ke-4 terdapat suatu statemen pemberian yaitu p := i yang dieksekusi ketika daftar mempunyai panjang 1 atau i = j, Sehingga running timenya adalah O(1).
5. Pada baris ke-5 terdapat suatu komentar yang menyatakan deret dibagi dan disortir. Karena hanya berupa komentar, maka hal ini tidak mempengaruhi analisa.
6. Pada baris ke-6 terdapat statemen pemberian, yaitu pemberian harga pada suatu variabel m = (i + j) div 2. Karena berupa statemen pemberian, maka running timenya adalah O(1).
7. Pada baris ke-7 dan ke-8 terdapat pemanggilan Prosedur MERGESORT secara rekursif yaitu MERGESORT (s, i, m, q) dan MERGESORT (s, m+1, j, r). Karena kedua Prosedur MERGESORT masing - masing mempunyai daftar dengan panjang n/2, maka running time untuk setiap pemanggilan prosedur tersebut adalah T(n/2). Dengan demikian running time pemanggilan secara rekursif kedua Prosedur MERGESORT tersebut adalah 2 T(n/2).

8. Pada baris ke-9 terdapat suatu komentar gabungkan. Karena hanya berupa komentar maka tidak berpengaruh terhadap analisa.
9. Pada baris ke-10 terdapat pemanggilan Prosedur MERGE (s, i, m, j, c). Karena terdapat dua sub daftar dengan panjang masing - masing n/2, maka berdasarkan teorema 3.2.1 dibutuhkan paling banyak n/2 + n/2 + 1 = n - 1 perbandingan, sehingga running time untuk mengeksekusi Prosedur MERGE adalah O(n).
10. Pada baris ke-11 merupakan akhir dari prosedur MERGESORT dengan waktu O(1).

Misalkan T(n) adalah worst case running time prosedur MERGESORT. Dengan demikian untuk beberapa konstanta c_1 dan c_2 dapat dituliskan persamaan rekurensi sebagai berikut :

$$T(n) = \begin{cases} c_1 & , \text{jika } n = 1 \\ 2\,T(n/2) + c_2 n & , \text{jika } n > 1 \end{cases} \qquad (3.9)$$

Keterangan

Suku c_1 pada persamaan (3.9) merupakan suatu konstanta dari langkah - langkah yang ditempuh ketika daftar mempunyai panjang 1. Pada kasus n > 1, waktu yang ditempuh oleh Prosedur MERGESORT dapat dibagi menjadi dua bagian yaitu

1. Pemecahan daftar menjadi dua bagian yang sama yang memerlukan waktu O(1) dan pemanggilan secara rekursif dua Prosedur MERGESORT dengan panjang daftar masing-masing n/2 dengan waktu tempuh 2 T(n/2), sehingga berdasarkan aturan penjumlahan waktu tempuh kedua proses tersebut adalah 2 T(n/2).

2. Pemanggilan Prosedur MERGE yang memerlukan waktu O(n).

Misalkan waktu yang dibutuhkan untuk mengeksekusi Prosedur MERGE c_2n. Dengan berdasarkan aturan penjumlahan (rule of sums), maka waktu yang ditempuh oleh Prosedur MERGESORT untuk $n > 1$ adalah $2\,T(n/2) + c_2n$.

Persamaan rekurensi (3.9) dapat diselesaikan dengan dua cara yaitu menggunakan metode pemecahan persamaan rekurensi secara iteratif atau menggunakan teorema 3.1.1.

1. Pemecahan persamaan rekurensi secara iteratif.

Misalkan untuk $n = 1$, maka $c_1 = T(n) = T(1) = 1$

Untuk $n > 1$, maka $T(n) = 2\,T(n/2) + c_2n$. Dengan memisalkan $c_2 = 1$, maka didapatkan

$$T(n) = 2\,T(n/2) + n \qquad (3.10)$$

Dengan mengganti n dengan n/2, maka persamaan (3.10) menjadi :

$$T(n/2) = 2\,T(n/4) + n/2$$

Jika masing - masing ruas dikalikan dengan dua, maka didapatkan :

$$2\,T(n/2) = 2\,(\,2\,T(n/4) + n/2\,) = 4\,T(n/4) + n$$

Dengan demikian persamaan (3.10) dapat ditulis kembali sebagai berikut :

$$T(n) = 4\,T(n/4) + 2\,n \qquad (3.11)$$

Dengan mengganti n dengan n/4, maka persamaan (3.10) dapat ditulis :

$$T(n/4) = 2\,T(n/8) + n/4$$

Jika masing - masing ruas dikalikan dengan 4, maka didapatkan :

$$4\,T(n/4) = 4\,(\,2\,T(n/8) + n/4\,)$$

$$= 8\,T(n/8) + n$$

Dengan demikian persamaan (3.11) dapat ditulis kembali sebagai berikut :

$T(n) = 8\ T(n/8) + n + 2n$

$= 8\ T(n/8) + 3n$

Jika langkah tersebut diteruskan, maka akan diperoleh bentuk umum :

$T(n) = 2^k\ T(n/2^k) + k\ n \quad , k = 1, 2, 3, \ldots$ (3.12)

Dengan memisalkan $k = \log_2 n$, maka persamaan (3.12) dapat ditulis :

$T(n) = 2^{\log_2 n}\ T(n / 2^{\log_2 n}) + \log_2 n\ .\ n$

$= n\ T(n/n) + n \log_2 n$

$= n\ T(1) + n \log_2 n$

$= n + n \log_2 n$

Dengan demikian berdasarkan aturan penjumlahan, running time Prosedur MERGESORT menggunakan teknik Divide and Conquer adalah O(n log n).

2. Menggunakan teorema 3.1.1

Misalkan untuk $n = 1$, maka $c_1 = T(n) = T(1) = 1$

Untuk $n > 1$, maka $T(n) = 2\ T(n/2) + c_2 n$.

Misalkan $c_2 n = \theta(n)$, sedemikian sehingga didapatkan

$T(n) = 2\ T(n/2) + \theta(n)$, untuk $n > 1$

Berdasarkan teorema 3.1.1 didapatkan $a = b = 2$, $k = 1$ sehingga memenuhi syarat $a = b^k$. Dengan demikian $T(n) = O(n \log n)$.