Penerapan Algoritma BFS dan DFS dalam Penentuan Toko *Online* Asli atau *Dropship* di Shopee

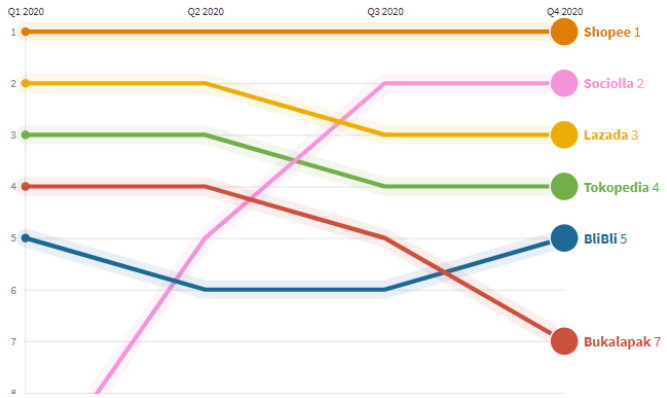
Jesica - 13519011
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13519011@std.stei.itb.ac.id

***Abstract*—Di era yang serba canggih seperti sekarang ini, hampir segala sesuatu dilakukan secara *online*. Kegiatan jual beli yang dulunya ramai memenuhi pasar, kini juga sudah berubah menjadi pasar *online*. Apalagi semenjak adanya pandemi Covid-19 yang membatasi interaksi sosial antar manusia, rasanya transaksi *online* ini menjadi alternatif paling ideal karena tidak dibutuhkan interaksi secara langsung dalam membeli maupun menjual barang. Biasanya kegiatan jual beli *online* ini diperantarai oleh pihak ketiga, yaitu sebuah *marketplace* tempat bagi para penjual membuka lapaknya dan juga tempat bagi para pembeli untuk membeli keperluannya. Di Indonesia sendiri ada berbagai macam *marketplace*, contohnya adalah Tokopedia dan Shopee atau sering disebut juga lapak hijau dan lapak oranye yang merupakan *marketplace* terbesar di negara ini.**

Keywords—online shop, BFS, DFS, Shopee, lapak, Dropship

I. Pendahuluan

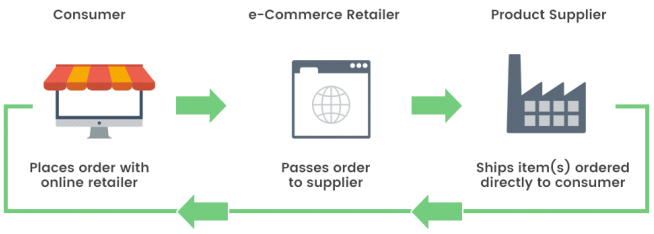
Shopee merupakan salah satu *marketplace* terbesar di Indonesia jika ditinjau dari jumlah masyarakat yang menginstal dan menggunakannya. Dari data yang dirangkum oleh iPrice, terlihat bahwa Shopee memiliki jumlah pengunjung sekaligus peringkat tertinggi ditinjau dari banyaknya pengunduh pada tahun 2020 selama 4 kuartal berturut.



Gambar 1.1 Peringkat unduhan aplikasi *e-commerce* Indonesia di Google Play Store, diambil dari [2]

Peringkat dan peningkatan pengunjung yang ada di Shopee juga berbanding lurus dengan jumlah *seller* yang ada. Ini dikarenakan oleh tingginya *traffic* di Shopee sehingga banyak yang tertarik untuk membuka lapaknya di sana. Pada kuartal kedua tahun 2020 Shopee berhasil mencatat peningkatan transaksi sebesar 260 juta dengan rata-rata transaksi per harinya mencapai 2,8 juta transaksi yang mana angka telah ini meningkat jauh lebih pesat dibandingkan tahun sebelumnya.

Dengan banyaknya *seller* yang ada, tidak semua *seller* di Shopee itu menjual produknya sendiri. Maksud dari menjual produknya sendiri adalah produk yang dijual merupakan produk yang *ready stock* atau hasil produksi sendiri. Ada juga beberapa penjual yang dalam tanda kutip tidak menggunakan modal dalam mengoperasikan tokonya. Penjual ini dikenal dengan sebutan *dropshipper*. *Dropshipper* biasanya hanya melayani dan mencari pembeli lalu ketika transaksi sudah *deal, dropshipper* tidak mengurus bagian pengemasan dan pengiriman produk. Berbeda dengan *reseller* yang mengharuskan penjual untuk membeli produk sebelum ditawarkan kepada konsumen, *dropshipper* tidak perlu membeli produk terlebih dahulu. Dengan kata lain, bisnis *online dropship* adalah menjual produk yang tidak dimiliki.



Gambar 1.2 Ilustrasi proses *dropshipping*, diambil dari [5]

Tentu sebagai konsumen, kita akan cenderung mencari produk dengan harga serendah mungkin namun dengan kualitas sebaik mungkin. Namun ada beberapa konsumen yang mengadopsi sifat ini secara berlebihan dan konsumen jenis ini bisa dibilang konsumen yang cerewet apabila ekspektasinya tidak terpenuhi. Melalui makalah ini penulis ingin membantu untuk mewujudkan keinginan para konsumen tersebut untuk memperoleh produk yang diinginkannya dengan harga serendah mungkin. Salah satu cara untuk memperoleh produk dengan harga yang lebih rendah adalah menghindari untuk membeli barang dari *dropshipper* karena biasanya sudah dilakukan *mark up* harga pada barang yang dijual oleh *dropshipper*. Sebaliknya, produk yang dibeli dari

seller asli cenderung memiliki harga yang relatif lebih rendah dan memungkinkan untuk memiliki kualitas yang sama dengan produk yang dijual oleh *dropshipper.* Dalam melakukan hal ini akan dimanfaatkan algoritma BFS dan DFS untuk melakukan penelusuran data setiap toko yang telah penulis kumpulkan dengan metode *web scraping*.

II. Dasar Teori

A. *Web Scraping*

Web scraping adalah metode untuk mengambil dan memproses informasi dari suatu *website*. Salah satu keuntungan dari *web scraping* dalam *e-commerce* adalah dapat memperoleh data untuk dianalisis. Pada kesempatan kali ini metode *web scraping* digunakan untuk mengekstrak data dari Shopee, yaitu untuk memperoleh data dari toko yang menjual '*tumblr tshirt*'.

Web scraping akan dilakukan dengan bantuan beberapa kakas seperti Selenium dan BeautifulSoup serta bahasa pemrograman Python. Penulis memilih Python karena Python merupakan *high-level programming language* serta dapat dijalankan di berbagai platform yang berbeda dan juga Python memiliki sintaks yang sederhana serta mudah untuk dimengerti. Selain itu, digunakan juga *browser* Microsoft Edge dan *browser* ini akan dioperasikan dengan bantuan Selenium. Sedangkan BeautifulSoup berfungsi untuk mengekstrak data tertentu dari halaman web, menghapus berbagai *tag* HTML yang digunakan dan kemudian menyimpan data yang sudah diekstrak.

Langkah pertama untuk mengekstrak informasi dari Shopee adalah dengan membuat sebuah fungsi untuk meng-*generate url* dari suatu pencarian dengan *keyword* tertentu. Kemudian buka halaman dari *url* yang telah diperoleh sebelumnya lalu ekstrak *link* dari semua produk yang ada di halaman tersebut. Untuk halaman pencarian produk di Shopee biasanya 1 halaman memuat 49-50 produk dan ada kemungkinan beberapa produk muncul secara berulang karena adanya penggunaan fitur iklan Shopee.

Berikut merupakan implementasi algoritma untuk langkah pertama dalam *web scraping* Shopee dalam bahasa Python.

```
# Generate an url from search form
def get_url(keyword):
    template =
'https://shopee.co.id/search?keyword={}'
    search_term = keyword.replace(' ','%20')
    return template.format(search_term)
```

```
# Retrieve all product links
def search(url):
    links = []

    # maximize window
    driver.maximize_window()

    # open link
    driver.get(url)

    # wait until page loads completely
    time.sleep(2)
    driver.execute_script('window.scrollTo(0, 1500);')
    time.sleep(2)
    driver.execute_script('window.scrollTo(0, 2800);')
    time.sleep(2)
    soup = BeautifulSoup(driver.page_source,
'html.parser')
    products = soup.find('div', class_='row
shopee-search-item-result__items')
    for link in products.find_all('a'):
        links.append(link.get('href'))
        print(link.get('href'))
    return links
```

Kemudian buka setiap *link* produk yang telah diperoleh sebelumnya dan ekstrak informasi yang diperlukan. Pada kasus ini, informasi yang diekstrak adalah *link* toko dan informasi pengiriman.

```
# extract product information
def extract_product(product_url):
    url = 'https://shopee.co.id' + product_url
    driver.get(url)

    timeout = 5
    WebDriverWait(driver, timeout).until(

EC.visibility_of_element_located((By.CLASS_NAME,
"_2Sv-zW")))

    # Click pilihan kurir
    e =
driver.find_elements_by_class_name('shopee-drawer')[1]
    e.click()

    # Mengumpulkan semua informasi pengiriman
    soupb = BeautifulSoup(driver.page_source,
'html.parser')
    allpengiriman = soupb.find_all(class_ = '_3NgYBG')
    pengiriman = []

    # Shop url
    shop_url =
'https://shopee.co.id'+soupb.find('a',{'class':"_267Jf
9"}).get('href')

    for p in allpengiriman:
        pengiriman.append(p.text)
    return(shop_url,pengiriman)
```

Sama dengan sebelumnya, buka setiap *link* toko yang diperoleh dan ekstrak informasi toko. Pada kasus ini, informasi toko yang diperlukan adalah nama toko, persentase toko dalam membalas *chat* dan *rating* toko.

```
# extract shop information
def extract_shop(shop_url):
    driver.get(shop_url)

    timeout = 7
```

```
    WebDriverWait(driver, timeout).until(

EC.visibility_of_element_located((By.CLASS_NAME,
"shop-page__info")))

    # Mengumpulkan semua informasi toko yang
dibutuhkan
    soupc = BeautifulSoup(driver.page_source,
'html.parser')
    namaToko = soupc.find(class_ =
'section-seller-overview-horizontal__portrait-name').t
ext
    respRate = get_digits(soupc.find_all(class_ =
'section-seller-overview__item-text-value')[4].text)
    shopRating = get_digits(soupc.find_all(class_ =
'section-seller-overview__item-text-value')[8].text)

    if(shopRating>5.0): # ada tingkat pembatalan
        shopRating = get_digits(soupc.find_all(class_
=
'section-seller-overview__item-text-value')[10].text)
    return(namaToko, respRate, shopRating)
```

Langkah terakhir, gabungkan semua fungsi yang telah dibuat sebelumnya. Berikut implementasi algoritma untuk menggabungkan semua fungsi secara keseluruhan.

```
def get_data(url):
    linkcomp = search(url)
    temprecord = []
    for l in linkcomp:
        temprecord.append(extract_product(l))

    # get all shops link
    allshoplink = []
    for t in temprecord:
        allshoplink.append(t[0])

    temprecord2 = []
    for l in allshoplink:
        temprecord2.append(extract_shop(l))

    data = []
    for i in range(50):
        data1 = []
        if(temprecord2[i]==None):
            data1.extend('')
        else:
            data1.extend(list(temprecord2[i]))

        if(temprecord[i]==None):
            data1.extend('')
        else:
            data1.extend(list(temprecord[i]))
      data1.append('https://shopee.co.id'+linkcomp[i])

        data.append(data1)
    return data
```

B. Struktur Data Pohon

Pohon adalah graf tak-berarah terhubung yang tidak mengandung sirkuit. Terdapat beberapa komponen penyusun struktur data pohon, yaitu simpul, sisi dan akar. Simpul adalah entitas yang mengandung sebuah nilai yang merepresentasikan data tertentu. Sisi adalah penghubung antar simpul. Akar adalah simpul teratas dari sebuah pohon.

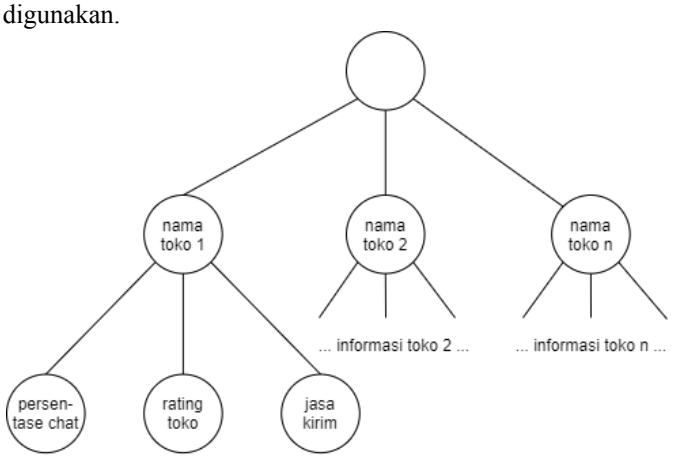
Data yang diekstrak dengan metode *web scraping* sebelumnya akan disimpan dalam bentuk pohon. Akar pohon dibiarkan kosong (tidak menyimpan informasi apapun) dan kemudian setiap nama toko akan disimpan sebagai simpul-simpul lain yang terhubung dengan simpul akar serta setiap informasi dari toko akan disimpan sebagai simpul baru yang terhubung dengan simpul nama toko.

Berikut potongan kode dalam pembentukan struktur data pohon.

```
class Tree:
    # Constructor
    def __init__(self, n, map):

        # initialize dict
        self.graph = {}
        self.nNode = n
        self.map = map

        list_n = list(range(n+1))
        for i in list_n:
            self.graph[i] = []

        # construct node
        root = 0
        temp = -99

        for i in range(1,n+1):
            if i % 4 == 1:
                temp = i
                self.addEdge(0,i)
            else:
                self.addEdge(temp,i)
```

Jumlah simpul yang ada pada pohon apabila dikurangi 1 (simpul akar) pasti merupakan kelipatan 4 karena data yang disimpan berjumlah 4 buah, yaitu nama toko, persentase *chat*, *rating* toko, dan informasi pengiriman. Pembuatan struktur data pohon disini memanfaatkan *dictionary* yang ada pada Python. Ada 2 pemanfaatan *dictionary* yang digunakan. Pertama, *dictionary* yang digunakan untuk merepresentasikan pohon itu sendiri, dengan melakukan pemetaan antara *key* yang merupakan id sebuah simpul dengan *value* yang berupa kumpulan id simpul lain yang merupakan anak dari simpul *key*. Sedangkan pemanfaatan *dictionary* yang kedua adalah untuk menyimpan informasi yang direpresentasikan oleh setiap id simpul. Sehingga dilakukan pemetaan antara setiap id simpul (berperan sebagai *key*) dengan informasi yang disimpan (berperan sebagai *value*).

Berikut merupakan ilustrasi dari struktur data pohon yang digunakan.



Gambar 2.1 Ilustrasi penyimpanan data dalam struktur pohon *n-ary*

Setelah struktur data pohon terbentuk, langkah selanjutnya adalah melakukan pengecekan pada setiap informasi toko untuk mengkategorikan apakah toko tersebut termasuk ke dalam *dropship* atau bukan. Pengecekan ini dilakukan dengan melakukan penelusuran pada setiap simpul yang ada pada pohon. Penelusuran yang dilakukan ada 2 cara, yaitu secara BFS dan DFS.

Toko yang masuk ke dalam kategori toko asli atau toko bukan *dropship* adalah toko yang memenuhi kriteria berikut.

Tabel 2.1 Kriteria toko asli dan *dropship*

Parameter	**Toko *Online***	
	Asli	***Dropship***
Response rate	≥ 90%	< 90%
Rating toko	≥ 4	< 4
Jasa kirim	Terdapat *same day* dan *instant service*	Hanya terdapat pengiriman *logistic*

Tentu terdapat beberapa alasan dibalik penentuan kriteria ini. Alasan-alasan tersebut antara lain adalah

1. *Response rate* minimal 90% untuk toko asli. Biasanya interaksi pertama antara penjual dan pembeli di *e-commerce* dimulai dari pembeli yang bertanya perihal ketersediaan barang. Umumnya penjual asli sangat mengetahui tentang ketersediaan barang yang dijualnya sehingga mereka dapat lebih cepat dan tanggap dalam memberikan respon. Sedangkan bagi *dropshipper*, mereka cenderung tidak terlalu mengetahui tentang ketersediaan barang yang dijualnya karena barang yang mereka jual bukanlah barang mereka sendiri. Hal ini tidak hanya berlaku untuk ketersediaan barang namun juga berlaku ketika pembeli menanyakan tentang spesifikasi barang.
2. *Rating* toko minimal 4 untuk toko asli. Biasanya toko asli cenderung memiliki *rating* yang lebih baik karena pelayanan yang lebih tanggap dan kesesuaian barang secara fisik dengan *display* barang yang ada pada halaman produk. Sementara foto produk yang *dropshipper* gunakan sebagai *display* di halaman produk biasanya bukan merupakan foto *real*. Sehingga konsumen cenderung tidak puas ketika bertransaksi di toko *dropship*.
3. Adanya *same day* dan *instant service* sebagai pilihan pengiriman pada toko asli. Produk yang toko asli jual biasanya merupakan produk yang secara fisik mereka simpan di dalam gudang mereka. Sehingga ketika ada transaksi yang menggunakan kurir *same day* ataupun *instant service*, maka penjual dapat langsung mengemas produknya dan kurir dapat langsung melakukan *pick up* untuk produk tersebut. Beda halnya dengan *dropshipper* yang biasanya tidak menyediakan opsi *same day* dan *instant* dalam opsi pengirimannya. Ini dikarenakan toko *dropship* yang tidak memiliki produk tersebut secara fisik dan pengemasan serta pengiriman produk dilakukan oleh pihak *supplier* dari *dropshipper* tersebut.

Perlu diketahui bahwa kriteria ini dibuat setelah melakukan pengamatan terhadap berbagai *seller* yang ada di Shopee. Sehingga kriteria ini tidaklah 100% akurat.

C. Algoritma Breadth First Search (BFS)

BFS atau dikenal dengan pencarian melebar adalah penelusuran data dalam struktur pohon yang dilakukan dengan cara mengunjungi semua simpul yang bertetangga dengan simpul yang sedang diproses saat itu. Misalnya terdapat sebuah pohon dengan *v* sebagai simpul akar, maka langkah pertama adalah kunjungi simpul *v* terlebih dahulu kemudian kunjungi semua simpul yang bertetangga dengan simpul *v* terlebih dahulu. Kunjungi simpul yang belum dikunjungi dan bertetangga dengan simpul-simpul yang tadi dikunjungi, demikian seterusnya.

Berikut merupakan implementasi algoritma BFS untuk melakukan penelusuran terhadap setiap simpul yang ada pada pohon. Penelusuran dimulai dari simpul dengan id 0 atau simpul akar. Penelusuran akan berhenti ketika semua simpul sudah diproses atau ditelusuri. Pada setiap simpul yang ditelusuri, akan dicek apakah simpul tersebut mengandung informasi yang memenuhi kriteria toko asli atau *dropship*. Kemudian apabila penelusuran berakhir maka fungsi akan memberi keluaran berupa data hasil pengecekan toko.

```
def BFS(self):
        s = 0
        data = []
        # inisialisasi array untuk visited node
        visited = [False] * (max(self.graph) + 1)

        # membuat queue
        queue = []

        # tandai node s sbg visited lalu enqueue
        queue.append(s)
        visited[s] = True

        while queue:
            # dequeue simpul dari queue
            s = queue.pop(0)
            dt = []
```

```
            # proses s jika bukan akar
            if s == 0:
                pass

            elif s % 4 == 1:
                # nama toko
                dt = [self.map[s],-99,-99,-99]
                data.append(dt)

            else:
                # atribut toko
                temp = -99
                if isinstance(self.map[s],float):
                    if(self.map[s]>5):
                        # persentase chat
                        if(self.map[s]>=90):
                            temp = 'Asli'
                        else:
                            temp = 'Dropship'

                    else:
                        # rating toko
                        if(self.map[s]>=4):
                            temp = 'Asli'
                        else:
                            temp = 'Dropship'

                else:
                    if 'Same Day' in self.map[s] and
'Instant' in self.map[s]:
                        temp = 'Terdapat same day dan
instant services'
                    else:
                        temp = 'Hanya terdapat kurir
logistik'
                hasil, sisa = self.division(s-1)
                data[hasil][sisa] = temp

            # kunjungi simpul tetangganya
            for i in self.graph[s]:
                if visited[i] == False:
                    queue.append(i)
                    visited[i] = True
        return data
```

D. Algoritma Depth First Search (DFS)

DFS atau dikenal dengan pencarian mendalam adalah penelusuran data dalam struktur pohon yang dilakukan dengan cara mengunjungi tetangga dari simpul yang diproses kemudian mengunjungi tetangganya lagi, ini dilakukan secara terus menerus sampai simpul yang diproses tidak memiliki tetangga lagi atau merupakan simpul daun. Misal terdapat pohon dengan simpul *v* sebagai simpul akar, maka langkah pertama adalah kunjungi simpul *v* itu. Kemudian kunjungi simpul *w* yang bertetangga dengan simpul *v*. Ulangi DFS dari simpul *w*. Ketika mencapai simpul *u* sedemikian sehingga semua simpul yang bertetangga dengannya telah dikunjungi, pencarian dirunut-balik (*backtrack*) ke simpul terakhir yang dikunjungi sebelumnya dan mempunyai simpul *w* yang belum dikunjungi. Pencarian berakhir bila tidak ada lagi simpul yang belum dikunjungi yang dapat dicapai dari simpul yang telah dikunjungi.

Berikut merupakan implementasi algoritma DFS untuk melakukan penelusuran terhadap setiap simpul yang ada pada pohon. Sama seperti penelusuran dengan menggunakan BFS, penelusuran dimulai dari simpul dengan id 0 atau simpul akar. Penelusuran akan berhenti ketika semua simpul sudah diproses atau ditelusuri. Pada setiap simpul yang ditelusuri, akan dicek apakah simpul tersebut mengandung informasi yang memenuhi kriteria toko asli atau *dropship*. Kemudian apabila penelusuran berakhir maka fungsi akan memberi keluaran berupa data hasil pengecekan toko. Pada proses penelusuran dengan DFS ini dibuat sebuah fungsi perantara yang akan dipanggil secara rekursif untuk membantu proses penelusuran setiap simpul.

```
def DFSUtil(self, v, visited, data):
        # tandai node visited
        visited.add(v)
        dt = []
        s = v
        if s == 0:
            pass

        elif s % 4 == 1:
            # nama toko
            dt = [self.map[s],-99,-99,-99]
            data.append(dt)

        else:
            # atribut toko
            temp = -99
            if isinstance(self.map[s],float):
                if(self.map[s]>5):
                    # persentase chat
                    if(self.map[s]>=90):
                        temp = 'Asli'
                    else:
                        temp = 'Dropship'
                else:
                    # rating toko
                    if(self.map[s]>=4):
                        temp = 'Asli'
                    else:
                        temp = 'Dropship'

            else:
                if 'Same Day' in self.map[s] and
'Instant' in self.map[s]:
                    temp = 'Terdapat same day dan
instant services'
                else:
                    temp = 'Hanya terdapat kurir
logistik'
            hasil, sisa = self.division(s-1)
            data[hasil][sisa] = temp

        # kunjungi tetangga dan panggil DFS lagi
        for neighbour in self.graph[v]:
            if neighbour not in visited:
                self.DFSUtil(neighbour, visited, data)
```

```
def DFS(self):
        # inisialisasi set untuk visited node
        visited = set()
        data = []

        # memanggil fungsi rekursif DFS
        self.DFSUtil(0, visited, data)
        return data
```

III. Pengujian dan Analisis

A. *Pengujian*

Pengujian dilakukan dengan program yang telah dibuat oleh penulis sesuai dengan potongan-potongan kode yang sudah terlampir sebelumnya. Pengujian terbagi menjadi 2 tahapan, yaitu tahap *web scraping* dan tahap pemindaian setiap toko untuk memeriksa apakah toko merupakan toko asli atau *dropship*.

Pada tahap *web scraping,* penulis menggunakan kakas Jupyter Notebook. Pada pengujian ini, jumlah data produk yang diekstrak adalah 100 data yang mana ini berarti *web scraping* dilakukan pada 2 halaman pertama hasil pencarian (1 halaman pencarian memuat 49-50 produk).

```
In [10]:
def get_100(search_term):
    temp_url = get_url(search_term)

    # first page
    data = get_data(temp_url)

    # second page
    data1 = get_data(temp_url+'&page={}'.format(str(1)))
    data = data + data1

    return data

In [11]:
start_time = time.perf_counter()
kaos = get_100("tumblr tshirt")
durasi = time.perf_counter() - start_time
kaos.append(durasi)

In [17]:
durasi

Out[17]: 662.0114893000009

In [18]:
len(kaos)

Out[18]: 100
```

Gambar 3.1 Hasil ekstrak 100 data produk

Dari gambar 3.1 terlihat bahwa banyak data yang berhasil terekstrak sebanyak 100 buah dengan durasi waktu kurang lebih 662 detik atau sekitar 11 menit. Data hasil ekstrak ini disimpan dalam *array* bernama kaos.

Berikut tampilan data yang tersimpan dalam *array* kaos setelah di-export menjadi *csv*.

Tabel 3.1 Hasil ekstrak 20 data produk pertama dalam format *csv*

nama toko	respon rate	rating toko	url toko	pengiriman
elbee_id	65	4.9	https://shc	['Reguler', 'Hemat
Cotton Addict	90	4.9	https://shc	['Reguler', 'Instant
vibel.collection	98	4.8	https://shc	['Reguler', 'Hemat
vibel.collection	98	4.8	https://shc	['Reguler', 'Hemat
Hypestore.Official	98	4.8	https://shc	['Reguler', 'Hemat
NajwaFashionMuslim	83	4.5	https://shc	['Reguler', 'Kargo'
ABAM Official Shop	86	4.8	https://shc	['Reguler', 'Hemat
lins_lino	88	4.9	https://shc	['Reguler']
Surya_Online_Supplier	82	4.6	https://shc	[]
frelynshop	73	4.8	https://shc	['Reguler']
Iseven Fashion Official	88	4.5	https://shc	['Reguler']
NajwaFashionMuslim	83	4.5	https://shc	['Reguler', 'Kargo'
Glowcollection88	48	4.6	https://shc	[]
Sofie Fashion	58	4.6	https://shc	['Reguler', 'Instant
Glowcollection88	48	4.6	https://shc	['Reguler', 'Kargo'
FINE Collection	37	4.5	https://shc	['Reguler', 'Kargo'
Noveline Official Shop	99	4.8	https://shc	['Reguler']
Surya_Online_Supplier	82	4.6	https://shc	['Reguler']
21 Fashion Store	81	4.5	https://shc	['Reguler']
FINE Collection	37	4.5	https://shc	[]

Selain nama toko, *response rate*, *rating* toko, *url* dan informasi pengiriman, terdapat juga informasi mengenai *url* produk di sebelah kanan kolom pengiriman. Namun karena keterbatasan ruang maka informasi tersebut tidak ditampilkan pada gambar 3.2.

Selanjutnya data hasil *web scraping* akan diolah kembali pada tahap pemindaian setiap toko. Pada tahap ini, data dikonversi terlebih dahulu menjadi struktur pohon. Setelah itu dilakukan penelusuran setiap simpul dalam pohon baik secara BFS maupun DFS.

```
C:\sem4\stima\makalah>python bd.py
Masukan nama file csv yang ingin diproses : baju100.csv
Masukan nama file output : result_baju.csv
Waktu proses dengan BFS : 0.000656 detik
Waktu proses dengan DFS : 0.001252 detik

--- Hasil Pemindaian Seluruh Toko ---
berdasarkan response rate   : 36 asli & 64 dropship
berdasarkan rating toko     : 100 asli & 0 dropship
berdasarkan opsi pengiriman : 9 asli & 91 dropship
```

Gambar 3.3 *Input output* program untuk mengolah data

Tabel 3.1 Hasil ekstrak 20 data produk pertama dalam format *csv*

Nama Toko	Persentase	Rating Tok	Pengiriman
elbee_id	Dropship	Asli	Hanya terdapat kurir logistik
Cotton Addict	Asli	Asli	Hanya terdapat kurir logistik
vibel.collection	Asli	Asli	Terdapat same day dan instant s
vibel.collection	Asli	Asli	Terdapat same day dan instant s
Hypestore.Official	Asli	Asli	Hanya terdapat kurir logistik
NajwaFashionMuslim	Dropship	Asli	Hanya terdapat kurir logistik
ABAM Official Shop	Dropship	Asli	Hanya terdapat kurir logistik
lins_lino	Dropship	Asli	Hanya terdapat kurir logistik
Surya_Online_Supplier	Dropship	Asli	Hanya terdapat kurir logistik
frelynshop	Dropship	Asli	Hanya terdapat kurir logistik
Iseven Fashion Official	Dropship	Asli	Hanya terdapat kurir logistik
NajwaFashionMuslim	Dropship	Asli	Hanya terdapat kurir logistik
Glowcollection88	Dropship	Asli	Hanya terdapat kurir logistik
Sofie Fashion	Dropship	Asli	Hanya terdapat kurir logistik
Glowcollection88	Dropship	Asli	Hanya terdapat kurir logistik
FINE Collection	Dropship	Asli	Hanya terdapat kurir logistik
Noveline Official Shop	Asli	Asli	Hanya terdapat kurir logistik
Surya_Online_Supplier	Dropship	Asli	Hanya terdapat kurir logistik
21 Fashion Store	Dropship	Asli	Hanya terdapat kurir logistik
FINE Collection	Dropship	Asli	Hanya terdapat kurir logistik

B. Analisis

Waktu yang dibutuhkan untuk menjalankan program secara keseluruhan berada di kisaran 11 menit. Program sendiri dijalankan dengan menggunakan laptop pribadi penulis yang memiliki spesifikasi sebagai berikut.

- Operating System: Windows 10
- Processor: Intel Core i3-7100U Processor (3 MB Cache, 2.4 GHz)
- Memory: 4096 RAM

Waktu untuk menjalankan program merupakan akumulasi dari waktu pada tahap *web scraping* dan waktu pada tahap pemindaian setiap toko. Waktu pada tahap *web scraping* bisa dibilang cukup lama yaitu berada di kisaran 11 menit. Hal ini dikarenakan tahap *web scraping* sangat bergantung pada kecepatan internet dalam meload setiap halaman yang ingin diambil informasinya. Apabila koneksi internet terputus maka proses *web scraping* mungkin gagal karena *timeout*. Sedangkan untuk waktu pada tahap pemindaian toko relatif cepat yaitu dibawah 1 detik baik ketika menggunakan BFS maupun DFS. Hal ini mungkin dikarenakan oleh jumlah data yang digunakan relatif sedikit, yaitu 100 data.

Pada tabel 3.1 dan tabel 3.2 dapat dilihat terdapat beberapa pengulangan data toko. Hal ini dapat disebabkan oleh 2 faktor, yaitu adanya penggunaan layanan iklan oleh penjual atau terdapat lebih dari 1 produk yang dijual oleh penjual yang sama. Kemudian juga terdapat beberapa toko yang tidak memiliki jasa pengiriman yang dapat dilihat pada tabel 3.1. Ini dikarenakan toko tersebut masuk ke dalam kategori Shopee *mall* dimana proses *web scraping* tidak bisa mengekstrak informasi pengiriman dari toko yang merupakan Shopee *mall*.

IV. KESIMPULAN

Hasil pemindaian setiap produk menunjukan bahwa terdapat 36 toko asli dan 64 toko *dropship* berdasarkan parameter persentase respon toko, 100 toko asli dan 0 toko *dropship* berdasarkan parameter *rating* toko, dan 9 toko asli serta 91 toko *dropship* berdasarkan parameter berupa opsi pengiriman yang disediakan toko. Pengecekan dilakukan per parameter dan bukan secara keseluruhan sehingga pengkategorian sebuah toko asli atau *dropship* berdasarkan pada parameter *response rate, rating* toko atau opsi pengiriman yang disediakan. Berdasarkan pada pengujian yang sudah dilakukan, program untuk melakukan pengecekan toko *online* di Shopee ini bisa dibilang dapat berjalan dengan cukup baik meskipun belum sempurna. Adanya penggunaan fitur iklan produk ataupun adanya lebih dari 1 produk yang dijual oleh *seller* yang sama mengakibatkan jumlah toko yang dipindai kurang dari 100 toko.

VIDEO LINK AT YOUTUBE

Berikut penulis lampirkan video penjelasan untuk menambah pemahaman pembaca. Video tersebut dapat diakses melalui *link* berikut https://youtu.be/kGlIGFOVvXo

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada Tuhan Yang Maha Esa dan juga kepada Bapak dan Ibu dosen pengampu mata kuliah IF2211 karena sudah membimbing penulis dan membekali penulis dengan berbagai pengetahuan baru di semester ini. Penulis juga berterima kasih karena melalui tugas makalah ini, penulis berkesempatan untuk mengeksplorasi hal-hal baru khususnya mengimplementasikan algoritma BFS dan DFS dalam kehidupan nyata.

REFERENCES

[1] Daftar 50 Website & Aplikasi E-Commerce di Indonesia 2019 https://iprice.co.id/insights/mapofecommerce/ diakses pada 9 Mei 2021 pukul 09.30

[2] Wafl, R. N. (n.d.). Tech in Asia Indonesia - Komunitas Online Startup di Asia https://id.techinasia.com/iprice-shopee-e-commerce-terpopuler-2020#:~:text=Data%20iPrice%20menunjukkan%20kinerja%20Shopee,persen%20dibanding%20kuartal%20ketiga%202019. diakses pada 9 Mei 2021 pukul 10.00

[3] Setiawan, S. R. D. (Ed.). (2020, September 1). *Shopee Bukukan 260 Juta Transaksi pada Kuartal II 2020 Halaman all*. KOMPAS.com. Shopee Bukukan 260 Juta Transaksi pada Kuartal II 2020 Halaman all - Kompas.com

[4] Dwi. (2020, July 14). Mengupas Tuntas Bisnis *Reseller* dan *Dropship* dari A Sampai Z Storelogy. Mengupas Tuntas Bisnis Reseller dan Dropship dari A Sampai Z - Storelogy

[5] *Deco Sign & Code B.V. - Groothandel en importeur winkelmaterialen.* Deco Sign & Code. (2021, May 4). Kies bij Deco Sign & Code BV voor drop shipment / drop shipping (sell2shops.com)

[6] Henrys, K. (2021). Importance of Web Scraping in E-commerce and E-marketing. *SSRN Electronic Journal.* https://doi.org/10.2139/ssrn.3769593

[7] Nurdin, Hutomi, M., Qamal, M., & Bustami, B. (2020). Sistem Pengecekan Toko Online Asli atau Dropship pada Shopee Menggunakan Algoritma Breadth First Search. *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi), 4(6).* https://doi.org/10.29207/resti.v4i6.2514

[8] Munir, Rinaldi .Diktat Kuliah IF2211 Strategi Algoritmik. Institut Teknologi Bandung. 2007.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Jakarta, 11 Mei 2021

Jesica 13519011