NAMA : Dipa Dyah Puspita Arum

KELAS/NIM:03TPLP027 / 231011401098

Kerjakan soal dibawah ini dengan Uraian yang jelas!

1. Anda diberikan sebuah array yang berisi data acak. Anda diminta untuk mengimplementasikan algoritma sorting yang paling efisien untuk data tersebut.

Pertanyaan:

a) Bagaimana cara Anda menentukan algoritma sorting yang paling tepat? (10 Poin)
Jawaban: Untuk menentukan algoritma sorting yang paling tepat, perlu mempertimbangkan:
1. Ukuran Data:
Untuk dataset kecil (), algoritma sederhana seperti Insertion Sort lebih efisien karena overhead rendah.
Untuk dataset besar (), algoritma seperti Merge Sort, Quick Sort, atau Heap Sort lebih efisien.
2. Tingkat Keterurutan Awal:
Jika data hampir terurut, Insertion Sort sangat cocok karena kompleksitasnya mendekati .
Jika data tidak terurut sama sekali, Quick Sort atau Merge Sort lebih ideal.
3. Stabilitas Algoritma:
Jika stabilitas penting (misalnya, elemen dengan nilai sama harus tetap dalam urutan aslinya), gunakan algoritma stabil seperti Merge Sort atau Bubble Sort.
4. Sumber Daya yang Tersedia:
jika memori terbatas, Heap Sort lebih cocok karena bekerja in-place.
Jika memori bukan masalah, Merge Sort dapat dipertimbangkan.

b) Jelaskan faktor-faktor yang perlu dipertimbangkan (misalnya, ukuran data, tingkat
keterurutan awal, stabilitas algoritma). (10 Poin)
Jawaban: Faktor-faktor yang Perlu Dipertimbangkan
1. Kompleksitas Waktu:
Quick Sort: untuk rata-rata kasus, tetapi untuk kasus terburuk.

Merge Sort: Selalu.
Bubble Sort: , lambat untuk data besar.
2. Kompleksitas Ruang:
Quick Sort: In-place (untuk stack rekursi).
Merge Sort: Membutuhkan ruang tambahan .
Heap Sort: In-place, tetapi tidak stabil.
3. Stabilitas:
Stabil: Merge Sort, Insertion Sort.
Tidak stabil: Quick Sort, Heap Sort.
4. Tingkat Keterurutan Awal:
Data hampir terurut: Insertion Sort atau Bubble Sort.
Data acak: Quick Sort atau Merge Sort.

C) Bandingkan dan kontraskan antara algoritma binary search dan linear search. Dalam situasi apa binary search lebih disukai dibandingkan linear search? Jelaskan dengan contoh program dalam C++ atau Python. (30 Poin)

Jawaban: Perbandingan Binary Search dan Linear Search
1. Binary Search:
Kelebihan: Cepat () untuk data besar jika array sudah terurut.
Kekurangan: Membutuhkan data terurut.
2. Linear Search:
Kelebihan: Dapat digunakan untuk array yang tidak terurut.
Kekurangan: Lambat () untuk dataset besar.
Situasi Penggunaan:
Binary Search digunakan saat data terurut (contoh: mencari nama dalam daftar yang disusun secara alfabetis).
Linear Search digunakan jika data tidak terurut atau ukuran dataset kecil.
Coding: linear search

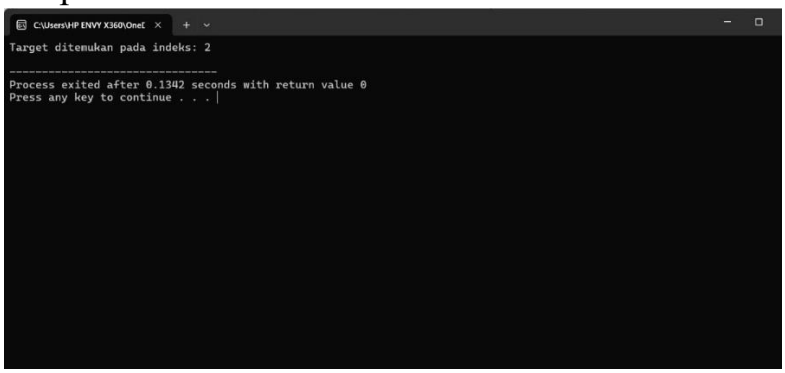
UAS ALGO_LINEAR SEARCH.cpp

```cpp
#include <iostream>
using namespace std;

int binarySearch(int arr[], int n, int target) {
    int left = 0, right = n - 1;
    while (left <= right) {
        int mid = (left + right) / 2;
        if (arr[mid] == target) {
            return mid;
        } else if (arr[mid] < target) {
            left = mid + 1;
        } else {
            right = mid - 1;
        }
    }
    return -1;
}

int main() {
    int arr[] = {1, 2, 4, 7, 9};  // Data harus terurut
    int n = 5, target = 7;
    int result = binarySearch(arr, n, target);
    if (result != -1) {
        cout << "Target ditemukan pada indeks: " << result << endl;
    } else {
cout << "Target tidak ditemukan." << endl;
}
    return 0;
}
```

Output:



2. Anda diberikan dua buah array yang sudah terurut.
Pertanyaan dan Tugas:
a) Bagaimana cara Anda menggabungkan kedua array tersebut menjadi satu array yang baru dan tetap terurut dalam waktu yang efisien? (10 Poin)
Jawaban: Untuk menggabungkan dua array terurut menjadi satu array yang tetap terurut, kita dapat menggunakan algoritma merge. Langkah-langkahnya adalah:
1. Gunakan dua pointer, masing-masing menunjuk ke elemen pertama dari kedua array.
2. Bandingkan elemen yang ditunjuk kedua pointer.
3. Masukkan elemen yang lebih kecil ke array baru, lalu geser pointer array tersebut.
4. Ulangi langkah ini hingga salah satu array habis.

5. Tambahkan elemen yang tersisa dari array yang belum
habis ke array baru.

b) Algoritma penggabungan mana yang akan Anda pilih dan mengapa? (10 Poin)

Jawaban: Algoritma Penggabungan yang Dipilih

Algoritma Merge dipilih karena:

1. Kompleksitas Waktu: , di mana adalah panjang array pertama dan adalah panjang array kedua. Ini lebih efisien dibandingkan algoritma sorting lainnya karena hanya memanfaatkan fakta bahwa kedua array sudah terurut.
2. Efisiensi: Menghindari proses sorting ulang dari awal.
3. Kesederhanaan Implementasi: Algoritma ini mudah dipahami dan diimplementasikan.

c) Buatkan Program dalam C++ atau Python untuk algoritma sorting yang telah anda pilih! (30 Poin)

Jawaban:

Coding**:** binary search

uas binary search.cpp

```cpp
#include <iostream>
#include <vector>
using namespace std;

vector<int> mergeSortedArrays(vector<int> arr1, vector<int> arr2) {
    int i = 0, j = 0;
    vector<int> merged;

    // Bandingkan elemen dan tambahkan ke array hasil
    while (i < arr1.size() && j < arr2.size()) {
        if (arr1[i] < arr2[j]) {
            merged.push_back(arr1[i]);
            i++;
        } else {
            merged.push_back(arr2[j]);
            j++;
        }
    }

    // Tambahkan elemen yang tersisa
    while (i < arr1.size()) {
        merged.push_back(arr1[i]);
        i++;
    }
    while (j < arr2.size()) {
        merged.push_back(arr2[j]);
        j++;
    }

    return merged;
}

int main() {
    vector<int> array1 = {1, 3, 5, 7};
    vector<int> array2 = {2, 4, 6, 8};

    vector<int> result = mergeSortedArrays(array1, array2);

    cout << "Array yang digabung: ";
    for (int num : result) {
        cout << num << " ";
    }
    cout << endl;

    return 0;
}
```

Output: