JOBSHEET 09

Selection Sort and Merge Sort

Fakultas	Proram Studi	Kode MK	Waktu
Teknik	Teknik Informatika	INF1.62.2014	2 x 50 Menit

TUJUAN PRAKTIKUM

1. Mahasiswa memahami konsep pengurutan Selection dan Merge dalam pemrograman C.
2. Mahasiswa memahami operasi yang ada dalam pencarian pengurutan Selection dan Merge.
3. Mahasiswa mampu mengimplementasikan pencarian struktur data pengurutan Selection dan Merge menggunakan pemrograman C dengan IDE.

HARDWARE & SOFTWARE

1. Personal Computer
2. DevC++ IDE

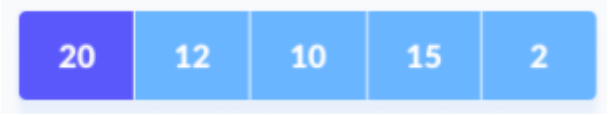
TEORI SINGKAT

1. Selection Sort

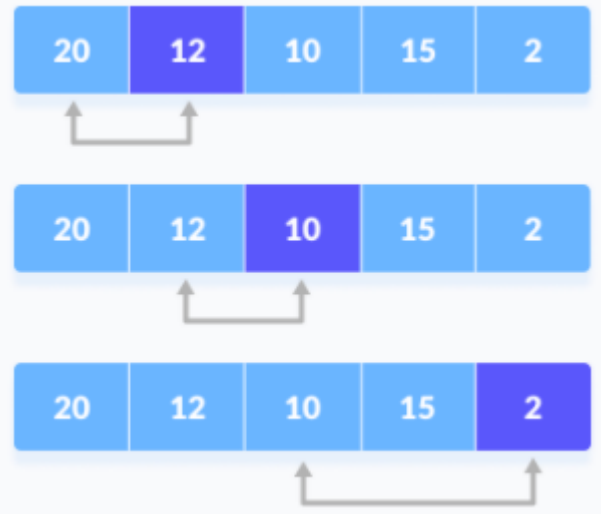
Selection sort adalah sebuah algoritma yang digunakan untuk mencari dna memilih element/data terkecil dalam sebuah list yang belum tersusun dalam setiap iterasi dan menempatkan elemen terkecil di urutan paling depan dari urutan list data.

Cara kerja Selection Sort :

1. Tentukan elemen pertama dari urutan menjadi elemen terkecil masukkan sebagai elemen "minimum".

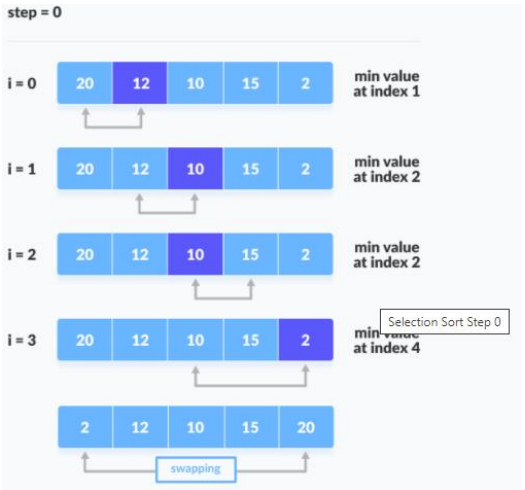


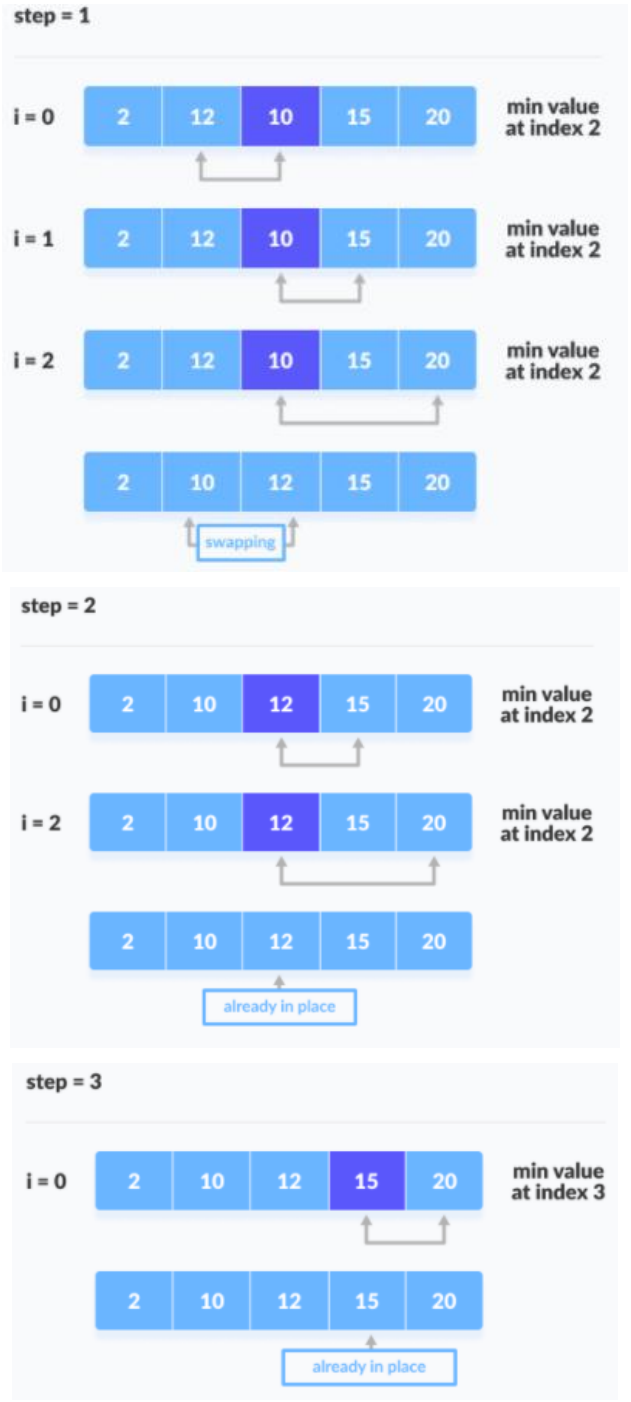
2. Bandingkan mnimum dengan urutan elemen kedua. Jika elemen kedua lebih kcil dibandingkan minimum, jadikan elemen kedua sebagai minimum. Bandingkan kembali minimum dengan elemen ketiga. Lakukan secara terus menerus samapai dengan elemen yang terakhir.



3. Setelah setiap perulangan, minimum ditempatkan di paling depan urutan list data

4. pada setiap pengulangan, indeksasi (pemberian alamat) dimulai dari elemen pertama dari list . langkah 1-3 dilakukan berulang kali samapi keseluruhan elemen digantikan pada posisi yang terurut.

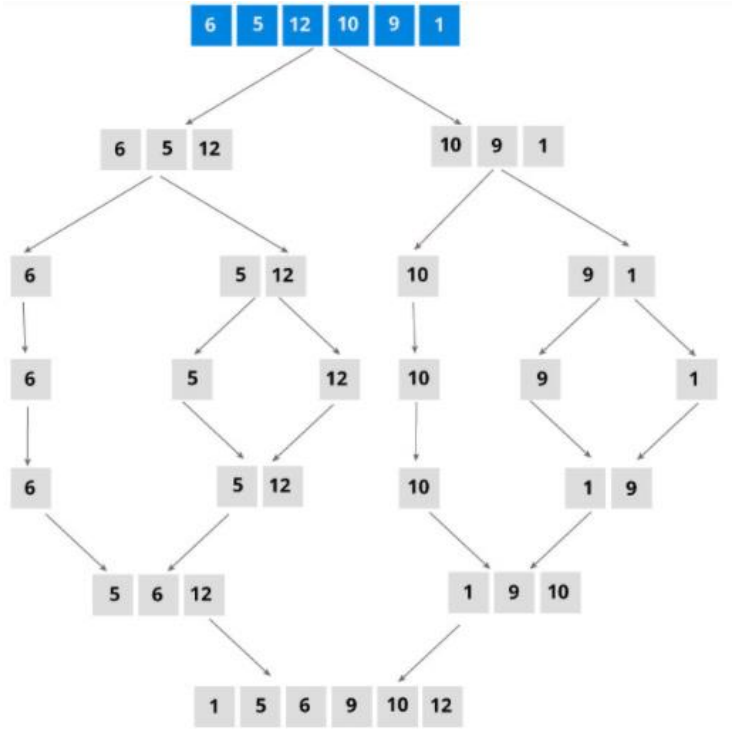




2. Merge Sort

Merge sort is a sorting technique based on divide and conquer technique. With worst-case time complexity being O(n log n), it is one of the most respected algorithms. Merge sort first divides the array into equal halves and then combines them in a sorted manner.

Merge sort adalah sebuah teknik pengurutan berdasarkan teknik membagi dan menggabungkan (divide and conquer). Menggunakan merge sort hal pertama yang dilakukan adalah membagi sama banyak data kemudian menggabungkannya dalam list data yang terurut.



Divide and Conquer Strategy

Menggunakan teknik Divide and Conquer, langkah pertama bagi sebuah permasalahan menjadi sub bagian kecil permasalahan. Di saat sub permasalahan telah diselesaikan dan siap, gabungkan hasil dari sub permasalahan untuk mememcahkan maslaah utama.

Dimisalkan dalam sebuah pemograman harus mengurutkan sebuah array A. Menggunakan teknik ini array A dibag menjadi subpermasalahan diulai dari index p dan diakhiri index r, dimisalkan arrat A[p..r].

a. Divide

   Jika elemen ’q’ terletak diantara p dan r, lalu bagi subarray menjadi 2 array A[p..q] dan A[q+1,r]

b. Conquer

   Pada langkah conquer ini cobalah untuk mengurutkan masing-masing subarray A[p..q] dan A[q+1,r] . jika belum memenuhi maka bagi kembali masing-masing subarray menjadi subarray yang lebih kecil dan lakukan pengurutan.

c. Combine

   Langkah coquer telah memenuhi dan mendapatkan 2 subarray ayng telah diurutkan A[p..q] dan A[q+1, r] untuk array A[p..r], lakukan kombinasi dari sub array A[p..q] and A[q+1, r] menghasilkan array A[p..r] yang terurut

   Link video klik

PERCOBAAN

1. **Merge Sort – Mengurutkan Data Random Dalam Sebuah Array**

```
#include <stdio.h>
#include <stdlib.h>
#define max 10000

void input();
void tukar(int *, int *);
void tampil();
void partisi(int data[],int low,int high);
void mergesort(int data[],int low,int mid,int high);
int data[max],hasil[max];
int n;

int main()
{
 input();
 int awal=0, akhir=n-1;
        partisi(data,awal,akhir);
 tampil();
}
//fungsi input
void input()
{
 int i;
 printf("Masukkan jumlah total elemen: ");
    scanf("%d",&n);

    puts(" ");
    for(i=0;i<n;i++)
 {
  data[i]=rand();
  printf("%d\t",data[i]);
  //printf("Elemen ke-%d: ",i+1);
        //scanf("%d",&data[i]);
    }
}
//fungsi mergesort
void partisi(int data[],int low,int high)
{
   int mid;
   if(low<high)
 {
    mid=(low+high)/2;
    partisi(data,low,mid);
    partisi(data,mid+1,high);
    mergesort(data,low,mid,high);
  }
}
void mergesort(int data[],int low,int mid,int high)
{
    int i,m,k,l,temp[max];
    l=low;
    i=low;
    m=mid+1;
    while((l<=mid)&&(m<=high))
```

```
        {
           if(data[l]<=data[m])
          {
                 temp[i]=data[l];
             l++;
          }
           else
          {
               temp[i]=data[m];
             m++;
          }
           i++;
        }
        if(l>mid)
        {
          for(k=m;k<=high;k++)
          {
             temp[i]=data[k];
             i++;
           }
         }
        else
        {
          for(k=l;k<=mid;k++)
          {
             temp[i]=data[k];
             i++;
          }
        }
        for(k=low;k<=high;k++)
        {
           data[k]=temp[k];
        }
       }
       //fungsi tampil
       void tampil()
       {
        int j;
        puts("\n");
        for(j=0;j<n;j++)
        {printf("%d\t",data[j]);}
        puts("\n");
}
```

2. **Merge Sort – Before and After**

```
#include <stdio.h>
#define max 10

int a[11] = { 10, 14, 19, 26, 27, 31, 33, 35, 42, 44, 0 };
int b[10];

void merging(int low, int mid, int high) {
   int l1, l2, i;
   for(l1 = low, l2 = mid + 1, i = low; l1 <= mid && l2 <= high;
i++) {
      if(a[l1] <= a[l2])
         b[i] = a[l1++];
```

```
        else
            b[i] = a[l2++];
    }
    while(l1 <= mid)
        b[i++] = a[l1++];
    while(l2 <= high)
        b[i++] = a[l2++];
    for(i = low; i <= high; i++)
        a[i] = b[i];
}
void sort(int low, int high) {
    int mid;

    if(low < high) {
        mid = (low + high) / 2;
        sort(low, mid);
        sort(mid+1, high);
        merging(low, mid, high);
    } else {
        return;
    }
}
int main() {
    int i;
    printf("Data sebelum diurut\n");
    for(i = 0; i <= max; i++)
        printf("%d ", a[i]);
    sort(0, max);
    printf("\nData setelah diurut\n");
    for(i = 0; i <= max; i++)
        printf("%d ", a[i]);
}
```

3. **Merge Sort 3 – Entry Elements**

```
#include <stdio.h>

// function to sort the subsection a[i .. j] of the array a[]
void merge_sort(int i, int j, int a[], int aux[]) {
    if (j <= i) {
        return;     // the subsection is empty or a single
element
    }
    int mid = (i + j) / 2;

    // left sub-array is a[i .. mid]
    // right sub-array is a[mid + 1 .. j]

    merge_sort(i, mid, a, aux);     // sort the left sub-array
recursively
    merge_sort(mid + 1, j, a, aux);     // sort the right sub-
array recursively

    int pointer_left = i;       // pointer_left points to the
beginning of the left sub-array
    int pointer_right = mid + 1;        // pointer_right points
to the beginning of the right sub-array
    int k;      // k is the loop counter
```

```
    // we loop from i to j to fill each element of the final
merged array
    for (k = i; k <= j; k++) {
        if (pointer_left == mid + 1) {      // left pointer has
reached the limit
            aux[k] = a[pointer_right];
            pointer_right++;
        } else if (pointer_right == j + 1) {        // right
pointer has reached the limit
            aux[k] = a[pointer_left];
            pointer_left++;
        } else if (a[pointer_left] < a[pointer_right]) {
// pointer left points to smaller element
            aux[k] = a[pointer_left];
            pointer_left++;
        } else {        // pointer right points to smaller
element
            aux[k] = a[pointer_right];
            pointer_right++;
        }
    }

    for (k = i; k <= j; k++) {  // copy the elements from aux[] to a[]
        a[k] = aux[k];
    }
}

int main() {
  int a[100], aux[100], n, i, d, swap;

  printf("Masukan jumlah elemen didalam array:\n");
  scanf("%d", &n);

  printf("Masukan %d bilangan integer\n", n);

  for (i = 0; i < n; i++)
    scanf("%d", &a[i]);

  merge_sort(0, n - 1, a, aux);

  printf("Menampilkan array yang telah diurutkan:\n");

  for (i = 0; i < n; i++)
     printf("%d\n", a[i]);

  return 0;
}
```

4. **Selection Sort – Proses Iterasi Pengurutan Data**

```
#include <stdio.h>
#include <stdbool.h>
#define MAX 7

int intArray[MAX] = {4,6,3,2,1,9,7};

void printline(int count) {
   int i;
```

```
   for(i = 0;i < count-1;i++) {
      printf("=");
   }

   printf("=\n");
}

void display() {
   int i;
   printf("[");

   // navigate through all items
   for(i = 0;i < MAX;i++) {
      printf("%d ", intArray[i]);
   }

   printf("]\n");
}

void selectionSort() {
   int indexMin,i,j;

   // loop through all numbers
   for(i = 0; i < MAX-1; i++) {

      // set current element as minimum
      indexMin = i;

      // check the element to be minimum
      for(j = i+1;j < MAX;j++) {
         if(intArray[j] < intArray[indexMin]) {
            indexMin = j;
         }
      }

      if(indexMin != i) {
         printf("Items swapped: [ %d, %d ]\n" , intArray[i],
intArray[indexMin]);

         // swap the numbers
         int temp = intArray[indexMin];
         intArray[indexMin] = intArray[i];
         intArray[i] = temp;
      }

      printf("Iteration %d#:",(i+1));
      display();
   }
}
int main() {
   printf("Input Array: ");
   display();
   printline(50);
   selectionSort();
   printf("Output Array: ");
   display();
   printline(50);
}
```

5. **Selection Sort – Ascending**

```
#include <stdio.h>

void swap(int *a, int *b)
{
  int temp = *a;
  *a = *b;
  *b = temp;
}
void selectionSort(int array[], int size)
{
  for (int step = 0; step < size - 1; step++)
  {
    int min_idx = step;
    for (int i = step + 1; i < size; i++)
    {
      if (array[i] < array[min_idx])
        min_idx = i;
    }
    swap(&array[min_idx], &array[step]);
  }
}
void printArray(int array[], int size)
{
  for (int i = 0; i < size; ++i)
  {
    printf("%d  ", array[i]);
  }
  printf("\n");
}
int main()
{
  int data[] = {20, 12, 10, 15, 2};
  int size = sizeof(data) / sizeof(data[0]);
  selectionSort(data, size);
  printf("Sorted array in Acsending Order:\n");
  printArray(data, size);
}
```

6. **Selection Sort – Entry Elements**

```
#include <stdio.h>
void SelSort(int array[],int n);

int main()
{
     int array[100], n,i;
     printf("Masukan jumlah elemen ");
     scanf("%d", &n);
     printf("Masukan %d buah bilangan\n", n);
     for(i = 0; i < n; i++)
     scanf("%d", &array[i]);
     SelSort(array,n);
     return 0;
}
```

```
void SelSort(int array[], int n)
{
    int i, j, position, swap;
    for(i = 0; i < (n - 1); i++)
    {
        position=i;
        for(j = i + 1; j < n; j++)
        {
            if(array[position]>array[j])
            position=j;
        }
        if(position != i)
            {
                swap=array[i];
                array[i]=array[position];
                array[position]=swap;
            }
     }
     printf("Array yang telah diurutkan:\n");
     for(i = 0; i < n; i++)
     printf("%d\n", array[i]);
}
```

7. **Selection Sort – Entry Elements Cara 2**

```
#include <stdio.h>
int main()
{
     int a[100], n, i, j, position, swap;
     printf("Masukan jumlah elemen ");
     scanf("%d", &n);
     printf("Masukan %d buah bilangan\n", n);
     for (i = 0; i < n; i++)
           scanf("%d", &a[i]);
     for(i = 0; i < n - 1; i++)
     {
         position=i;
         for(j = i + 1; j < n; j++)
         {
             if(a[position] > a[j])
             position=j;
         }
         if(position != i)
         {
             swap=a[i];
             a[i]=a[position];
             a[position]=swap;
           }
     }
     printf("Array yang telah diurutkan:\n");
     for(i = 0; i < n; i++)
     printf("%d\n", a[i]);
     return 0;
}
```

TUGAS

1. Carilah contoh aplikasi yang mengimplementasikan selection sort dan merge sort serta jelaskan bagaimana aplikasi tersebut bekerja sesuai dengan prinsip kedua metode sorting tersebut!

DAFTAR PUSTAKA

1. Kernighan, Brian W, & Ritchie, Dennis M. 1988. The Ansi C Programming Language Second Edition, Prentice-Hall.
2. Cipta Ramadhani. 2015. Dasar Algoritma & Struktur Data. Yogyakarta: ANDI.