

# LAPORAN TUGAS ULANGAN AKHIR SEMESTER INDIVIDU

---

NAMA : NAHDLIYAH ZAHRAH  
KELAS : SAINS DATA 2022A  
NIM : 22031554024  
MATKUL : STRUKTUR DATA dan ALGORITMA

TUGAS : Merubah Algoritma sorting dari Bubble sort ke Linear sort

JAWAB :

## Pengurutan Linear

Algoritma pengurutan linear adalah algoritma yang tidak menggunakan tanda perbandingan ( $>$ ,  $<$ ,  $>=$ ,  $<=$ ,  $==$ ) untuk menentukan urutan elemen dalam data (proses pengurutan tidak memakai perbandingan). Pengurutan dilakukan dengan cara logika dasar dan waktu eksekusi yang digunakan oleh algoritma bersifat linear.

Linear sort membutuhkan lebih banyak space memori dalam melakukan operasinya. Ini dikarenakan metode linear sort yang bersifat linear selalu membutuhkan memori untuk operasi. Sedangkan tingkat kecepatan eksekusi dapat dikatakan lumayan cepat jika dibandingkan dengan operasi / metode pengurutan menggunakan perbandingan.

Algoritma pengurutan yang sesuai dengan pernyataan tersebut adalah Counting Sort dan Radix Sort. Kedua algoritma ini adalah contoh algoritma pengurutan linier yang tidak menggunakan operasi perbandingan langsung antara elemen-elemen yang akan diurutkan. Dalam hal ini saya akan menggunakan Counting sort karena Radix sort membutuhkan struktur data yang lebih kompleks dan tidak terbatas pada perbandingan harga produk.

Counting sort merupakan sebuah teknik pengurutan dengan cara menghitung jumlah kemunculan dari setiap data yang berada di dalam array.

Perubahan :

Bubble sort

```
def bubble_sort(self, reverse):
    n = len(self.products)
    for i in range(n - 1):
        for j in range(0, n - i - 1):
            if reverse:
                if self.products[j].price < self.products[j + 1].price:
```

```

        self.products[j], self.products[j + 1] = self.products[j
+ 1], self.products[j]
    else:
        if self.products[j].price > self.products[j + 1].price:
            self.products[j], self.products[j + 1] = self.products[j
+ 1], self.products[j]

```

Counting sort :

```

def counting_sort(self, reverse):
    # Step 1: Menghitung nilai maksimum dan minimum dari harga produk
    max_price = max(self.products, key=lambda p: p.price).price
    min_price = min(self.products, key=lambda p: p.price).price

    # Step 2: Menghitung rentang harga
    price_range = max_price - min_price + 1

    # Step 3: Membuat array count untuk menghitung frekuensi kemunculan
    # setiap harga produk
    count = [0] * price_range

    for product in self.products:
        count[product.price - min_price] += 1

    # Step 4: Jika reverse=True, pengurutan dilakukan dari harga tertinggi ke
    # terendah
    if reverse:
        sorted_products = []
        for i in range(price_range - 1, -1, -1):
            while count[i] > 0:
                # Step 5a: Mencari harga produk yang sesuai dengan i +
                min_price

                price = i + min_price
                for product in self.products:
                    if product.price == price:
                        # Step 5b: Menambahkan produk ke dalam
                        sorted_products dan mengurangi count[i] sebanyak 1
                        sorted_products.append(product)
                        count[i] -= 1
                        break # Melanjutkan ke iterasi selanjutnya setelah
                        menemukan produk yang sesuai

    # Step 6: Jika reverse=False, pengurutan dilakukan dari harga terendah ke
    # tertinggi

```

```

else:
    sorted_products = []
    for i in range(price_range):
        while count[i] > 0:
            # Step 7a: Mencari harga produk yang sesuai dengan i +
min_price

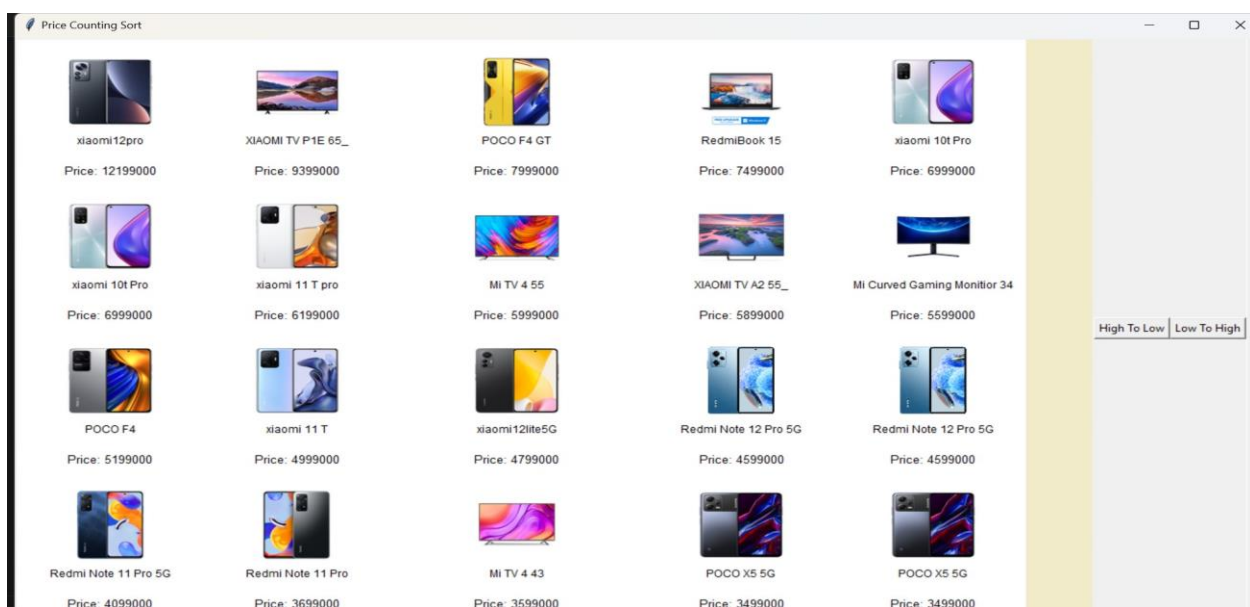
            price = i + min_price
            for product in self.products:
                if product.price == price:
                    # Step 7b: Menambahkan produk ke dalam
sorted_products dan mengurangi count[i] sebanyak 1
                    sorted_products.append(product)
                    count[i] -= 1
                    break # Melanjutkan ke iterasi selanjutnya setelah
menemukan produk yang sesuai

            # Step 8: Mengganti self.products dengan sorted_products setelah
pengurutan selesai
            self.products = sorted_products





















```

- Kompleksitas waktu Counting Sort adalah  $O(n + k)$ , di mana  $n$  adalah jumlah elemen dalam himpunan data yang akan diurutkan, dan  $k$  adalah rentang nilai dari elemen-elemen tersebut.
- Kompleksitas ruang (space complexity) Counting Sort adalah  $O(n + k)$ , di mana  $n$  adalah jumlah elemen dalam himpunan data yang akan diurutkan, dan  $k$  adalah rentang nilai dari elemen-elemen tersebut.

Hasil counting sort (Descending sort)



## Hasil counting sort (Ascending sort)

 Mi Watch Charging Dock Price: 69000	 Xiaomi 6A Type-A to Type-C Cable Price: 79000	 Mi Wireless Switch Price: 99000	 Mi 37W Dual-Port Car Charger Price: 109000	 Mi Router 4C Price: 199000
 Mi WiFi Range Extender AC1200 Price: 249000	 Xiaomi 67W Charging Combo (Type-A) EU Price: 299000	 Mi 360 Camera (1080p) Price: 449000	 Redmi buds 4 Price: 549000	 Redmi buds 4 Price: 549000
 Mi TV Stick Price: 599000	 Mi TV Stick Price: 599000	 Mi TV Stick Price: 599000	 Mi Vacuum Cleaner Mini.jpg Price: 649000	 Redmi Buds 3 Pro Price: 699000
 Redmi Buds 3 Pro Price: 699000	 Redmi Buds 3 Pro Price: 699000	 Redmi buds 4 Pro Price: 949000	 Mi Smart Air Fryer (3.5L) Price: 1199000	 Mi Smart Air Fryer (3.5L) Price: 1199000

Kesimpulannya, dengan algoritma yang berbeda hasil tetap sama. Jika kita ingin mengurutkan elemen dengan rentang nilai terbatas dan ingin mencapai efisiensi waktu, Counting Sort adalah pilihan yang lebih baik. Namun, jika kita memiliki jumlah elemen yang relatif kecil atau mengurutkan data yang hampir terurut, Bubble Sort bisa menjadi pilihan yang lebih sederhana.