**LAPORAN**



**ANALISIS SORTING**

Nama : Diaz Adha Asri Prakoso

NIM : 0102518007

Program Studi : Teknik Informatika

**UNIVERSITAS AL AZHAR INDONESIA**

**JAKARTA**

**2019**

**Daftar Isi**

[**Bab 1 : Bubble, Selection, dan Insertion Sort** 1](#_Toc9985640)

[**1.1** **Pengertian Sorting** 1](#_Toc9985641)

[**1.1.1** **Bubble Sort** 1](#_Toc9985642)

[**1.1.2** **Selection Sort** 1](#_Toc9985643)

[**1.1.3** **Insertion Sort** 1](#_Toc9985644)

[**1.2** **Deskripsi Tugas** 2](#_Toc9985645)

[**Bab 2 : Source Program** 3](#_Toc9985646)

[**Bab 3 : Analisis Hasil Eksekusi Program** 33](#_Toc9985647)

[**Daftar Pustaka** 35](#_Toc9985648)

# **Bab 1 : Bubble, Selection, dan Insertion Sort**

* 1. **Pengertian Sorting**

Sorting merupakan teknik untuk mengurutkan data yang acak hingga bisa tersusun rapi dari terkecil ke terbesar atau sebaliknya. Ada berbagai macam teknik sorting ini diantaranya bubble sort,insertion sort,selection sort. 

* + 1. **Bubble Sort**

Metode/algoritma pengurutan dengan dengan cara melakukan penukaran data dengan tepat disebelahnya secara terus menerus sampai bisa dipastikan dalam satu iterasi tertentu tidak ada lagi perubahan. Jika tidak ada perubahan berarti data sudah terurut. Disebut pengurutan gelembung karena masing-masing kunci akan dengan lambat menggelembung ke posisinya yang tepat.

* + 1. **Selection Sort**

Sebuah algoritma pengurutan yang secara berulang mencari *item*yang belum terurut dan mencari paling sedikit satu untuk dimasukkan ke dalam lokasi akhir. Metode ini memiliki konsep memilih data yang maksimum/minimum dari suatu kumpulan data larik L, lalu menempatkan data tersebut ke elemen paling akhir atau paling awal sesuai pengurutan yang diinginkan. Data maksimum/minimum yang diperoleh, diasingkan ke tempat lain, dan tidak diikutsertakan pada proses pencarian data maksimum/minimum berikutnya.

Di dalam selection sort memiliki 2 varian seleksi/pengurutan data yaitu maximum sort dan minimum sort.

1.    Maximum sort yaitu mencari data terbesar(X) yang dibandingan dengan index 1 kemudian jika   index1 < data terbesar(X) maka data terbesar menjadi index 1.

2.    Minimum sort yaitu kebalikan dari maximum sort yaitu mencari data terkecil(X) yang   dibandingkan dengan index 1 kemudian jika index1 > data terkecil(X) maka data terkecil menjadi       index1.

* + 1. **Insertion Sort**

Sebuah metode pengurutan data dengan menempatkan setiap elemen data pada pisisinya dengan cara melakukan perbandingan dengan data – data yang ada. Ide algoritma dari metode insertion sort ini dapat dianalogikan sama seperti mengurutkan kartu, dimana jika suatu kartu dipindah tempatkan menurut posisinya, maka kartu yang lain akan bergeser mundur atau maju sesuai kondisi pemindahanan kartu tersebut. Dalam pengurutan data, metode ini dipakai bertujuan untuk menjadikan bagian sisi kiri array terurutkan sampai dengan seluruh array diurutkan.

* 1. **Deskripsi Tugas**
* Eksekusi semua jenis sorting dengan : best, average, dan worst case
* Tampilkan di layer setiap iterasinya untuk dibandingkan tiap jenis sorting
* Dengan Array 10 elemen :

Best case : -5, 2, 16, 23, 44, 68, 71, 90, 101, 150.

Average case : 68, 2, 71, 150, 16, -5, 101, 90, 44, 23.

Worst case : 150, 101, 90, 71, 68, 44, 23, 16, 2, -5.

# **Bab 2 : Source Program**

* **Program Bubble Sort (Best Case)**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121 | #include <stdio.h>  #include <stdlib.h>  /\*\*\*\*\*\*\*\*\*  Program Sorting  Deskripsi : Mengurutkan array A dengan berbagai metode Sorting  IS : Array A telah berisi nilai sembarang  FS : Array A terurut  Dibuat oleh : Diaz Adha Asri Prakoso  Tanggal : 28 Mei 2019  \*\*\*\*\*\*\*\*\*/  //header procedure  void bubblesort (int akhirtakterurut, int A[11]);  main (void)  { //KAMUS  int A[11] = {999, -5, 2, 16, 23, 44, 68, 71, 90, 101, 150, -9999};  int i;  //ALGORITMA  printf ("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");  printf ("Program SORTING Bubble Sort Best Case \n");  printf ("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");    //Menampilkan isi array    printf ("isi array awal : \n");  for (i=1;i<=10;i++)  {  printf ("A[%d] = %d\n",i, A[i]);  }  printf ("\n\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");  printf ("Proses pengurutan \n");  printf ("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n\n");  //Mengurutkan dengan bubble sort  bubblesort (10, A);    //Menampilkan isi array    printf ("Array setelah diurut : \n");  for (i=1;i<=10;i++)  {  printf ("A[%d] = %d\n",i, A[i]);  }      printf ("\n\n");  system ("PAUSE");  }  void bubblesort (int akhirtakterurut, int A[11])  {  int Aw, Ak, i, j, temp, iterasi\_luar, iterasi\_dalam;  // Deskripsi : Mengurutkan array A dari elemen ke-1 hingga ke-AkhirTakTerurut  // secara ascending dengan metode Bubble Sort  // IS : Array A telah berisi nilai sembarang  // FS : Array A terurut  // Dibuat oleh : Diaz Adha Asri Prakoso  // Tanggal : 28 Mei 2019  //algoritma    Aw = 1;  Ak = akhirtakterurut;  iterasi\_luar = 0;  iterasi\_dalam = 0;  while (Ak>=2)  {  printf ("---->Cari nilai yang tepat untuk lokasi terbelakang A[%d] \n", Ak);  //Periksa urutan bilangan yang bertetangga  i = Aw;  j = i + 1; //j adalah tetangga  while (j<=Ak)  {  printf ("Bandingkan A[%d] = %d dan A[%d] = %d", i, A[i], j, A[j]);  if (A[i]>A[j])  {  printf ("--->Tukar\n");  //swap  temp = A[i];  A[i] = A[j];  A[j] = temp;  iterasi\_dalam = iterasi\_dalam + 1;  //tampilkan array saat ini  for (i=1;i<=10;i++)  {  printf ("%d ", A[i]);  }  printf ("\n\n");  }  else  {  printf ("--->Sudah benar urutannya, maka biarkan saja\n\n");  }  i = j;  j = i + 1;    }  Ak = Ak - 1;  iterasi\_luar = iterasi\_luar + 1;  }  printf ("\n\n");  printf ("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");  printf ("Hasil iterasi : \n");  printf ("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");    printf ("Iterasi Luar : %d\n", iterasi\_luar);  printf ("Iterasi Dalam : %d\n\n", iterasi\_dalam);  } |

* **Program Bubble Sort (Average Case)**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66 | #include <stdio.h>  #include <stdlib.h>  /\*\*\*\*\*\*\*\*\*  Program Sorting  Deskripsi : Mengurutkan array A dengan berbagai metode Sorting  IS : Array A telah berisi nilai sembarang  FS : Array A terurut  Dibuat oleh : Diaz Adha Asri Prakoso  Tanggal : 28 Mei 2019  \*\*\*\*\*\*\*\*\*/  //header procedure  void bubblesort (int akhirtakterurut, int A[11]);  main (void)  { //KAMUS  int A[11] = {999, 68, 2, 71, 150, 16, -5, 101, 90, 44, 23, -9999};  int i;  //ALGORITMA  printf ("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");  printf ("Program SORTING Bubble Sort Average Case \n");  printf ("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");    //Menampilkan isi array    printf ("isi array awal : \n");  for (i=1;i<=10;i++)  {  printf ("A[%d] = %d\n",i, A[i]);  }  printf ("\n\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");  printf ("Proses pengurutan \n");  printf ("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n\n");  //Mengurutkan dengan bubble sort  bubblesort (10, A);    //Menampilkan isi array    printf ("Array setelah diurut : \n");  for (i=1;i<=10;i++)  {  printf ("A[%d] = %d\n",i, A[i]);  }  printf ("\n\n");  system ("PAUSE");  }  void bubblesort (int akhirtakterurut, int A[11])  {  int Aw, Ak, i, j, temp, iterasi\_luar, iterasi\_dalam;  // Deskripsi : Mengurutkan array A dari elemen ke-1 hingga ke-AkhirTakTerurut  // secara ascending dengan metode Bubble Sort  // IS : Array A telah berisi nilai sembarang  // FS : Array A terurut  // Dibuat oleh : Diaz Adha Asri Prakoso  // Tanggal : 28 Mei 2019  //algoritma    Aw = 1;  Ak = akhirtakterurut;  iterasi\_luar = 0;  iterasi\_dalam = 0;  while (Ak>=2)  {  printf ("---->Cari nilai yang tepat untuk lokasi terbelakang A[%d] \n", Ak);  //Periksa urutan bilangan yang bertetangga  i = Aw;  j = i + 1; //j adalah tetangga  while (j<=Ak)  {  printf ("Bandingkan A[%d] = %d dan A[%d] = %d", i, A[i], j, A[j]);  if (A[i]>A[j])  {  printf ("--->Tukar\n");  //swap  temp = A[i];  A[i] = A[j];  A[j] = temp;  iterasi\_dalam = iterasi\_dalam + 1;  //tampilkan array saat ini  for (i=1;i<=10;i++)  {  printf ("%d ", A[i]);  }  printf ("\n\n");  }  else  {  printf ("--->Sudah benar urutannya, maka biarkan saja\n\n");  }  i = j;  j = i + 1;    }  Ak = Ak - 1;  iterasi\_luar = iterasi\_luar + 1;  }  printf ("\n\n");  printf ("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");  printf ("Hasil iterasi : \n");  printf ("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");    printf ("Iterasi Luar : %d\n", iterasi\_luar);  printf ("Iterasi Dalam : %d\n\n", iterasi\_dalam);  } |

* **Program Bubble Sort (Worst Case)**

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  /\*\*\*\*\*\*\*\*\*  Program Sorting  Deskripsi : Mengurutkan array A dengan berbagai metode Sorting  IS : Array A telah berisi nilai sembarang  FS : Array A terurut  Dibuat oleh : Diaz Adha Asri Prakoso  Tanggal : 28 Mei 2019  \*\*\*\*\*\*\*\*\*/  //header procedure  void bubblesort (int akhirtakterurut, int A[11]);  main (void)  { //KAMUS  int A[11] = {999, 150, 101, 90, 71, 68, 44, 23, 16, 2, -5, -9999};  int i;  //ALGORITMA  printf ("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");  printf ("Program SORTING Bubble Sort Worst Case \n");  printf ("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");  //Menampilkan isi array  printf ("isi array awal : \n");  for (i=1;i<=10;i++)  {  printf ("A[%d] = %d\n",i, A[i]);  }  printf ("\n\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");  printf ("Proses pengurutan \n");  printf ("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n\n");  //Mengurutkan dengan bubble sort  bubblesort (10, A);  //Menampilkan isi array  printf ("Array setelah diurut : \n");  for (i=1;i<=10;i++)  {  printf ("A[%d] = %d\n",i, A[i]);  }  printf ("\n\n");  system ("PAUSE");  }  void bubblesort (int akhirtakterurut, int A[11])  {  int Aw, Ak, i, j, temp, iterasi\_luar, iterasi\_dalam;  // Deskripsi : Mengurutkan array A dari elemen ke-1 hingga ke-AkhirTakTerurut  // secara ascending dengan metode Bubble Sort  // IS : Array A telah berisi nilai sembarang  // FS : Array A terurut  // Dibuat oleh : Diaz Adha Asri Prakoso  // Tanggal : 28 Mei 2019  //algoritma  Aw = 1;  Ak = akhirtakterurut;  iterasi\_luar = 0;  iterasi\_dalam = 0;  while (Ak>=2)  {  printf ("---->Cari nilai yang tepat untuk lokasi terbelakang A[%d] \n", Ak);  //Periksa urutan bilangan yang bertetangga  i = Aw;  j = i + 1; //j adalah tetangga  while (j<=Ak)  {  printf ("Bandingkan A[%d] = %d dan A[%d] = %d", i, A[i], j, A[j]);  if (A[i]>A[j])  {  printf ("--->Tukar\n");  //swap  temp = A[i];  A[i] = A[j];  A[j] = temp;  iterasi\_dalam = iterasi\_dalam + 1;  //tampilkan array saat ini  for (i=1;i<=10;i++)  {  printf ("%d ", A[i]);  }  printf ("\n\n");  }  else  {  printf ("--->Sudah benar urutannya, maka biarkan saja\n\n");  }  i = j;  j = i + 1;  }  Ak = Ak - 1;  iterasi\_luar = iterasi\_luar + 1;  }  printf ("\n\n");  printf ("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");  printf ("Hasil iterasi : \n");  printf ("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");  printf ("Iterasi Luar : %d\n", iterasi\_luar);  printf ("Iterasi Dalam : %d\n\n", iterasi\_dalam);  } |

* **Program Selection Sort (Best Case)**

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  /\*\*\*\*\*\*\*\*\*  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Program Sorting  Deskripsi : Mengurutkan array A dengan berbagai metode Sorting  IS : Array A telah berisi nilai sembarang  FS : Array A terurut  Dibuat oleh : Diaz Adha Asri Prakoso  Tanggal : 28 Mei 2019  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  //header procedure  void selectionsort (int akhirtakterurut, int A[11]);  main (void)  { //KAMUS  int A[11] = {999, -5, 2, 16, 23, 44, 68, 71, 90, 101, 150, -9999};  int i;  //ALGORITMA  printf ("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");  printf ("Program SORTING Selection Sort Best Case \n");  printf ("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");    //Menampilkan isi array    printf ("isi array awal : \n");  for (i=1;i<=10;i++)  {  printf ("A[%d] = %d\n",i, A[i]);  }  printf ("\n\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");  printf ("Proses pengurutan \n");  printf ("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n\n");  //Mengurutkan dengan selection sort  selectionsort (10, A);    //Menampilkan isi array    printf ("Setelah diurut : \n");  for (i=1;i<=10;i++)  {  printf ("A[%d] = %d\n",i, A[i]);  }  printf ("\n\n");  system ("PAUSE");  }  void selectionsort (int akhirtakterurut, int A[11])  {  // Deskripsi : Mengurutkan array A dari elemen ke-1 hingga ke-AkhirTakTerurut  // secara ascending dengan metode Selection Sort  // IS : Array A telah berisi nilai sembarang  // FS : Array A terurut  // Dibuat oleh : Diaz Adha Asri Prakoso  // Tanggal : 28 Mei 2019  int aw, ak, i, j, indeks, MAX, iterasi\_luar, iterasi\_dalam, temp;      ak = akhirtakterurut;  iterasi\_luar = 0;  iterasi\_dalam = 0;  while (ak >= 2) //Lakukan pengulangan hingga tinggal 2 elemen  {  printf ("Sekarang mencari nilai yang tepat untuk lokasi terbelakang A[%d] \n", ak);  //Cari MAX di ruang yang belum terurut, yaitu dari aw hingga ak  aw = 1;  MAX = aw; //MAX diisi oleh nilai array pertama  printf ("Mula-mula MAX[%d] = %d \n", aw, A[MAX]);  aw = aw + 1; //mulai bandingkan dengan elemen ke 2  while (aw <= ak)  {  //bandingkan MAX dan A[i]  if (A[aw] > A[MAX])  {  MAX = aw;  iterasi\_dalam++;  };    printf ("Banding dg elemen ke-%d : MAX = A[%d] = %d \n", aw, MAX, A[MAX]);  aw++;    }  printf ("\n");    //Taruh nilai MAX di akhir array yang belum terurut  temp = A[ak];  A[ak] = A[MAX];  A[MAX] = temp;    for (i=1; i<=10; i++)  {  printf ("%d ", A[i]);  }  printf ("\n");    iterasi\_luar++;  ak = ak - 1;  }  printf ("\n\n");  printf ("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");  printf ("Hasil iterasi : \n");  printf ("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");    printf ("Iterasi Luar : %d\n", iterasi\_luar);  printf ("Iterasi Dalam : %d\n", iterasi\_dalam);  } |

* **Program Selection Sort (Average Case)**

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  /\*\*\*\*\*\*\*\*\*  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Program Sorting  Deskripsi : Mengurutkan array A dengan berbagai metode Sorting  IS : Array A telah berisi nilai sembarang  FS : Array A terurut  Dibuat oleh : Diaz Adha Asri Prakoso  Tanggal : 28 Mei 2019  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  //header procedure  void selectionsort (int akhirtakterurut, int A[11]);  main (void)  { //KAMUS  int A[11] = {999, 68, 2, 71, 150, 16, -5, 101, 90, 44, 23, -9999};  int i;  //ALGORITMA  printf ("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");  printf ("Program SORTING Selection Sort Average Case \n");  printf ("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");    //Menampilkan isi array    printf ("isi array awal : \n");  for (i=1;i<=10;i++)  {  printf ("A[%d] = %d\n",i, A[i]);  }  printf ("\n\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");  printf ("Proses pengurutan \n");  printf ("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n\n");  //Mengurutkan dengan selection sort  selectionsort (10, A);    //Menampilkan isi array    printf ("Setelah diurut : \n");  for (i=1;i<=10;i++)  {  printf ("A[%d] = %d\n",i, A[i]);  }  printf ("\n\n");  system ("PAUSE");  }  void selectionsort (int akhirtakterurut, int A[11])  {  // Deskripsi : Mengurutkan array A dari elemen ke-1 hingga ke-AkhirTakTerurut  // secara ascending dengan metode Selection Sort  // IS : Array A telah berisi nilai sembarang  // FS : Array A terurut  // Dibuat oleh : Diaz Adha Asri Prakoso  // Tanggal : 28 Mei 2019  int aw, ak, i, j, indeks, MAX, iterasi\_luar, iterasi\_dalam, temp;      ak = akhirtakterurut;  iterasi\_luar = 0;  iterasi\_dalam = 0;  while (ak >= 2) //Lakukan pengulangan hingga tinggal 2 elemen  {  printf ("Sekarang mencari nilai yang tepat untuk lokasi terbelakang A[%d] \n", ak);  //Cari MAX di ruang yang belum terurut, yaitu dari aw hingga ak  aw = 1;  MAX = aw; //MAX diisi oleh nilai array pertama  printf ("Mula-mula MAX[%d] = %d \n", aw, A[MAX]);  aw = aw + 1; //mulai bandingkan dengan elemen ke 2  while (aw <= ak)  {  //bandingkan MAX dan A[i]  if (A[aw] > A[MAX])  {  MAX = aw;  iterasi\_dalam++;  };    printf ("Banding dg elemen ke-%d : MAX = A[%d] = %d \n", aw, MAX, A[MAX]);  aw++;    }  printf ("\n");    //Taruh nilai MAX di akhir array yang belum terurut  temp = A[ak];  A[ak] = A[MAX];  A[MAX] = temp;  for (i=1; i<=10; i++)  {  printf ("%d ", A[i]);  }  printf ("\n");    iterasi\_luar++;  ak = ak - 1;  }    printf ("\n\n");  printf ("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");  printf ("Hasil iterasi : \n");  printf ("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");    printf ("Iterasi Luar : %d\n", iterasi\_luar);  printf ("Iterasi Dalam : %d\n", iterasi\_dalam);  } |

* **Program Selection Sort (Worst Case)**

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  /\*\*\*\*\*\*\*\*\*  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Program Sorting  Deskripsi : Mengurutkan array A dengan berbagai metode Sorting  IS : Array A telah berisi nilai sembarang  FS : Array A terurut  Dibuat oleh : Diaz Adha Asri Prakoso  Tanggal : 28 Mei 2019  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  //header procedure  void selectionsort (int akhirtakterurut, int A[11]);  main (void)  { //KAMUS  int A[11] = {999, 150, 101, 90, 71, 68, 44, 23, 16, 2, -5, -9999};  int i;  //ALGORITMA  printf ("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");  printf ("Program SORTING Selection Sort Worst Case \n");  printf ("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");    //Menampilkan isi array    printf ("isi array awal : \n");  for (i=1;i<=10;i++)  {  printf ("A[%d] = %d\n",i, A[i]);  }  printf ("\n\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");  printf ("Proses pengurutan \n");  printf ("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n\n");  //Mengurutkan dengan selection sort  selectionsort (10, A);    //Menampilkan isi array    printf ("Setelah diurut : \n");  for (i=1;i<=10;i++)  {  printf ("A[%d] = %d\n",i, A[i]);  }  printf ("\n\n");  system ("PAUSE");  }  void selectionsort (int akhirtakterurut, int A[11])  {  // Deskripsi : Mengurutkan array A dari elemen ke-1 hingga ke-AkhirTakTerurut  // secara ascending dengan metode Selection Sort  // IS : Array A telah berisi nilai sembarang  // FS : Array A terurut  // Dibuat oleh : Diaz Adha Asri Prakoso  // Tanggal : 28 Mei 2019  int aw, ak, i, j, indeks, MAX, iterasi\_luar, iterasi\_dalam, temp;  ak = akhirtakterurut;  iterasi\_luar = 0;  iterasi\_dalam = 0;  while (ak >= 2) //Lakukan pengulangan hingga tinggal 2 elemen  {  printf ("Sekarang mencari nilai yang tepat untuk lokasi terbelakang A[%d] \n", ak);  //Cari MAX di ruang yang belum terurut, yaitu dari aw hingga ak  aw = 1;  MAX = aw; //MAX diisi oleh nilai array pertama  printf ("Mula-mula MAX[%d] = %d \n", aw, A[MAX]);  aw = aw + 1; //mulai bandingkan dengan elemen ke 2  while (aw <= ak)  {  //bandingkan MAX dan A[i]  if (A[aw] > A[MAX])  {  MAX = aw;  iterasi\_dalam++;  };  printf ("Banding dg elemen ke-%d : MAX = A[%d] = %d \n", aw, MAX, A[MAX]);  aw++;    }  printf ("\n");    //Taruh nilai MAX di akhir array yang belum terurut  temp = A[ak];  A[ak] = A[MAX];  A[MAX] = temp;    for (i=1; i<=10; i++)  {  printf ("%d ", A[i]);  }  printf ("\n");    iterasi\_luar++;  ak = ak - 1;  }  printf ("\n\n");  printf ("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");  printf ("Hasil iterasi : \n");  printf ("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");    printf ("Iterasi Luar : %d\n", iterasi\_luar);  printf ("Iterasi Dalam : %d\n", iterasi\_dalam);  } |

* **Program Insertion Sort (Best Case)**

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Deskripsi : Mengurutkan array dari elemen ke-1 hingga akhir menggunakan metode insertion sort  IS : Array telah berisi nilai  FS : Array telah terurut  Dibuat oleh :  Nama : Diaz Adha Asri Prakoso  Tanggal : 25 Mei 2019  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void insertion\_sort (int a, int A[100]);  void main (void)  //KAMUS  { int x;  int A[11] = {999, -5, 2, 16, 23, 44, 68, 71, 90, 101, 150, -9999};  //ALGORITMA  printf ("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");  printf ("Program SORTING Insertion Sort Best Case \n");  printf ("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");  printf ("Isi array sebelum terurut adalah : ");  printf ("\n");  for (x=1;x<=10;x++)  {  printf ("A[%d] = %d\n", x, A[x]);  }  printf ("\n\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");  printf ("Proses pengurutan \n");  printf ("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n\n");  insertion\_sort (10, &A);    printf ("Isi array setelah terurut adalah : ");  printf ("\n");  for (x=1;x<=10;x++)  {  printf ("A[%d] = %d\n", x, A[x]);  }  }  void insertion\_sort (int a, int A[100])  {//KAMUS  int ind\_awal, ind\_akhir, i, i2, temp, iterasi\_luar, iterasi\_dalam, x;    ind\_awal = 1;  ind\_akhir = a;  iterasi\_dalam = 0;  iterasi\_luar = 0;  i = 2;  while (i<=ind\_akhir)  {  i2 = i;  while (i2>ind\_awal && A[i2]<A[i2-1])  {  printf ("Bandingkan A[%d] = %d dengan A[%d] = %d\n", i2, A[i2], i2-1, A[i2-1]);  temp = A[i2-1];  A[i2-1] = A[i2];  A[i2] = temp;  //Isi array setelah ditukar  for (x=1;x<=10;x++)  {  printf ("%d ", A[x]);  }  printf ("\n\n");  i2 = i2 - 1;  iterasi\_dalam++;  }  i = i + 1;  iterasi\_luar++;  }  printf ("\n\n");  printf ("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");  printf ("Hasil iterasi : \n");  printf ("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");  printf ("Iterasi Luar : %d\n", iterasi\_luar);  printf ("Iterasi Dalam : %d\n", iterasi\_dalam);  } |

* **Program Insertion Sort (Average Case)**

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Deskripsi : Mengurutkan array dari elemen ke-1 hingga akhir menggunakan metode insertion sort  IS : Array telah berisi nilai  FS : Array telah terurut  Dibuat oleh :  Nama : Diaz Adha Asri Prakoso  Tanggal : 25 Mei 2019  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void insertion\_sort (int a, int A[100]);  void main (void)  //KAMUS  { int x;  int A[11] = {999, 68, 2, 71, 150, 16, -5, 101, 90, 44, 23, -9999};  //ALGORITMA  printf ("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");  printf ("Program SORTING Insertion Sort Average Case \n");  printf ("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");  printf ("Isi array sebelum terurut adalah : ");  printf ("\n");  for (x=1;x<=10;x++)  {  printf ("A[%d] = %d\n", x, A[x]);  }  printf ("\n\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");  printf ("Proses pengurutan \n");  printf ("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n\n");  insertion\_sort (10, &A);    printf ("Isi array setelah terurut adalah : ");  printf ("\n");  for (x=1;x<=10;x++)  {  printf ("A[%d] = %d\n", x, A[x]);  }  }  void insertion\_sort (int a, int A[100])  {//KAMUS  int ind\_awal, ind\_akhir, i, i2, temp, iterasi\_luar, iterasi\_dalam, x;    ind\_awal = 1;  ind\_akhir = a;  iterasi\_dalam = 0;  iterasi\_luar = 0;  i = 2;  while (i<=ind\_akhir)  {  i2 = i;  while (i2>ind\_awal && A[i2]<A[i2-1])  {  printf ("Bandingkan A[%d] = %d dengan A[%d] = %d\n", i2, A[i2], i2-1, A[i2-1]);  temp = A[i2-1];  A[i2-1] = A[i2];  A[i2] = temp;  //Isi array setelah ditukar  for (x=1;x<=10;x++)  {  printf ("%d ", A[x]);  }  printf ("\n\n");  i2 = i2 - 1;  iterasi\_dalam++;  }  i = i + 1;  iterasi\_luar++;  }  printf ("\n\n");  printf ("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");  printf ("Hasil iterasi : \n");  printf ("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");  printf ("Iterasi Luar : %d\n", iterasi\_luar);  printf ("Iterasi Dalam : %d\n", iterasi\_dalam);  } |

* **Program Insertion Sort (Worst Case)**

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Deskripsi : Mengurutkan array dari elemen ke-1 hingga akhir menggunakan metode insertion sort  IS : Array telah berisi nilai  FS : Array telah terurut  Dibuat oleh :  Nama : Diaz Adha Asri Prakoso  Tanggal : 25 Mei 2019  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void insertion\_sort (int a, int A[100]);  void main (void)  //KAMUS  { int x;  int A[11] = {999, 150, 101, 90, 71, 68, 44, 23, 16, 2, -5, -9999};  //ALGORITMA  printf ("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");  printf ("Program SORTING Insertion Sort Worst Case \n");  printf ("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");  printf ("Isi array sebelum terurut adalah : ");  printf ("\n");  for (x=1;x<=10;x++)  {  printf ("A[%d] = %d\n", x, A[x]);  }  printf ("\n\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");  printf ("Proses pengurutan \n");  printf ("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n\n");  insertion\_sort (10, &A);    printf ("Isi array setelah terurut adalah : ");  printf ("\n");  for (x=1;x<=10;x++)  {  printf ("A[%d] = %d\n", x, A[x]);  }  }  void insertion\_sort (int a, int A[100])  {//KAMUS  int ind\_awal, ind\_akhir, i, i2, temp, iterasi\_luar, iterasi\_dalam, x;    ind\_awal = 1;  ind\_akhir = a;  iterasi\_dalam = 0;  iterasi\_luar = 0;  i = 2;  while (i<=ind\_akhir)  {  i2 = i;  while (i2>ind\_awal && A[i2]<A[i2-1])  {  printf ("Bandingkan A[%d] = %d dengan A[%d] = %d\n", i2, A[i2], i2-1, A[i2-1]);  temp = A[i2-1];  A[i2-1] = A[i2];  A[i2] = temp;  //Isi array setelah ditukar  for (x=1;x<=10;x++)  {  printf ("%d ", A[x]);  }  printf ("\n\n");  i2 = i2 - 1;  iterasi\_dalam++;  }  i = i + 1;  iterasi\_luar++;  }  printf ("\n\n");  printf ("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");  printf ("Hasil iterasi : \n");  printf ("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");  printf ("Iterasi Luar : %d\n", iterasi\_luar);  printf ("Iterasi Dalam : %d\n", iterasi\_dalam);  } |

# **Bab 3 : Analisis Hasil Eksekusi Program**

**Panjang Array = 10**

## **Tabel 3.1 : Hasil eksekusi 3 case program Bubble Sort**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Best Case | Average Case | Worst Case |
| Iterasi Luar | 9 | 9 | 9 |
| Iterasi Dalam | 0 | 23 | 45 |

## **Tabel 3.2 : Hasil eksekusi 3 case program Selection Sort**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Best Case | Average Case | Worst Case |
| Iterasi Luar | 9 | 9 | 9 |
| Iterasi Dalam | 45 | 15 | 20 |

## **Tabel 3.3 : Hasil eksekusi 3 case program Insertion Sort**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Best Case | Average Case | Worst Case |
| Iterasi Luar | 9 | 9 | 9 |
| Iterasi Dalam | 0 | 23 | 45 |

**Kesimpulan :**

**Best Case :** Algoritma bubble sort dan insertion sort menjadi algoritma yang paling cepat dan efektif karena hanya melakukan iterasi luar sebanyak 9 dan iterasi dalam sebanyak 0. Sedangkan pada algoritma selection sort menghasilkan iterasi luar sebanyak 9 dan iterasi dalam sebanyak 45. Hal tersebut bisa terjadi karena pada best case (array terurut naik) algoritma bubble sort dan insertion sort tidak sama sekali melakukan penukaran posisi sehinga tidak menghasilkan iterasi dalam sedangkan, pada selection sort akan selalu melakukan iterasi dalam karena metode selection sort mencari nilai max terlebih dahulu.

**Average Case :** Algoritma selection menjadi algoritma yang paling cepat dan efektif karena hanya melakukan iterasi luar sebanyak 9 dan iterasi dalam sebanyak 15. Sedangkan pada algoritma bubble sort dan insertion sort menghasilkan iterasi luar sebanyak 9 dan iterasi dalam sebanyak 23. Hal tersebut bisa terjadi karena pada average case (array acak tidak terurut) algoritma selection sort melakukanya dengan mencari nilai max terlebih dahulu lalu nilai tersebut ditukar pada elemen yang paling belakang sehingga nilai iterasi dalam bisa lebih sedikit dibandingkan dengan bubble sort dan insertion sort yang harus membandingkan satu per satu isi elemen.

**Worst Case :** Algoritma selection menjadi algoritma yang paling cepat dan efektif karena hanya melakukan iterasi luar sebanyak 9 dan iterasi dalam sebanyak 20. Sedangkan pada algoritma bubble sort dan insertion sort menghasilkan iterasi luar sebanyak 9 dan iterasi dalam sebanyak 45. Hal tersebut bisa terjadi karena pada worst case (array terurut menurun) algoritma selection sort melakukanya dengan mencari nilai max terlebih dahulu lalu nilai tersebut ditukar pada elemen yang paling belakang sehingga nilai iterasi dalam bisa lebih sedikit dibandingkan dengan bubble sort dan insertion sort yang harus membandingkan satu per satu isi elemen.

# **Daftar Pustaka**

<http://kukuhapriyanto.blogspot.com/2017/06/sorting-dalam-bahasa-c.html>

<https://www.academia.edu/6797207/Analisis_Algoritma_Bubble_Sort>