**LAPORAN PRAKTIKUM**

**STRUKTUR DATA DAN ALGORITMA II**

**Divide And Conquer**

****

**DISUSUN OLEH :**

**IMAM NAJIBULLOH HR M0516027**

**ASISTEN :**

1. **ARIEF NUR HUDA M0514010**
2. **KHOIRUL FIKRI M0514028**
3. **KELVIN HERWANDA T. M0514027**
4. **SALSABILA FAUZIA S M0515036**

**PROGRAM STUDI INFORMATIKA**

**FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM**

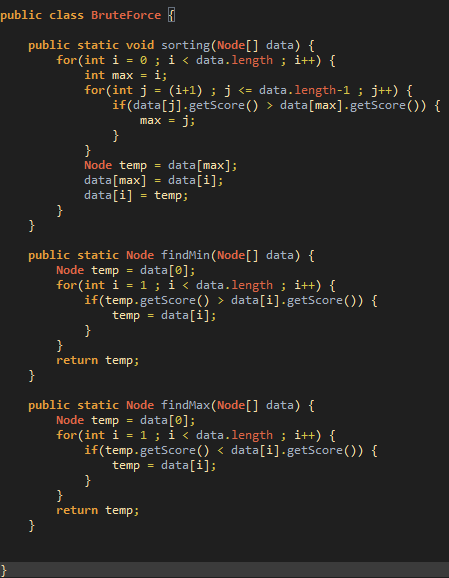
**UNIVERSITAS SEBELAS MARET**

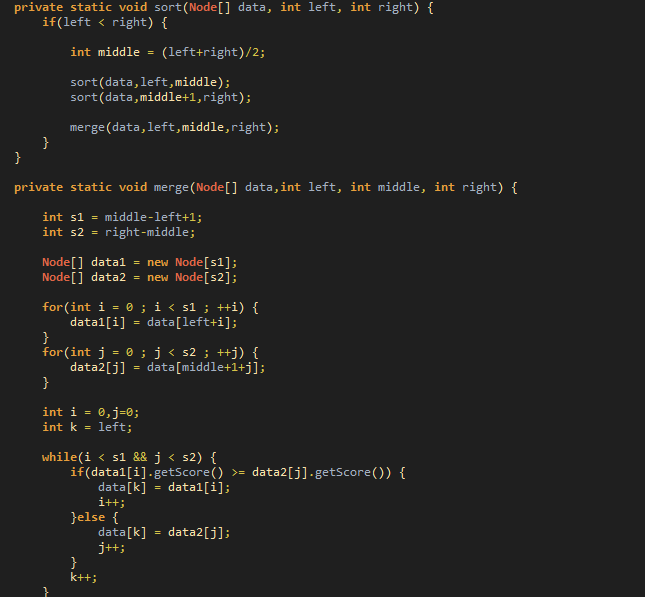
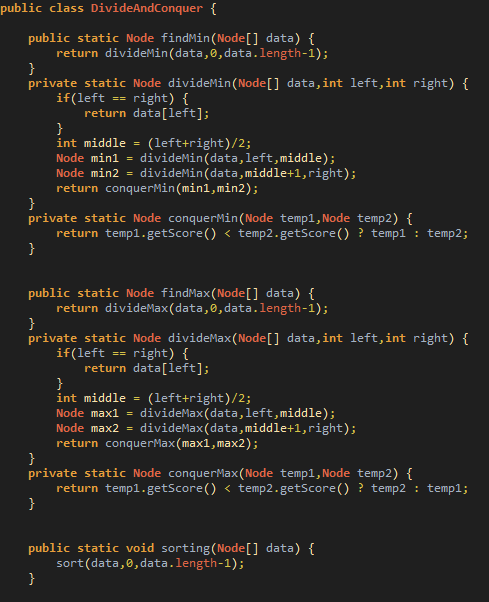
**SURAKARTA**

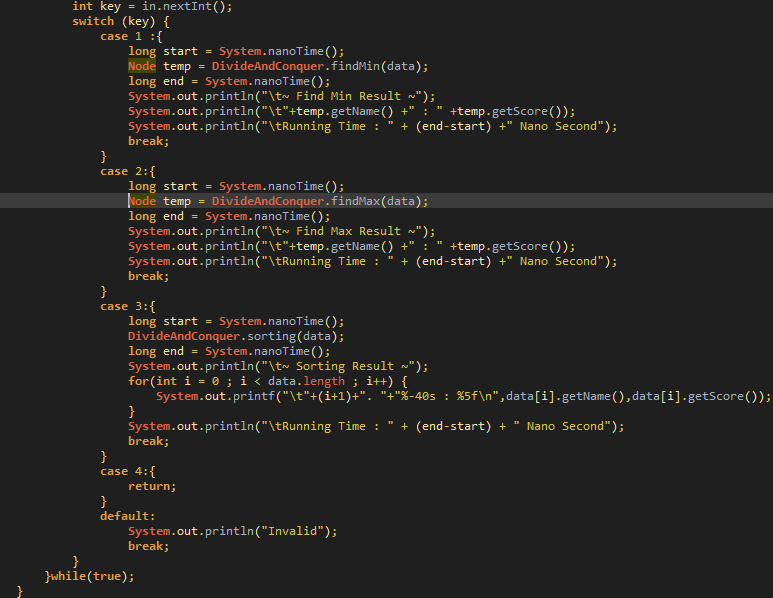
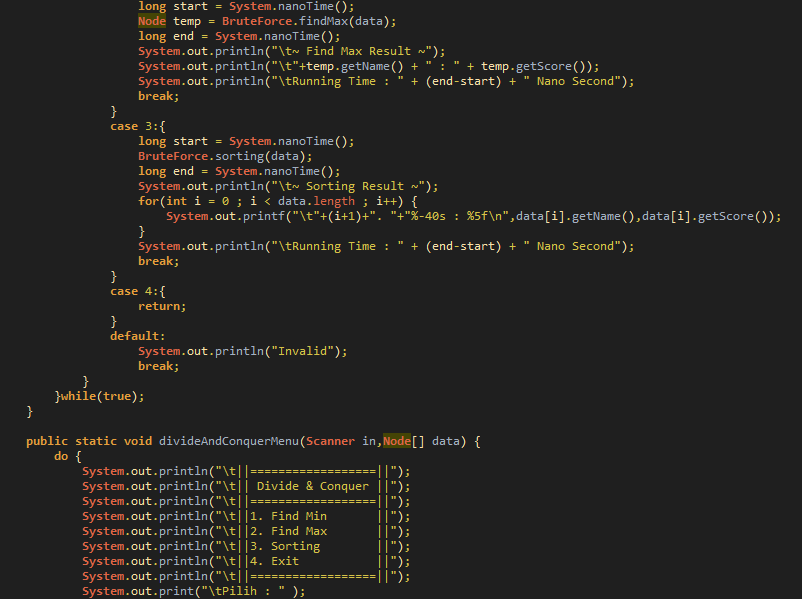
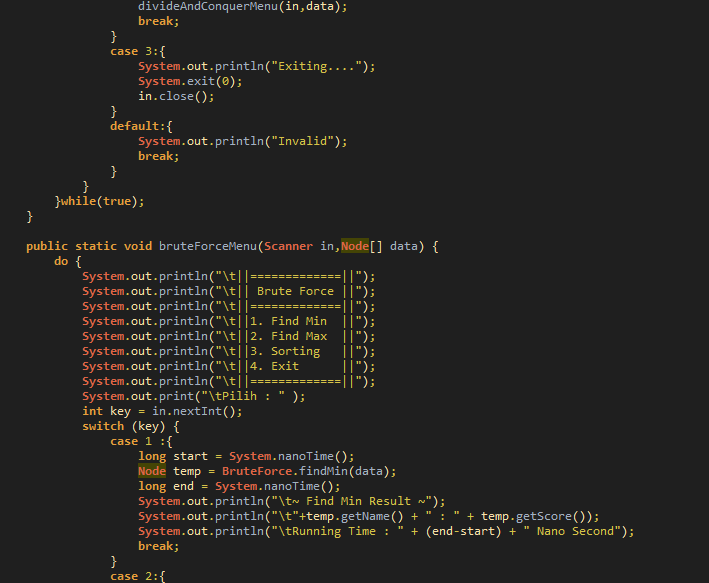
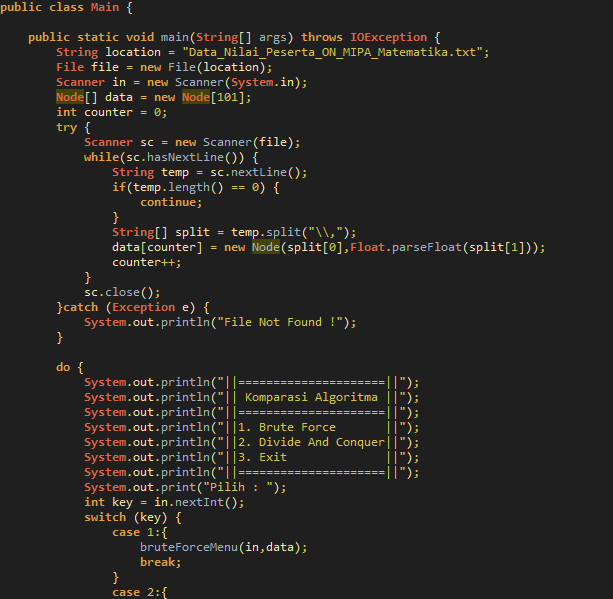
**BAB I**

**Screenshot**

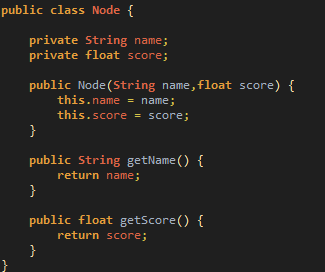
1. **Screenshot Source Code**
   * 1. **BruteForce**

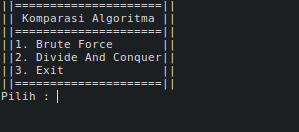
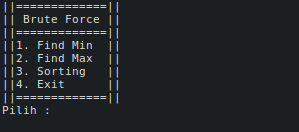
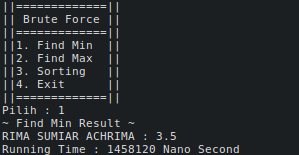
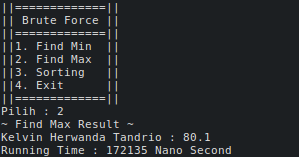
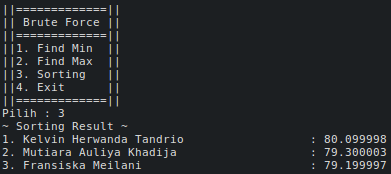
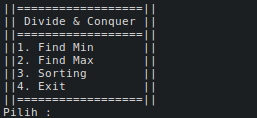
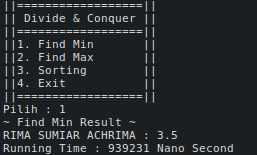
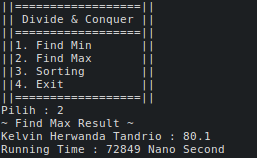
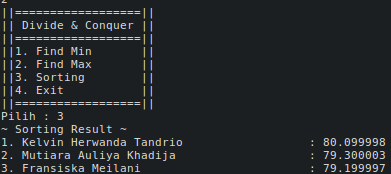
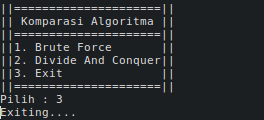


* + 1. **DevideAndConquer**
    2. **Main**



* + 1. **Node**



1. **Screenshot Running**
   1. **Main Menu  
      **
   2. **Brute Force  
      **
      1. **Find Min  
         **
      2. **Find Max  
         **
      3. **Sorting  
         **
   3. **Divide And Conquer  
      **
      1. **Find Min  
         **
      2. **Find Max  
         **
      3. **Sorting**
   4. **Exit  
      **

**BAB II**

**Analisa**

1. **Analisa Source Code**

|  |
| --- |
|  |

Dalam program ini terdapat 3 Menu yaitu, menu Brute Force, menu Divide And Conquer dan Menu Exit. Pada Kedua menu Brute Foce dan Divide And Conquer terdapat sub menu yang sama yaitu sub menu Find Min, Find Max, Sorting.

1. Menu Brute Force
   1. Find Min

|  |
| --- |
|  |

Cara Kerja Menu ini adalah dengan mengecek satu satu Node[] data dari indeks pertama sampai indeks terakhir. Elemen terkecil pertama adalah Node pada indeks ke-0 yang akan disimpan pada Node temp, lalu dicek mulai dari Node indeks ke-1 sampai akhir, jika terdapat Node dengan *score* lebih kecil dari *score* dari Node temp, maka Node temp akan digantikan Node tersebut

* 1. Find Max

|  |
| --- |
|  |

Cara kerja Menu ini adalah hampir sama seperti metode Find Min, namun sekarang yang dicari adalah Node dengan *score* paling besar.

* 1. Sorting

|  |
| --- |
|  |

Dapat diketahui algoritma sorting diatas adalah algoritma *selection sort*, cara kerjanya adalah

1. int i = 0
2. Cari indeks dari Node dengan score terbesar dari array Node[] data indeks ke-i sampai data.length dan simpan pada variable max
3. Tukarkan posisi Node indeks ke-i dengan Node indeks ke-max
4. Ulangi langkah no 2 dengan menambah nilai i dengan 1
5. Menu Divide And Conquer
   1. Find Min

|  |
| --- |
|  |

Cara kerja Menu ini adalah dengan menggunakan konsep divide and conquer yaitu membagi tugas besar(divide) menjadi tugas tugas yang kecil, lalu solusi dari tugas kecil kecil tersebut digabungkan untuk dijadikan 1 solusi (conquer), visualisasinya dapat dilihat pada gambar dibawah ini

4

3

6

9

4

3

6

9

4

3

6

9

3

6

3

* 1. Find Max

|  |
| --- |
|  |

Cara kerja Menu ini hampir sama seperti menu find Min sebelumnya, yaitu menggunakan konsep dari divide and conquer. Namun pada menu ini yang dicari adalah Node dengan *score* yang paling besar. Visualisasinya dapat dilihat pada gambar dibawah

4

3

6

9

4

3

6

9

4

3

6

9

4

9

9

* 1. Sorting

|  |
| --- |
|  |

Cara kerja Menu sorting ini adalah menggunakan algoritma *merge sort*. Yaitu ukuran array akan dipecah pecah menjadi bagian bagian kecil lalu akan digabungkan lagi, visualisasinya dapat dilihat pada gambar dibawah ini

4

3

6

9

4

3

6

9

4

3

6

9

4

3

9

6

9

6

4

1. Menu Exit

|  |
| --- |
|  |

Menu ini bertujuan untuk keluar dari program

1. **Analisa Algoritma**
   1. **Brute Force**
      1. **Sorting**

|  |
| --- |
| **Gambar 1.1 Method Sorting() pada class BruteForce** |

Dapat dilihat pada gambar diatas, method bertujuan untuk menyorting data menggunakan metode brute force, untuk analisa algoritmanya dapat dilihat pada tabel dibawah ini

|  |  |
| --- | --- |
| **COST** | **TIMES** |
| C1  *for(int i = 0 ; i < data.length ; i++)* | *n* |
| C2  *int min = 1* | *n*  Keterangan : 1\*C1 |
| C3  *for(int j = (i+1) ; j <= data.length-1 ; j++)* | *n\*(n-1)*  Keterangan : C1\*(*n-1)* |
| C4  *if(data[j].getScore() < data[min].getScore())* | *1\*{n\*(n-1)}*  Keterangan : 1\*C3 |
| C5  *min = j* | *1\*{n\*(n-1)}*  Keterangan : 1\*C3 |
| C6  *Node temp = data[min];* | *1\*n*  Keterangan : 1\*C1 |
| C7  *data[min] = data[i]* | *1\*n*  Keterangan : 1\*C1 |
| C8  *data[i] = temp;* | *1\*n*  Keterangan : 1\*C1 |

Total Cost **:** *{n} + {n} + {n\*(n-1)} + {n\*(n-1)} + {n} + {n} + {n}*

: 2{} + 4{}

:

Jadi Dapat disimpulkan Notasi Big O nya adalah

* + 1. **Find Min**

|  |
| --- |
| **Gambar 1.2 Method findMin() pada class BruteForce** |

Dapat dilihat pada gambar diatas, method tersebut bertujuan untuk mencari data dengan elemen terkecil menggunakan metode brute force, untuk analisa algoritmanya dapat dilihat pada tabel dibawah ini

|  |  |
| --- | --- |
| **COST** | **TIMES** |
| C1  *Node temp = data[0];* | *1* |
| C2  *for(int i = 1 ; i < data.length ; i++)* | *(n-1)* |
| C3 *if(temp.getScore() > data[i].getScore())* | *1\*(n-1)*  Keterangan : 1\*C2 |
| C4 *temp = data[i];* | *1\*(n-1)*  Keterangan : 1\*C2 |
| C5 *return temp;* | 1 |

Total Cost :

:

:

Jadi Dapat disimpulkan Notasi Big O nya adalah

* + 1. **Find Max**

|  |
| --- |
| **Gambar 1.3 Method findMax() pada class BruteForce** |

Dapat dilihat pada gambar diatas, method tersebut bertujuan untuk mencari data dengan elemen terbesar menggunakan metode brute force, untuk analisa algoritmanya dapat dilihat pada tabel dibawah ini

|  |  |
| --- | --- |
| **COST** | **TIMES** |
| C1  *Node temp = data[0];* | *1* |
| C2  *for(int i = 1 ; i < data.length ; i++)* | *(n-1)* |
| C3 *if(temp.getScore() > data[i].getScore())* | *1\*(n-1)*  Keterangan : 1\*C2 |
| C4 *temp = data[i];* | *1\*(n-1)*  Keterangan : 1\*C2 |
| C5 *return temp;* | 1 |

Total Cost :

:

:

Jadi Dapat disimpulkan Notasi Big O nya adalah

* 1. **Divide And Conquer**
     1. **Sorting**

|  |
| --- |
| **Gambar 2.1 Method sort() pada class DivideAndConquer** |

Dapat dilihat pada gambar diatas, method tersebut bertujuan untuk menyorting data menggunakan metode divide and conquer, untuk analisa algoritmanya dapat dilihat pada tabel dibawah ini

|  |  |
| --- | --- |
| **COST** | **TIMES** |
| C1  *int middle = (left+right)/2;* | 1 |
| C2  *sort(data,left,middle);* | *T(n/2)* |
| C3  *sort(data,middle+1,right);* | *T(n/2)* |
| C4  *merge(data,left,middle,right);* | *n* |

|  |
| --- |
| Total Cost dapat ditulis :  (Anggap 1 tidak ada)  Dari Persamaan diatas, dapat diurai lagi menjadi seperti dibawah ini    (diurai lagi)      Dari hasil penguraian persamaan diatas, maka didapat persamaan karakteristik :  …………….(i) |
| Jika persamaan diatas diurai terus sampai menjadi { Langkah *divide* selesai) maka didapatkan persamaan  ……………………………..(ii)  ……………………..…(iii)  Langkah Terakhir subtitusi persamaan (ii) dan (iii) ke persamaan (i), hasilnya :  Maka dapat disimpulkan bahwa notasi Big O dari algoritma diatas adalah |

* + 1. **Find Min**

|  |
| --- |
| **Gambar 2.2 Method divideMin() pada class DivideAndConquer** |

Dapat dilihat pada gambar diatas, method tersebut bertujuan untuk mencari elemen terkecil pada data menggunakan metode divide and conquer, untuk analisa algoritmanya dapat dilihat pada tabel dibawah ini

|  |  |
| --- | --- |
| **COST** | **TIMES** |
| C1  *if(left==right)* | *1* |
| C2  *return data[left]* | *1* |
| C3  *Int middle = (left+right)/2* | *1* |
| C4  *Node min1 = divideMin(data,left,middle)* | *T(n/2)* |
| C5  *Node min2 = divideMin(data,middle+1,right)* | *T(n/2)* |
| C6  *return conquerMin(min1,min2)* | *1* |

Total Cost dapat ditulis :

(1 tidak dianggap)

Jika persamaan diatas diurai lagi maka akan didapat

Dari hasil penguraian persamaan diatas, maka didapat persamaan karakteristik :

………………….(i)

Jika persamaan diatas diurai terus sampai menjadi { Langkah *divide* selesai) maka didapatkan persamaan

……………………………..(ii)

Terakhir subtitusi persamaan (ii) ke persamaan (i) maka akan didapat  
   
Maka dapat disimpulkan bahwa notasi Big O dari algoritma diatas adalah

* + 1. **Find Max**

|  |
| --- |
| **Gambar 2.3 Methode divideMax() pada class DivideAndConquer** |

Dapat dilihat pada gambar diatas, method tersebut bertujuan untuk mencari elemen terbesar pada data menggunakan metode divide and conquer, untuk analisa algoritmanya dapat dilihat pada tabel dibawah ini

|  |  |
| --- | --- |
| **COST** | **TIMES** |
| C1  *if(left==right)* | *1* |
| C2  *return data[left]* | *1* |
| C3  *Int middle = (left+right)/2* | *1* |
| C4  *Node min1 = divideMax(data,left,middle)* | *T(n/2)* |
| C5  *Node min2 = divideMax(data,middle+1,right)* | *T(n/2)* |
| C6  *return conquerMax(min1,min2)* | *1* |

Total Cost dapat ditulis :

(1 tidak dianggap)

Jika persamaan diatas diurai lagi maka akan didapat

Dari hasil penguraian persamaan diatas, maka didapat persamaan karakteristik :

………………….(i)

Jika persamaan diatas diurai terus sampai menjadi { Langkah *divide* selesai) maka didapatkan persamaan

……………………………..(ii)

Terakhir subtitusi persamaan (ii) ke persamaan (i) maka akan didapat  
   
Maka dapat disimpulkan bahwa notasi Big O dari algoritma diatas adalah