Nama : Naufal Ariful Amri

NPM : 140810180009

Kelas : A

Studi Kasus Divide and Conquer

1. Merge Sort

Setelah anda mengetahui algoritma Merge-Sort mengadopsi paradigma divide and conquer, lakukan hal berikut.

a. Buat program Merge-Sort dengan bahasa C++

b. Kompleksitas waktu algoritma merge sort adalah O(n log n). Cari tahu kecepatan komputer anda dalam memproses program. Hitung berapa running time yang dibutuhkan apabila input untuk merge-sortnya adalah 20

jawab :

a. Code

```
#include<iostream>
using namespace std;
int Merge(int array[] , int p , int q , int r){
     int left_size,right_size,i,j,k;

     left_size = q - p + 1; // array kiri
     right_size = r - q ; // array kanan

     int L[left_size] , R[right_size];
     for(i = 0 ; i < left_size ; i++) {
          L[i] = array[p+i] ;  // bagian kiri
     }
     for(j = 0 ; j < right_size ; j++) {
          R[j] = array[q+j+1]; // bagian kanan
     }


     i = 0,j = 0 ;
     // perbandingan untuk gabung
     for(k = p ; i < left_size && j < right_size ; k++) {
          if(L[i] < R[j]) {
              array[k] = L[i++];
          }
          else {
              array[k] = R[j++];
          }
     }

     while(i < left_size) {
         array[k++] = L[i++];
     }
     while(j < right_size) {
```

```
        array[k++] = R[j++];
    }

}


// merge sort
int MergeSort(int array[],int p,int r){
    int q;
    if(p < r){
        q = (p + r) /2;
        MergeSort(array , p , q);
        MergeSort(array , q+1 , r);
        Merge(array , p , q , r);
    }
}

int main() {
    int n ;
    cout << "Jumlah Array : ";
    cin >> n;
    int array[n] , i;

    for(i = 0 ; i < n ; i++){
        cout << "Input array ke - " << i+1 << " : ";
        cin >> array[i];
    }

   MergeSort(array , 0 , n-1);
   cout << "Merge Sort Sukses !!" << endl ;

   for(i = 0 ; i < n ; i++) {
       cout << array[i] << " ";
   }
   return 0;
}
```

b. Running time ketika data yang diinput adalah 20, di komputer saya kecepatannya 3 microsecond

2. Selection Sort

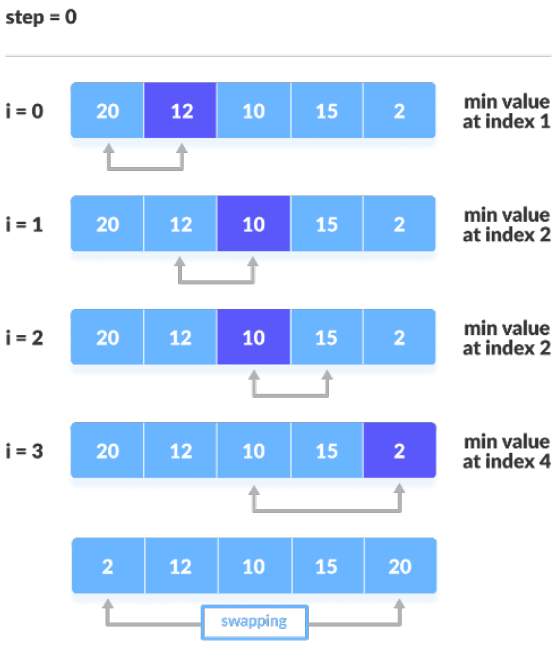
Selection sort merupakan salah satu algoritma sorting yang berparadigma divide and conquer. Untuk membedah algoritma selection sort, lakukan langkah berikut

- Pelajari cara kerja algoritma selection sort
- Tentukan $T(n)$ dari rekurensi selection sort berdasarkan rekurensi divide and conquer
- Selesaikan persamaan rekurensi dengan metode recursion-tree untuk mendapatkan kompleksitas waktu asmptotik dalam big-O, big-Ω, big-θ
- Lakukan implementasi koding program untuk algoritma selection sort dengan bahasa c++

Jawab :

Metode Insertion sort

- Bandingkan array ke-0 dengan array selanjutnya
- Jika array selanjutnya lebih kecil, maka yang akan dibandingkan selanjutnya adalah array yang lebih kecil (sebut saja array temp)
- Jika sudah mencapai array akhir, ganti array pertama dengan array temp
- Lanjut dengan array ke- 1. Gunakan langkah yang sama sehingga sampai array ke-i.




```
#include<iostream>
using namespace std;

int main(int argc, char const *argv[]) {
     int array[10] = {1 , 4 , 5 , 2 , 3 , 6 , 8 , 3 , 7 , 8} ;
     int temp , swap ;
     int nilai ;

     for (int i = 0; i < 10; i++) {
          temp = array[i] ;
          for (int j = i + 1 ; j < 10 ; j++) {
               if (temp > array[j]){
                    temp = array[j] ;
                    nilai = j ;
               }
          }

          if(temp != array[i]){
               swap = array[i] ;
               array[i] = array[nilai] ;
               array[nilai] = swap ;
          }
     }
```

```
    for (int i = 0; i < 10; i++) {
         cout << array[i] << " " ;
    }

    return 0;
}
```

Kompleksitas waktu

$$T(n) = (n-1) + (n-2) + \cdots + 1$$

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

$$T(n) = O(n^2)$$

$$T(n) = \Omega(n^2)$$

Karena $O(n^2) = \Omega(n^2)$,Maka $\odot (n^2)$

3. Insertion Sort

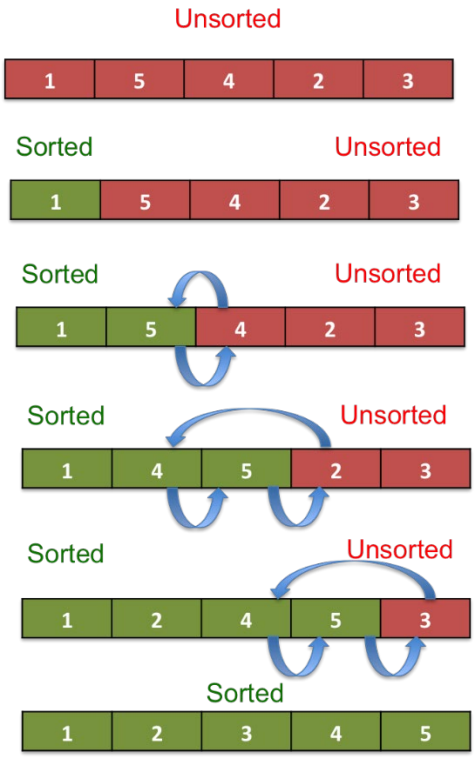
Selection sort merupakan salah satu algoritma sorting yang berparadigma divide and conquer. Untuk membedah algoritma selection sort, lakukan langkah ini.

- Pelajari cara kerja algoritma insertion sort
- Tentukan T(n) dari rekurensi pengulangan dengan metode recursion-tree untuk kompleksitas waktu asimptotik big-O, big-Ω, big-θ
- Lakukan implementasi koding dengan program insertion sort dengan menggunakan bahasa c++

Jawab :

Algoritam insertion sort

- Kunci data 2 dan simpan di temp
- Bandingkan dengan data sebelumnya, jika lebih kecil di swap
- Kunci data ke 3 dan simpan di temp
- Bandingkan denngan data sebelumnya, jika lebih kecil di swap




```
#include<iostream>
using namespace std ;


int main(int argc, char const *argv[]) {
     int array[10] ;
     int temp ;
     int nilai ;
     int x ;

     int y ;
     cout << "Input jumlah data : " ; cin >> y ;
     for (int i = 0; i < y; i++) {
          cout << "input data ke-" << i+1 << " : " ; cin >> array[i] ;
     }

     cout << "\nData belum disorting\n";
     for (int i = 0; i < y; i++) {
          cout <<   array[i] << " " ;
     }


     for (int i = 0; i < y; i++) {
          x = i ;
          for (int j = i ; j >= 0 ; j--) {
               if (array[x] < array[j]) {
                    temp = array[j] ;
                    array[j] = array[x] ;
                    array[x] = temp ;
                    x = j ;

               }
          }
```

```
    }

    cout << "\nsetelah di insertion sort\n" ;
    for (int k = 0; k < y; k++) {
         cout << array[k] << " ";
    }
    cout << endl ;
    return 0;
}
```

Banyaknya penyelesaian = n-1

Waktu divide = n

Waktu conquer = n

$$T(n) = \{\Theta(1)\ T(n-1) + \Theta(n)$$

Worst Case

$T(n) = cn + (cn - c) + (cn - 2c) + \cdots + 2c + c \leq 2cn^2 + cn^2$

$T(n) = \frac{c(n-1)(n-2)}{2} + c \leq 2cn^2 + cn^2$

$T(n) = \frac{n^2 - 3n + 2}{2} + c \leq 2cn^2 + cn^2$

$T(n) = \frac{cn^2}{2} - \frac{3n}{2} + 2c \leq 2cn^2 + cn^2$

$T(n) = O(n^2)$

Best Case

$T(n) = cn \geq 2cn$

$T(n) = \Omega(n)$

Average Case

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

$T(n) = \ \theta(n)$

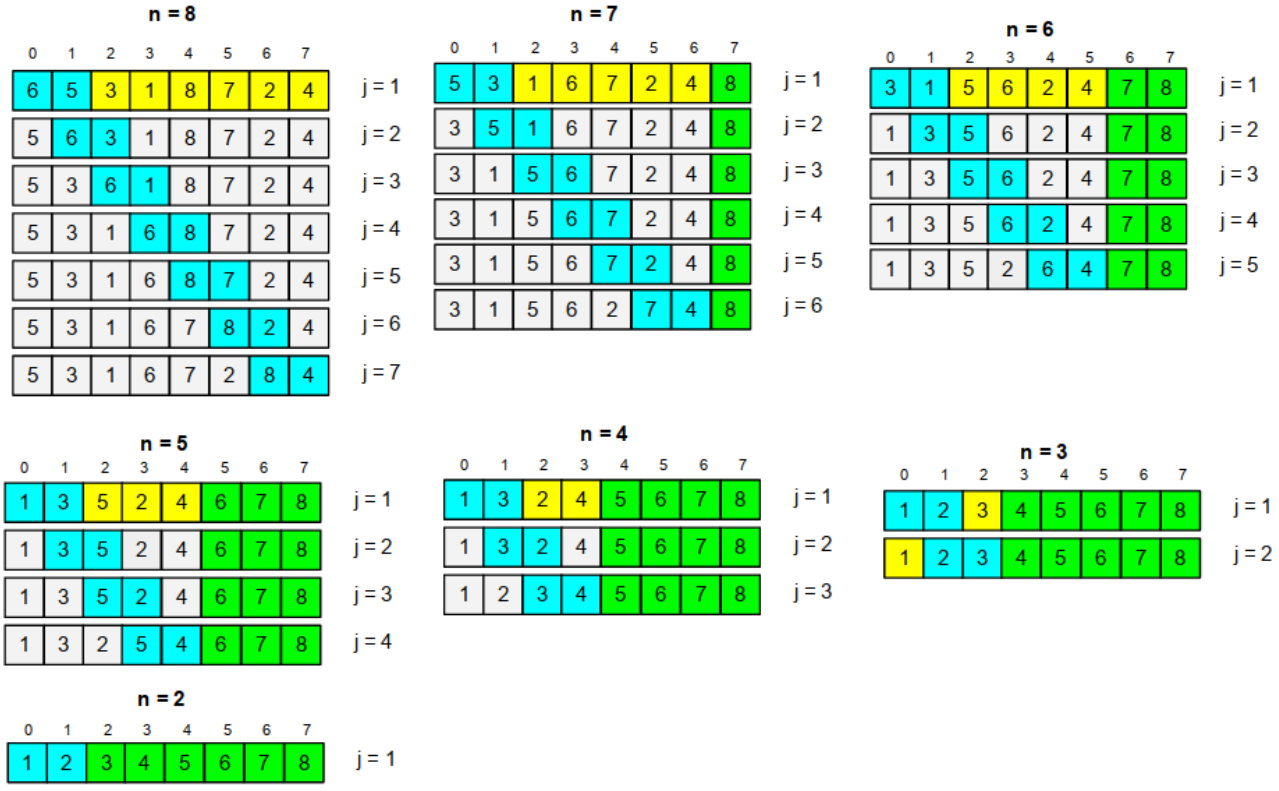
4. Bubble Sort

Selection sort merupakan salah satu algoritma sorting yang berparadigma divide and conquer. Untuk membedah algoritma selection sort, lakukan langkah ini.

- Pelajari cara kerja algoritma selection sort
- Tentukan T(n) dari rekurensi pengulangan dengan metode recursion-tree untuk kompleksitas waktu asimptotik big-O, big-Ω, big-θ
- Lakukan implementasi koding dengan program bubble sort dengan menggunakan bahasa c++

Jawab :

Algoritma bubble sort



- Loop data dari 1
- Bandingkan setiap data dengan data didepannya
- Jika lebih kecil lakukan swap
- Selesaikan 1 looping.
- Loop data dari 2 dan lakukan seperti langkah diatas

Banyaknya penyelesaian = n-1

Waktu divide = n

Waktu conquer = n

$$T(n) = \{\Theta(1)\, T(n-1) + \Theta(n)$$

Worst Case

$$T(n) = cn + (cn - c) + (cn - 2c) + \cdots + 2c + c \leq 2cn^2 + cn^2$$

$$T(n) = \frac{c(n-1)(n-2)}{2} + c \leq 2cn^2 + cn^2$$

$$T(n) = \frac{n^2 - 3n + 2}{2} + c \leq 2cn^2 + cn^2$$

$$T(n) = \frac{cn^2}{2} - \frac{3n}{2} + 2c \leq 2cn^2 + cn^2$$

$$T(n) = O(n^2)$$

Best Case

$$T(n) = cn + (cn - c) + (cn - 2c) + \cdots + 2c + c \geq 2cn^2 + cn^2$$

$$T(n) = \frac{c(n-1)(n-2)}{2} + c \geq 2cn^2 + cn^2$$

$$T(n) = \frac{n^2 - 3n + 2}{2} + c \geq 2cn^2 + cn^2$$

$$T(n) = \frac{cn^2}{2} - \frac{3n}{2} + 2c \geq 2cn^2 + cn^2$$

$$T(n) = \Omega(n^2)$$

Average Case

$$T(n) = 2cn^2 + cn^2$$

$$T(n) = \theta(n^2)$$

```
#include<iostream>
using namespace std ;

int main(int argc, char const *argv[])
{
    int array[8] ;
    int temp ;
```

```
    int n ;
    cout << "Input Jumlah Data : " ; cin >> n ;
    for (int i = 0 ; i < n; i++) {
        cout << "Data ke-" << i+1 << " : " ;
        cin >> array[i];
    }
    cout << "\nData belum disorting\n";
    for (int i = 0; i < n; i++) {
        cout << array[i] << " " ;
    }

    for (int i = 0 ; i < n ; i++) {
        for (int j = 0; j < n-1 ; j++) {
            if (array[j] > array[j+1]){
                temp = array[j] ;
                array[j] = array[j+1];
                array[j+1] = temp;
            }
        }
    }

    cout << "\nData telah disorting dengan bubble sort\n";
    for (int i = 0; i < n; i++) {
        cout << array[i] << " " ;
    }
    return 0;
}
```