Nama : M.Gilang Mulya Putra

NIM : 23343073

Prrodi : Informatika

**Penjelasan**

**Shell Sort**

1. **Prinsip Kerja**:
   * Shell Sort adalah perpanjangan dari Insertion Sort yang mengurangi jumlah pergeseran dengan membandingkan elemen yang terpisah jauh (gap) dan mengurutkannya, kemudian mengurangi gap dan mengulang proses hingga gap menjadi 1.
2. **Langkah-langkah**:
   * Tentukan gap awal, biasanya setengah dari panjang array.
   * Lakukan insertion sort pada elemen-elemen yang terpisah oleh gap.
   * Kurangi gap setengahnya dan ulangi proses hingga gap menjadi 1.
3. **Kelebihan dan Kekurangan**:
   * Lebih cepat dari Insertion Sort dan Bubble Sort untuk daftar besar karena pengurangan jumlah pergeseran.
   * Kompleksitas waktu terbaik O(n log n), tetapi tergantung pada pilihan gap sequence.

**Quick Sort**

1. **Prinsip Kerja**:
   * Quick Sort adalah algoritma berbasis divide-and-conquer yang bekerja dengan cara memilih elemen pivot dan mempartisi array menjadi dua subarray di mana elemen-elemen yang lebih kecil dari pivot berada di sebelah kiri dan yang lebih besar di sebelah kanan, kemudian mengurutkan subarray tersebut secara rekursif.
2. **Langkah-langkah**:
   * Pilih elemen pivot, biasanya elemen terakhir.
   * Partisi array sehingga elemen-elemen yang lebih kecil dari pivot berada di kiri dan yang lebih besar di kanan.
   * Urutkan subarray kiri dan kanan secara rekursif.
3. **Kelebihan dan Kekurangan**:
   * Sangat cepat untuk daftar besar dengan kompleksitas waktu rata-rata O(n log n).
   * Kompleksitas waktu terburuk O(n^2) tetapi jarang terjadi jika pivot dipilih dengan baik (misalnya dengan median-of-three atau randomization).

**Kesimpulan**

Implementasi Shell Sort dan Quick Sort di atas menunjukkan bagaimana kedua algoritma bekerja dalam bahasa C. Shell Sort lebih cepat dari Insertion Sort untuk daftar besar dengan pengurangan jumlah pergeseran menggunakan gap, sementara Quick Sort sangat efisien untuk daftar besar dengan pembagian dan penaklukan yang cerdas, meskipun memiliki potensi kompleksitas waktu terburuk yang lebih tinggi.

//Created by M.Gilang Mulya Putra\_23343073

#include <stdio.h>

// Fungsi untuk 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;

}

}

}

// Fungsi untuk pertukaran elemen

void swap(int\* a, int\* b) {

int t = \*a;

\*a = \*b;

\*b = t;

}

// Fungsi untuk partisi dalam Quick Sort

int partition(int arr[], int low, int high) {

int pivot = arr[high];

int i = (low - 1);

for (int j = low; j < high; j++) {

if (arr[j] < pivot) {

i++;

swap(&arr[i], &arr[j]);

}

}

swap(&arr[i + 1], &arr[high]);

return (i + 1);

}

// Fungsi untuk 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);

}

}

// Fungsi untuk mencetak array

void printArray(int arr[], int size) {

for (int i = 0; i < size; i++) {

printf("%d ", arr[i]);

}

printf("\n");

}

int main() {

int arr1[] = {12, 34, 54, 2, 3};

int arr2[] = {12, 34, 54, 2, 3};

int n1 = sizeof(arr1)/sizeof(arr1[0]);

int n2 = sizeof(arr2)/sizeof(arr2[0]);

printf("Array sebelum sorting:\n");

printArray(arr1, n1);

// Menggunakan Shell Sort

shellSort(arr1, n1);

printf("\nArray setelah Shell Sort:\n");

printArray(arr1, n1);

// Menggunakan Quick Sort

quickSort(arr2, 0, n2 - 1);

printf("\nArray setelah Quick Sort:\n");

printArray(arr2, n2);

return 0;

}

