## **Nama : Ahmad Wahyu Abadi S.**

## **Kelas / Absen : XI RPl / 02**

## **D. Latihan dan Studi Kasus**

#### **Soal Teori:**

#### **Soal Teori:**

1. **Mengapa O(log n) lebih baik daripada O(n) untuk set data yang sangat besar? Jelaskan dengan analogi kamus.** Karena O(log n) hanya perlu membagi masalah setiap langkahnya. Seperti mencari kata di kamus dengan membuka tengah dan mengecilkan ruang pencarian setiap saat, jauh lebih cepat dibanding mencari satu per satu.
2. **Jika Anda memiliki dua algoritma untuk menyelesaikan masalah yang sama, satu dengan kompleksitas O(n) dan satu lagi O(n²), mana yang akan Anda pilih untuk aplikasi yang akan digunakan oleh jutaan pengguna? Mengapa?** Pilih yang O(n), karena pertumbuhannya lebih lambat dan tetap efisien meskipun jumlah data sangat besar. O(n²) akan sangat lambat saat data bertambah banyak.

### **Soal Praktik**

**Soal:**Studi Kasus Praktik: Perhatikan dua potongan kode di bawah ini untuk mencari apakah ada nilai duplikat dalam sebuah list:

Kode A:

| def cek\_duplikat\_A(data\_list):  for i in range(len(data\_list)):  for j in range(i + 1, len(data\_list)):  if data\_list[i] == data\_list[j]:  return True  return False |
| --- |

Kode B:

| def cek\_duplikat\_B(data\_list):  dilihat = set()  for item in data\_list:  if item in dilihat:  return True  dilihat.add(item)  return False |
| --- |

Analisislah Big O Notation untuk Kode A dan Kode B. Jelaskan mana yang lebih efisien dan mengapa! (Petunjuk: Operasi add dan in pada set di Python rata-rata adalah O(1)).

**Jawab :**

### **✅ Kode A (O(n²)) – Pendekatan Sederhana dengan Dua Loop**

def cek\_duplikat\_A(data\_list):  
 for i in range(len(data\_list)):  
 for j in range(i + 1, len(data\_list)):  
 if data\_list[i] == data\_list[j]:  
 return True  
 return False

#### **🔍 Analisis Big O:**

* Dua loop bersarang yang memeriksa semua pasangan elemen satu per satu.
* Pada setiap elemen i, ia membandingkan dengan semua elemen setelahnya j.
* Maka jumlah total perbandingan ≈ n(n-1)/2 → **O(n²)**

#### **📌 Kelebihan:**

* Mudah dipahami dan tidak perlu struktur data tambahan.

#### **⚠️ Kekurangan:**

* Sangat tidak efisien untuk data besar. Waktu eksekusi membesar dengan cepat.

### **✅ Kode B (O(n)) – Pendekatan Menggunakan Set**

def cek\_duplikat\_B(data\_list):  
 dilihat = set()  
 for item in data\_list:  
 if item in dilihat:  
 return True  
 dilihat.add(item)  
 return False

#### **🔍 Analisis Big O:**

* Hanya satu loop (for), sehingga **O(n)**.
* Operasi item in dilihat dan dilihat.add(item) memiliki kompleksitas rata-rata **O(1)** karena menggunakan struktur data **set**.
* Maka total waktu adalah **O(n)** secara keseluruhan.

#### **📌 Kelebihan:**

* Jauh lebih efisien, terutama untuk jumlah data besar.
* Tidak membandingkan elemen satu per satu secara manual.

#### **⚠️ Kekurangan:**

* Menggunakan memori tambahan (set) untuk menyimpan elemen yang sudah dilihat.

### **🔚 Kesimpulan Studi Kasus:**

| **Aspek** | **Kode A** | **Kode B** |
| --- | --- | --- |
| Kompleksitas | O(n²) – sangat lambat | O(n) – sangat efisien |
| Pendekatan | Loop bersarang manual | Menggunakan set() |
| Efisiensi Data Besar | ❌ Tidak disarankan | ✅ Sangat direkomendasikan |

**🟢 Pilihan terbaik:**

**Kode B**, karena memiliki kompleksitas **O(n)** yang jauh lebih efisien dibanding **O(n²)** pada Kode A, terutama jika program akan digunakan pada data dalam jumlah besar.