PRAKTIKUM STRUKTUR DATA
Tugas Jobsheet 10

Dosen Pengampu :
Randi Proska Sandra, M.Sc

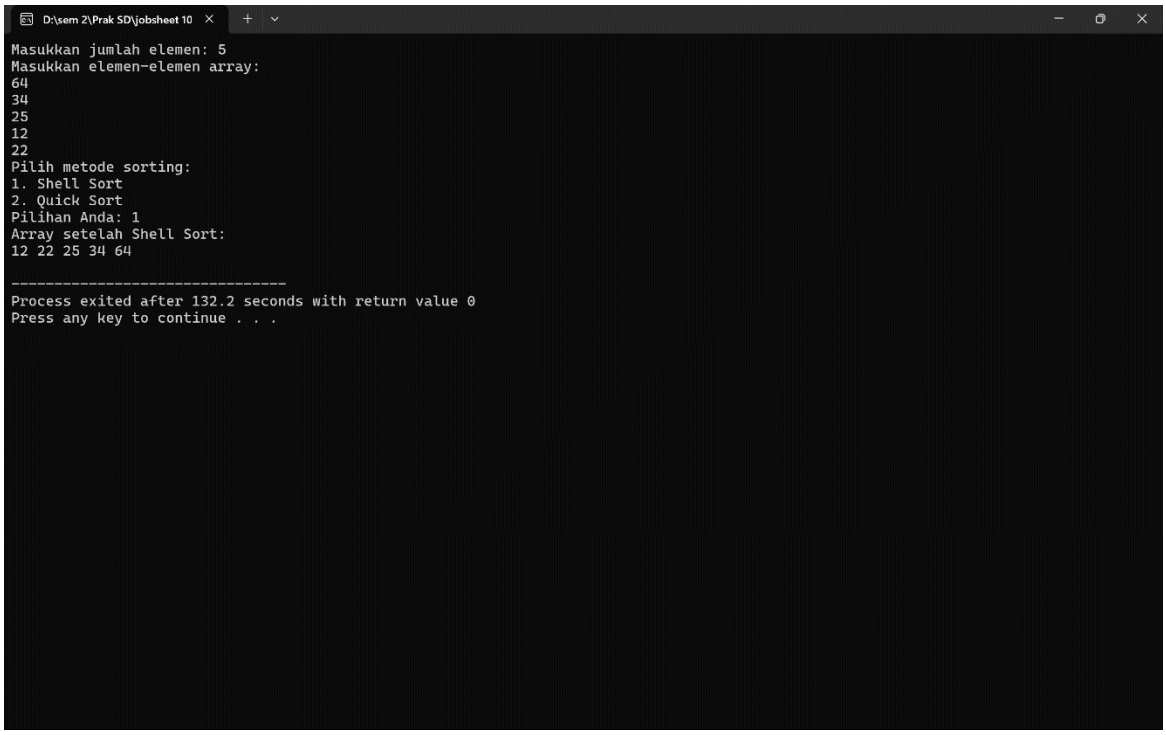
Disusun Oleh :

Joel Wiseda Simanungkalit
23343071

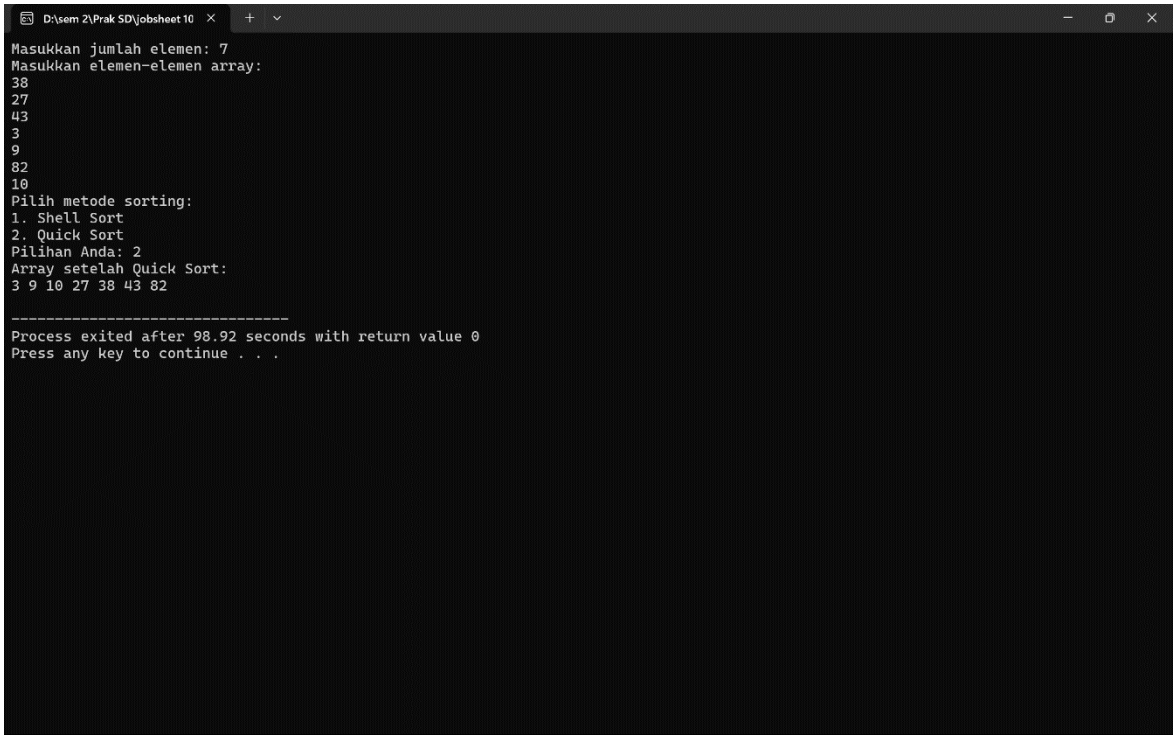
PROGRAM STUDI INFORMATIKA (NK)
FAKULTAS TEKNIK
UNIVERSITAS NEGERI PADANG
2024

A. Output

1. Shell Sort



2. Quick Sort



B. Penjelasan Prinsip Kerja Shell Sort dan Quick Sort

Shell Sort

Shell Sort adalah variasi dari Insertion Sort yang memungkinkan pertukaran elemen-elemen yang jauh. Prinsip kerja Shell Sort adalah sebagai berikut:

1. **Pembagian Gap**: Algoritma ini mulai dengan memilih gap yang besar dan kemudian menguranginya secara bertahap. Gap adalah jarak antara elemen yang dibandingkan.
2. **Pengurutan Elemen dengan Gap**: Untuk setiap gap, algoritma membandingkan elemen-elemen yang dipisahkan oleh gap dan melakukan pertukaran jika elemen-elemen tersebut tidak dalam urutan yang benar.
3. **Pengurangan Gap**: Proses ini diulangi dengan gap yang lebih kecil hingga gap menjadi 1, di mana Shell Sort berakhir sebagai Insertion Sort.

Pada program di atas, fungsi `shellSort` mengimplementasikan prinsip ini dengan menggunakan loop `for` yang mengurangi gap dari `n/2` hingga 1, dan loop nested yang melakukan perbandingan dan pertukaran elemen.

Kode Shell Sort:

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

Quick Sort

Quick Sort adalah algoritma sorting yang efisien dengan kompleksitas waktu rata-rata O(n log n). Prinsip kerja Quick Sort adalah sebagai berikut:

1. **Pilih Pivot**: Algoritma memilih elemen sebagai pivot. Elemen diatur sedemikian rupa sehingga elemen yang lebih kecil dari pivot berada di kiri dan elemen yang lebih besar berada di kanan.
2. **Partisi Array**: Array dipecah menjadi dua bagian: elemen yang lebih kecil dari pivot dan elemen yang lebih besar dari pivot.
3. **Panggilan Rekursif**: Algoritma memanggil dirinya sendiri secara rekursif untuk bagian kiri dan kanan hingga seluruh array terurut.

Pada program di atas, fungsi `quickSort` mengimplementasikan prinsip ini dengan menggunakan fungsi `partition` untuk membagi array berdasarkan pivot, dan memanggil dirinya sendiri untuk setiap bagian.

Kode Quick Sort:

```
void quickSort(int arr[], int low, int high) {
    if (low < high) {
        int pi = partition(arr, low, high);

        quickSort(arr, low, pi - 1);
        quickSort(arr, pi + 1, high);
    }
}

int partition(int arr[], int low, int high) {
    int pivot = arr[high];
    int i = (low - 1);

    for (int j = low; j <= high - 1; j++) {
        if (arr[j] < pivot) {
            i++;
            swap(&arr[i], &arr[j]);
        }
    }
    swap(&arr[i + 1], &arr[high]);
    return (i + 1);
}

void swap(int* a, int* b) {
    int t = *a;
    *a = *b;
    *b = t;
}
```

Kesimpulan

Shell Sort dan Quick Sort adalah dua algoritma sorting dengan pendekatan yang berbeda. Shell Sort adalah perbaikan dari Insertion Sort yang memperkenalkan konsep gap untuk mempercepat pengurutan elemen yang jauh, sementara Quick Sort adalah algoritma berbasis partisi yang efisien dengan kompleksitas waktu rata-rata O(n log n). Quick Sort bekerja dengan memilih pivot, membagi array, dan mengurutkan bagian kiri dan kanan secara rekursif. Kedua algoritma ini mengurutkan array dengan cara yang berbeda tetapi efektif dalam berbagai kasus.