PRAKTIKUM 11
BUBBLE SORT DAN SHELL SORT

Dosen Pengampu :
Ibu Entin Martiana Kusumaningtyas S.Kom, M.Kom

Disusun Oleh :

ADE HAFIS RABBANI
1 D3 IT A
3122500001

Politeknik Elektronika Negeri Surbaya
Program Studi Teknik Informatika

Mei 2023

DAFTAR ISI

D. PERCOBAAN

Percobaan 1 : Bubble sort secara ascending.

- **Program**

```c
#include<stdio.h>

#include<stdlib.h>

#include<time.h>

#define MAX 20
void BubbleSort(int arr[]) {
  int i, j, temp;
  for (i = 0; i < MAX - 1; i++) {
    for (j = 0; j < MAX - i - 1; j++) {
      if (arr[j] > arr[j + 1]) {
        temp = arr[j + 1];
        arr[j + 1] = arr[j];
        arr[j] = temp;

      }
    }
  }
}
void main() {
  int data_awal[MAX], data_urut[MAX];
  int i;
  long k1, k2;
  printf("Sebelum pengurutan : \n");
  for (i = 0; i < MAX; i++) {
    srand(time(NULL) * (i + 1));
    data_awal[i] = rand() % 100 + 1;
    printf("%d ", data_awal[i]);
  }
  printf("\nSetelah pengurutan : \n");
  for (i = 0; i < MAX; i++)
    data_urut[i] = data_awal[i];
  time( & k1);
  BubbleSort(data_urut);
  time( & k2);
  for (i = 0; i < MAX; i++)
    printf("%d ", data_urut[i]);
  printf("\nWaktu = %ld\n", k2 - k1);
}
```

- **Output**

```
Sebelum pengurutan :
72 5 38 71 4 37 2 35 68 1 34 67 32 65 98 31 64 97 30 95
Setelah pengurutan :
0 1 2 4 5 30 31 32 34 35 37 38 64 65 67 68 71 95 97 98
Waktu = 1683646894
```

- **Analisa**

  Program di atas merupakan implementasi algoritma Bubble Sort untuk mengurutkan array of integer sejumlah MAX yang di-generate secara acak menggunakan fungsi rand().

  Pada fungsi BubbleSort, dilakukan perulangan nested for untuk membandingkan setiap pasangan elemen dalam array dan melakukan swap jika nilai elemen sebelah kanan lebih kecil dari sebelah kirinya.

  Pada fungsi main, terdapat deklarasi dua array yaitu data_awal dan data_urut yang masing-masing berisi MAX elemen. Selain itu, juga dilakukan inisialisasi elemen-elemen array data_awal dengan angka acak menggunakan fungsi rand().

  Kemudian, array data_urut diisi dengan nilai-nilai dari data_awal dan dilakukan pengukuran waktu eksekusi BubbleSort pada data_urut menggunakan fungsi time(). Hasil pengurutan kemudian ditampilkan beserta waktu eksekusinya.

  Program ini masih memiliki beberapa kelemahan, seperti tidak adanya validasi untuk memastikan bahwa angka acak yang di-generate benar-benar unik. Selain itu, penggunaan time() tidak menjamin akurasi dalam mengukur waktu eksekusi algoritma. Program juga belum diuji coba dengan kasus uji yang berbeda-beda.

Percobaan 2 : Bubble sort secara ascending dengan flag

- **Program**

```c
#include<stdio.h>
#include <stdbool.h>
#include<stdlib.h>
#include<time.h>

#define MAX 20
void BubbleSortFlag(int arr[]) {
  int i = 0, j, temp;
  bool did_swap = true;
  while (i < MAX - 1 && did_swap) {
```

```
    for (j = 0; j < MAX - i - 1; j++) {
      did_swap = false;
      if (arr[j] > arr[j + 1]) {
        temp = arr[j + 1];
        arr[j + 1] = arr[j];
        arr[j] = temp;
        did_swap = true;
      }
    }
    i++;
  }
}
void main() {
  int data_awal[MAX], data_urut[MAX];
  int i;
  long k1, k2;
  printf("Sebelum pengurutan : \n");
  for (i = 0; i < MAX; i++) {
    srand(time(NULL) * (i + 1));
    data_awal[i] = rand() % 100 + 1;
    printf("%d ", data_awal[i]);
  }
  printf("\nSetelah pengurutan : \n");
  for (i = 0; i < MAX; i++)
    data_urut[i] = data_awal[i];
  time( & k1);
  BubbleSortFlag(data_urut);
  time( & k2);
  for (i = 0; i < MAX; i++)
    printf("%d ", data_urut[i]);
  printf("\nWaktu = %ld\n", k2 - k1);
}
```

- **Output**

```
-pid=11m1icdh.4g1   --dbgExe=C:\MinGW\bin\gdb.exe   --interpreter=mi
Sebelum pengurutan :
17 96 74 52 31 41 19 98 76 54 33 43 21 100 78 56 35 45 23 2
Setelah pengurutan :
0 2 19 21 23 31 33 35 41 43 45 52 54 56 74 76 78 96 98 100
Waktu = 1683647061
```

- **Analisa**

  Program tersebut merupakan sebuah implementasi algoritma sorting Bubble Sort dengan menggunakan flag.

Pada awal program, terdapat definisi dari konstanta MAX dengan nilai 20, serta terdapat dua buah fungsi utama yaitu BubbleSortFlag dan main.

Fungsi BubbleSortFlag digunakan untuk mengurutkan array dengan algoritma Bubble Sort yang diimplementasikan dengan menggunakan flag did_swap untuk mengoptimalkan performa program. Sedangkan pada fungsi main, terdapat inisialisasi dua buah array yaitu data_awal dan data_urut dengan ukuran MAX yang sama.

Selanjutnya, program akan melakukan generate bilangan acak dengan menggunakan fungsi rand() pada data_awal dan menampilkannya pada layar. Kemudian, data_urut diisi dengan nilai yang sama dengan data_awal sebelum diurutkan.

Setelah itu, program akan menjalankan fungsi BubbleSortFlag untuk mengurutkan data_urut secara ascending. Lalu, waktu mulai diambil sebelum pengurutan dimulai (k1), dan waktu selesai diambil setelah pengurutan selesai (k2).

Terakhir, program menampilkan data_urut yang telah diurutkan dan selisih waktu pengurutan dihitung dan ditampilkan pada layar.

Percobaan 3 : Shell sort secara ascending.

- **Program**

```c
#include<stdio.h>

#include<stdlib.h>

#include<time.h>

#include <stdbool.h>

#define MAX 20
void ShellSort(int arr[]) {
  int i, jarak, temp;
  bool did_swap = true;
  jarak = MAX;
  while (jarak > 1) {
    jarak = jarak / 2;
    did_swap = true;
    while (did_swap) {
```

```
      did_swap = false;
      i = 0;
      while (i < (MAX - jarak)) {
        if (arr[i] > arr[i + jarak]) {
          temp = arr[i];
          arr[i] = arr[i + jarak];
          arr[i + jarak] = temp;
          did_swap = true;

        }
        i++;
      }
    }
  }
}
void main() {
  int data_awal[MAX], data_urut[MAX];
  int i;
  long k1, k2;
  printf("Sebelum pengurutan : \n");
  for (i = 0; i < MAX; i++) {
    srand(time(NULL) * (i + 1));
    data_awal[i] = rand() % 100 + 1;
    printf("%d ", data_awal[i]);
  }
  printf("\nSetelah pengurutan : \n");
  for (i = 0; i < MAX; i++)
    data_urut[i] = data_awal[i];
  time( & k1);
  ShellSort(data_urut);
  time( & k2);
  for (i = 0; i < MAX; i++)
    printf("%d ", data_urut[i]);
  printf("\nWaktu = %ld\n", k2 - k1);
}
```

- **Output**

```
Sebelum pengurutan :
92 44 97 50 2 87 40 92 45 98 50 35 87 40 93 45 30 83 35 88
Setelah pengurutan :
0 2 30 35 35 40 40 44 45 45 50 50 83 87 87 88 92 93 97 98
Waktu = 1683647145
```

- **Analisa**

  Program tersebut merupakan implementasi algoritma sorting Shell Sort.

  Pada awal program, terdapat definisi dari konstanta MAX dengan nilai 20, serta terdapat dua buah fungsi utama yaitu ShellSort dan main.

Fungsi ShellSort digunakan untuk mengurutkan array dengan algoritma Shell Sort, dimana terdapat penggunaan jarak yang menurun secara bertahap dari MAX/2 hingga 1 untuk melakukan penukaran nilai antara dua elemen pada array. Selain itu, pada fungsi ini juga digunakan flag did_swap untuk mengoptimalkan performa program.

Pada fungsi main, terdapat inisialisasi dua buah array yaitu data_awal dan data_urut dengan ukuran MAX yang sama. Selanjutnya, program akan melakukan generate bilangan acak dengan menggunakan fungsi rand() pada data_awal dan menampilkannya pada layar. Kemudian, data_urut diisi dengan nilai yang sama dengan data_awal sebelum diurutkan.

Setelah itu, program akan menjalankan fungsi ShellSort untuk mengurutkan data_urut secara ascending. Lalu, waktu mulai diambil sebelum pengurutan dimulai (k1), dan waktu selesai diambil setelah pengurutan selesai (k2).

Terakhir, program menampilkan data_urut yang telah diurutkan dan selisih waktu pengurutan dihitung dan ditampilkan pada layar.

E. LATIHAN

1. Dari percobaan 1 tambahkan fungsi untuk melakukan pengurutan bubble sort secara descending.
   Jawab :
   - **Program**

```c
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#define MAX 20
void BubbleSort(int arr[])
{
    int i, j, temp;
        for(i=0; i<MAX-1; i++){
        for(j=0; j<MAX-i-1; j++){
            if(arr[j] < arr[j+1]){
                temp = arr[j+1];
                arr[j+1] = arr[j];
                arr[j]= temp;

            }
        }
    }
}

void main()
{
    int data_awal[MAX], data_urut[MAX];
    int i;
    long k1, k2;
    printf("Sebelum pengurutan : \n");
    for(i=0; i<MAX; i++){
        srand(time(NULL) * (i+1));
        data_awal[i] = rand() % 100 + 1;
        printf("%d ", data_awal[i]);
    }
    printf("\nSetelah pengurutan : \n");
    for(i=0; i<MAX; i++)
        data_urut[i] = data_awal[i];
    time(&k1);
    BubbleSort(data_urut);
    time(&k2);
    for(i=0; i<MAX; i++)
        printf("%d ", data_urut[i]);
    printf("\nWaktu = %ld\n", k2-k1);
}
```

- **Output**

```
Sebelum pengurutan :
64 21 78 3 60 17 42 99 56 81 38 96 53 78 35 92 17 74 31 56
Setelah pengurutan :
99 96 92 81 78 78 74 60 56 56 53 42 38 35 31 21 17 17 3 0
Waktu = 1683642308
PS E:\Kuliah\Sems2\ASD\Praktikum11>
```

- **Analisa**

  Program di atas merupakan implementasi algoritma pengurutan Bubble Sort dengan urutan descending pada array bilangan bulat. Program akan menghasilkan 20 bilangan bulat acak dengan rentang nilai 1 hingga 100 dan akan mengurutkannya secara descending menggunakan algoritma Bubble Sort. Program juga mencatat waktu yang dibutuhkan untuk melakukan pengurutan dengan menggunakan fungsi time() pada library time.h.

  Dalam algoritma Bubble Sort, setiap elemen dari array akan dibandingkan dengan elemen yang berada di sebelahnya. Jika nilai elemen tersebut lebih besar dari elemen yang berada di sebelahnya, maka kedua elemen tersebut akan ditukar posisinya. Proses ini akan terus dilakukan sampai tidak ada lagi pertukaran yang dilakukan pada iterasi tersebut. Setelah itu, proses akan dilakukan pada elemen yang tersisa dan akan terus berulang sampai seluruh elemen di dalam array terurut dengan benar.

2. Dari percobaan 2 tambahkan fungsi untuk melakukan pengurutan bubble sort dengan flag secara descending.

   Jawab :

   - **Program**

```c
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#include<stdbool.h>

#define MAX 20
void BubbleSortFlag(int arr[]) {
    int i = 0, j, temp;
    bool did_swap = true;
    while (i < MAX - 1 && did_swap) {
        for (j = 0; j < MAX - i - 1; j++) {
            did_swap = false;
            if (arr[j] < arr[j + 1]) {
                temp = arr[j + 1];
                arr[j + 1] = arr[j];
```

```
                arr[j] = temp;
                did_swap = true;
            }
        }
        i++;
    }
}
void main() {
    int data_awal[MAX], data_urut[MAX];
    int i;
    time_t k1, k2;
    printf("Sebelum pengurutan : \n");
    for (i = 0; i < MAX; i++) {
        srand(time(NULL) * (i + 1));
        data_awal[i] = rand() % 100 + 1;
        printf("%d ", data_awal[i]);
    }
    printf("\nSetelah pengurutan : \n");
    for (i = 0; i < MAX; i++)
        data_urut[i] = data_awal[i];
    time( & k1);
    BubbleSortFlag(data_urut);
    time( & k2);
    for (i = 0; i < MAX; i++)
        printf("%d ", data_urut[i]);
    printf("\nWaktu = %ld\n", k2 - k1);
}
```

- **Output**

```
Sebelum pengurutan :
55 3 51 99 47 63 11 59 7 55 3 19 67 15 63 11 59 75 23 71
Setelah pengurutan :
99 75 71 67 63 63 59 59 55 55 51 47 23 19 15 11 11 7 3 3
Waktu = 0
```

- **Analisa**

  Program di atas merupakan implementasi Bubble Sort dengan menggunakan flag untuk mempercepat proses sorting pada array dengan ukuran 20 elemen.

  Pertama-tama, program akan mengisi array data_awal dengan 20 nilai acak menggunakan fungsi rand(). Kemudian, nilai-nilai ini akan dicetak di layar sebelum sorting dilakukan.

  Setelah itu, array data_urut akan diisi dengan nilai-nilai dari data_awal dan akan diurutkan menggunakan fungsi BubbleSortFlag(). Proses sorting menggunakan Bubble Sort dan

dilakukan dengan memanipulasi flag did_swap yang menandakan apakah ada perubahan yang dilakukan pada iterasi sebelumnya. Jika tidak ada perubahan yang dilakukan, maka proses sorting dihentikan.

Setelah array terurut, nilai-nilai array tersebut akan dicetak di layar, beserta dengan waktu yang dibutuhkan untuk proses sorting

3. Dari percobaan 3 tambahkan fungsi untuk melakukan pengurutan shell sort secara descending.
   Jawab :
   - **Program**

```
#include<stdio.h>
#include<stdbool.h>
#include<stdlib.h>
#include<time.h>

#define MAX 20
void ShellSort(int arr[]) {
    int i, jarak, temp;
    bool did_swap = true;
    jarak = MAX;
    while (jarak > 1) {
        jarak = jarak / 2;
        did_swap = true;
        while (did_swap) {
            did_swap = false;
            i = 0;
            while (i < (MAX - jarak)) {
                if (arr[i] < arr[i + jarak]) {
                    temp = arr[i];
                    arr[i] = arr[i + jarak];
                    arr[i + jarak] = temp;
                    did_swap = true;

                }
                i++;
            }
        }
    }
}
void main() {
    int data_awal[MAX], data_urut[MAX];
    int i;
    long k1, k2;
    printf("Sebelum pengurutan : \n");
```

```
    for (i = 0; i < MAX; i++) {
        srand(time(NULL) * (i + 1));
        data_awal[i] = rand() % 100 + 1;
        printf("%d ", data_awal[i]);
    }
    printf("\nSetelah pengurutan : \n");
    for (i = 0; i < MAX; i++)
        data_urut[i] = data_awal[i];
    time( & k1);
    ShellSort(data_urut);
    time( & k2);
    for (i = 0; i < MAX; i++)
        printf("%d ", data_urut[i]);
    printf("\nWaktu = %ld\n", k2 - k1);
}
```

- **Output**

```
Sebelum pengurutan :
47 86 25 65 4 44 51 91 30 70 9 48 56 95 35 74 14 53 61 100
Setelah pengurutan :
100 95 91 86 74 70 65 61 56 53 51 48 44 35 30 25 14 9 4 0
Waktu = 1683643650
```

- **Analisa**

  Program di atas merupakan implementasi dari algoritma Shell Sort untuk mengurutkan array dengan 20 elemen secara menurun. Shell Sort merupakan pengembangan dari Insertion Sort yang dimana insertion sort mengurutkan elemen yang berdekatan sedangkan Shell Sort mengurutkan elemen dengan jarak tertentu. Pada program di atas, jarak antar elemen yang dibandingkan pada awalnya diatur sebesar MAX, kemudian jarak tersebut dibagi 2 pada setiap iterasi hingga jarak antar elemen yang dibandingkan menjadi 1. Selama proses pengurutan, program menggunakan variabel did_swap untuk mengecek apakah terdapat perubahan elemen pada iterasi tersebut. Proses pengurutan dilakukan dengan menukar elemen jika elemen di indeks i lebih kecil dari elemen di indeks i + jarak. Program diakhiri dengan mencetak array yang sudah terurut dan waktu yang diperlukan untuk melakukan pengurutan.

4. Buatlah struktur Mahasiswa dengan variable nrp dan nama yang memiliki tipe String dan kelas yang bertipe int. Buatlah fungsi pengurutan dengan bubble sort, bubble sort dengan flag dan shell sort berdasarkan nrp.

```
struct Mahasiswa {
    char nrp[10];
    char nama[20];
    int kelas;
};
```

Jawab :

- **Program**

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

struct Mahasiswa {
    char nrp[10];
    char nama[50];
    int kelas;
};

void bubbleSort(struct Mahasiswa arr[], int n) {
    int i, j;
    struct Mahasiswa temp;
    for (i = 0; i < n-1; i++) {
        for (j = 0; j < n-i-1; j++) {
            if (strcmp(arr[j].nrp, arr[j+1].nrp) > 0) {
                temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }
        }
    }
}

void bubbleSortFlag(struct Mahasiswa arr[], int n) {
    int i, j, flag = 1;
    struct Mahasiswa temp;
    for (i = 0; i < n-1 && flag; i++) {
        flag = 0;
        for (j = 0; j < n-i-1; j++) {
            if (strcmp(arr[j].nrp, arr[j+1].nrp) > 0) {
                temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
                flag = 1;
            }
        }
    }
}
```

```c
void shellSort(struct Mahasiswa arr[], int n) {
    int gap, i, j;
    struct Mahasiswa temp;
    for (gap = n/2; gap > 0; gap /= 2) {
        for (i = gap; i < n; i++) {
            temp = arr[i];
            for (j = i; j >= gap && strcmp(arr[j-gap].nrp,
temp.nrp) > 0; j -= gap) {
                arr[j] = arr[j-gap];
            }
            arr[j] = temp;
        }
    }
}

int main() {
    struct Mahasiswa arr[5];
    int i, n;
    printf("Masukkan jumlah data: ");
    scanf("%d", &n);
    for (i = 0; i < n; i++) {
        printf("Masukkan data ke-%d\n", i+1);
        printf("NRP: ");
        scanf("%s", arr[i].nrp);
        printf("Nama: ");
        scanf("%s", arr[i].nama);
        printf("Kelas: ");
        scanf("%d", &arr[i].kelas);
    }


    // Pengurutan dengan bubble sort
    bubbleSort(arr, n);
    printf("Setelah pengurutan:\n\n");
    printf("Bubble Sort:\n");

    for (i = 0; i < n; i++) {
        printf("%s %s %d\n", arr[i].nrp, arr[i].nama,
arr[i].kelas);
    }
    // Pengurutan dengan bubble sort dengan flag
    printf("\nBubble Sort dengan flag:\n");
    bubbleSortFlag(arr, n);
    for (i = 0; i < n; i++) {
```

```
        printf("%s %s %d\n", arr[i].nrp, arr[i].nama,
arr[i].kelas);
    }
    // Pengurutan dengan shell sort
    printf("\nShell Sort:\n");
    shellSort(arr, n);
    for (i = 0; i < n; i++) {
        printf("%s %s %d\n", arr[i].nrp, arr[i].nama,
arr[i].kelas);
    }
    return 0;
}
```

- **Output**

```
Masukkan jumlah data: 3
Masukkan data ke-1
NRP: 1
Nama: hafis
Kelas: 2
Masukkan data ke-2
NRP: 32
Nama: sadf
Nama: hafis
Kelas: 2
Masukkan data ke-2
NRP: 43
Nama: anan
Kelas: 3
Setelah pengurutan:

Bubble Sort:
23 hafis 2
43 anan 3

Bubble Sort dengan flag:
23 hafis 2
43 anan 3

Shell Sort:
23 hafis 2
43 anan 3
```

- **Analisa**

  Program di atas merupakan program untuk mengurutkan data mahasiswa berdasarkan NRP, nama, dan kelas menggunakan algoritma Bubble Sort, Bubble Sort dengan Flag, dan Shell Sort. Program ini menggunakan struktur data Mahasiswa yang terdiri dari atribut NRP, nama, dan kelas.

Pada program ini, terdapat tiga fungsi pengurutan, yaitu bubbleSort, bubbleSortFlag, dan shellSort, masing-masing menggunakan algoritma Bubble Sort, Bubble Sort dengan Flag, dan Shell Sort. Bubble Sort mengurutkan elemen-elemen array dengan membandingkan elemen-elemen tersebut secara berpasangan dan menukar posisi apabila elemen berikutnya lebih kecil daripada elemen sebelumnya. Bubble Sort dengan Flag memiliki cara kerja yang sama dengan Bubble Sort, namun ditambahkan dengan flag yang menandakan apakah ada swap atau tidak pada suatu iterasi. Apabila tidak ada swap, maka iterasi akan dihentikan. Sedangkan, Shell Sort mengurutkan elemen-elemen array dengan membandingkan elemen yang berjarak tertentu kemudian menukar posisi.