IK131 Algoritma dan Pemrograman

Materi: Pengurutan (Sort)

Di Susun Oleh:

Tim Pendamping Pak Jajang

2425 – Jajang Kusnendar, M.T.

PROGRAM STUDI PENDIDIKAN ILMU KOMPUTER

FAKULTAS PENDIDIKAN MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS PENDIDIKAN INDONESIA

1. Identitas

Nama Mata Kuliah – Kode	**:**	**IK131 – Algoritma dan Pemrograman**
Materi	**:**	**Algoritma Pengurutan**
Sub Materi	**:**	- **Bubble Sort** - **Selection Sort** - **Insertion Sort** - **Quick Sort**
Beban (Waktu Pelaksanaan)	**:**	**150 menit**
Semester	**:**	**1**
Pengampu MK	**:**	**Jajang Kusnendar, M.T**

2. Capaian Praktikum

Capaian Praktikum
- **Memahami konsep pengurutan dan algoritma pengurutan** - **Mampu membuat kode program algoritma pengurutan dalam bahasa C** - **Mampu menganalisis perbandingan keunggulan dan kekurangan antar sesama algoritma pengurutan**

3. Perangkat yang Dibutuhkan

Perangkat lunak
- Compiler bahasa C - Pada windows: DevC++, VisualStudio

Ringkasan Materi

https://www.geeksforgeeks.org/bubble-sort-algorithm/

https://www.geeksforgeeks.org/selection-sort-algorithm-2/

https://www.geeksforgeeks.org/insertion-sort-algorithm/

https://www.geeksforgeeks.org/bubble-sort-algorithm/

Lembar Kerja Mahasiswa

1. Bubble Sort

```
3  //Menukar
4  void tukar(int* a, int* b){
5      int temp = *a;
6      *a = *b;
7      *b = temp;
8  }
9
0  // Ascending, tumpuk kecil di depan
1  void bubble1(int arr[], int arr_size){
2      for (int i = (arr_size - 1); i > 0; --i){
3          for (int j = (arr_size - 1); j > (arr_size - 1) - i; --j){
4              if(arr[j] < arr[j-1]){
5                  tukar(&arr[j-1], &arr[j]);
6                  for (int k = 0; k < arr_size; ++k){
7                      printf("%d ", arr[k]);
8                  }
9                  printf("\n");
0              }
1          }
2      }
3  }
```

Sudah ada potongan program seperti di atas. Jika dijalankan akan mengurutkan array secara Ascending (kecil ke besar) dengan cara menumpuk bilangan terkecil di kiri (index 0 pada array).
Contoh:

```
67  int main(){
68      int arr[5] = {4,3,1,5,2};
69      bubble1(arr, 5);
70
```

```
4 3 1 2 5
4 1 3 2 5
1 4 3 2 5
1 4 2 3 5
1 2 4 3 5
1 2 3 4 5
[Finished in 1.3s]
```

Angka “1” harus bertukar dua kali supaya sampai di paling kiri
Angka “2” harus bertukar dua kali supaya sampai di paling kiri, tapi setelah “1”
Angka “3” harus bertukar satu kali supaya sampai di paling kiri, tapi setelah “2”
Angka “4” kebetulan sudah benar, tidak perlu bertukar
Angka “5” pasti benar, karena 1,2,3,4 sudah benar

Soal:
Buat fungsi/prosedur bubble sort dengan ketentuan (pilih salah satu saja): Descending tumpuk besar di kiri, Ascending tumpuk besar di kanan, atau Descending tumpuk kecil di kanan!

2. Selection Sort

```
4  void tukar(int* a, int* b){
5      int temp = *a;
6      *a = *b;
7      *b = temp;
8  }
9
10 //Ascending, pilih terkecil kiri ke kanan
11 void selection1(int arr[], int arr_size){
12     int index_maks;
13     for (int i = 0; i < arr_size - 1; ++i){
14         index_maks = i;
15         for (int j = i; j < arr_size; ++j){
16             if(arr[j] < arr[index_maks]){
17                 index_maks = j;
18             }
19         }
20         tukar(&arr[i], &arr[index_maks]);
21         for (int k = 0; k < arr_size; ++k){
22             printf("%d ", arr[k]);
23         }
24         printf("\n");
25     }
26 }
```

Sudah ada potongan program seperti di atas. Jika dijalankan akan mengurutkan array secara Ascending (kecil ke besar) dengan cara memilih bilangan terkecil dan kemudian ditukar ke posisi kiri.

Contoh:

```
79 int main(){
80     int arr[5] = {4,3,1,5,2};
81     selection1(arr, 5);
```

```
1 3 4 5 2
1 2 4 5 3
1 2 3 5 4
1 2 3 4 5
[Finished in 703ms]
```

Angka “1” terkecil, ditukar dengan posisi paling kiri
Angka “2” terkecil, ditukar dengan posisi paling kiri setelah “1”
Angka “3” terkecil, ditukar dengan posisi paling kiri setelah “2”
Angka “4” terkecil, ditukar dengan posisi paling kiri setelah “3”
Angka “5” pasti benar, karena 1,2,3,4 sudah benar

Soal:
Buat fungsi/prosedur selection sort dengan ketentuan (pilih salah satu saja): Descending tumpuk besar di kiri, Ascending tumpuk besar di kanan, atau Descending tumpuk kecil di kanan!

3. Insertion Sort

```
3  //Ascending, kiri ke kanan
4  void insertion1(int arr[], int arr_size){
5      int inserted;
6      for (int i = 0; i < arr_size; ++i){
7          inserted = arr[i];
8          int j = i;
9          while(j > 0 && arr[j-1] > inserted){
10             arr[j] = arr[j-1];
11             --j;
12         }
13         arr[j] = inserted;
14         for (int k = 0; k < arr_size; ++k){
15             printf("%d ", arr[k]);
16         }
17         printf("\n");
18     }
19 }
```

Sudah ada potongan program seperti di atas. Jika dijalankan akan mengurutkan array secara Ascending (kecil ke besar) dengan cara menumpuk sub-array yang sudah berurutan di kiri. Satu per satu, anggota array dimasukkan ke sub-array yang terurut pada posisi yang sesuai.

Contoh:

```
72  int main(){
73      int arr[5] = {4,3,1,5,2};
74      insertion1(arr, 5);
```

```
4 3 1 5 2
3 4 1 5 2
1 3 4 5 2
1 3 4 5 2
1 2 3 4 5
[Finished in 656ms]
```

Sub-array terurut ada di kiri, saat ini hanya ada 1 elemen, yaitu “4”
“3” dimasukkan ke sub-array, ditempatkan sebelum “4”, sekarang Panjang sub-array 2 anggota
“1” dimasukkan ke sub-array, ditempatkan sebelum “3”, sekarang Panjang sub-array 3 anggota
“5” dimasukkan ke sub-array, ditempatkan setelah “4”, sekarang Panjang sub-array 4 anggota
“2” dimasukkan ke sub-array, ditempatkan setelah “1” dan sebelum “3”, sekarang Panjang sub-array 5 anggota, selesai

Soal:
Buat fungsi/prosedur insertion sort dengan ketentuan (pilih salah satu saja): Descending sub-array terurut di kiri, Ascending sub-array terurut di kanan, atau Descending sub-array terurut di kanan!

4. Quick Sort

Diberikan prosedur/fungsi quick sort dengan partisi Hoare pivot Tengah (pivot median):

```
10  //Pivot tengah, Ascending, Partisi Hoare (bukan Lomuto)
11  void quick_tengah(int arr[], int arr_size, int batas_kiri, int batas_kanan){
12      if (batas_kiri >= batas_kanan){
13          return;
14      }
15
16      int pencari_kiri = batas_kiri;
17      int pencari_kanan = batas_kanan;
18      int index_pivot = (batas_kiri + batas_kanan)/2;
19
20      while(pencari_kiri <= pencari_kanan){
21          for (int i = 0; i < arr_size; ++i){
22              printf("%d ", arr[i]);
23          }
24          printf("\n");
25
26          while(arr[pencari_kiri] < arr[index_pivot]){ //balik tanda < untuk menjadi Desc, lakukan juga pada baris 29
27              ++pencari_kiri;
28          }
29          while(arr[pencari_kanan] > arr[index_pivot]){ //balik tanda > untuk menjadi Desc, lakukan juga pada baris 26
30              --pencari_kanan;
31          }
32          tukar(&arr[pencari_kiri], &arr[pencari_kanan]);
33
34          if (index_pivot == pencari_kiri){
35              index_pivot = pencari_kanan;
36              ++pencari_kiri;
37          }else if(index_pivot == pencari_kanan){
38              index_pivot = pencari_kiri;
39              --pencari_kanan;
40          }else{
41              ++pencari_kiri;
42              --pencari_kanan;
43          }
44      }
45
46      quick_tengah(arr, arr_size, batas_kiri, index_pivot-1);
47      quick_tengah(arr, arr_size, index_pivot+1, batas_kanan);
48  }
```

Diberikan fungsi/prosedur quick sort dengan partisi Hoare pivot kiri:

```
50  //Pivot kiri, Ascending, Partisi Hoare
51  void quick_kiri(int arr[], int arr_size, int batas_kiri, int batas_kanan){
52      if (batas_kiri >= batas_kanan){
53          return;
54      }
55
56      int index_pivot = batas_kiri;
57      int pencari_kiri = batas_kiri + 1;
58      int pencari_kanan = batas_kanan;
59
60      while(pencari_kiri <= pencari_kanan){
61          for (int i = 0; i < arr_size; ++i){
62              printf("%d ", arr[i]);
63          }
64          printf("\n");
65
66          while(arr[pencari_kiri] < arr[index_pivot] && pencari_kiri <= pencari_kanan){
67              ++pencari_kiri;
68          }
69          while(arr[pencari_kanan] > arr[index_pivot] && pencari_kiri <= pencari_kanan){
70              --pencari_kanan;
71          }
72          if (pencari_kiri <= pencari_kanan){
73              tukar(&arr[pencari_kiri], &arr[pencari_kanan]);
74              ++pencari_kiri;
75              --pencari_kanan;
76          }
77      }
78      tukar(&arr[index_pivot], &arr[pencari_kanan]);
79      for (int i = 0; i < arr_size; ++i){
80          printf("%d ", arr[i]);
81      }
82      printf("\n");
83
84      if (pencari_kanan+1 < batas_kanan){
85          quick_kiri(arr, arr_size, pencari_kanan+1, batas_kanan);
86      }
87      if (batas_kiri < pencari_kanan-1){
88          quick_kiri(arr, arr_size, batas_kiri, pencari_kanan-1);
89      }
90  }
```

Pahami kedua jenis quick sort tersebut!

Temukan, sebutkan, dan jelaskan perbedaan antara kedua quick sort tersebut!