Laporan Praktikum Kecerdasan Buatan

Semester Ganjil 2023/2024

Jurusan Teknik Elektro
Program Studi S1 Teknik Elektro
Fakultas Teknik
Universitas Tidar

Praktikum ke- : 2
Judul Praktikum : Pencarian

Nama : Jerico Christianto
NIM : 2220501082

“Laporan praktikum ini saya kerjakan dan selesaikan dengan sebaik-baiknya tanpa melakukan tindak kecurangan. Apabila di kemudian hari ditemukan adanya kecurangan pada laporan ini, maka saya bersedia menerima konsekuensinya.”

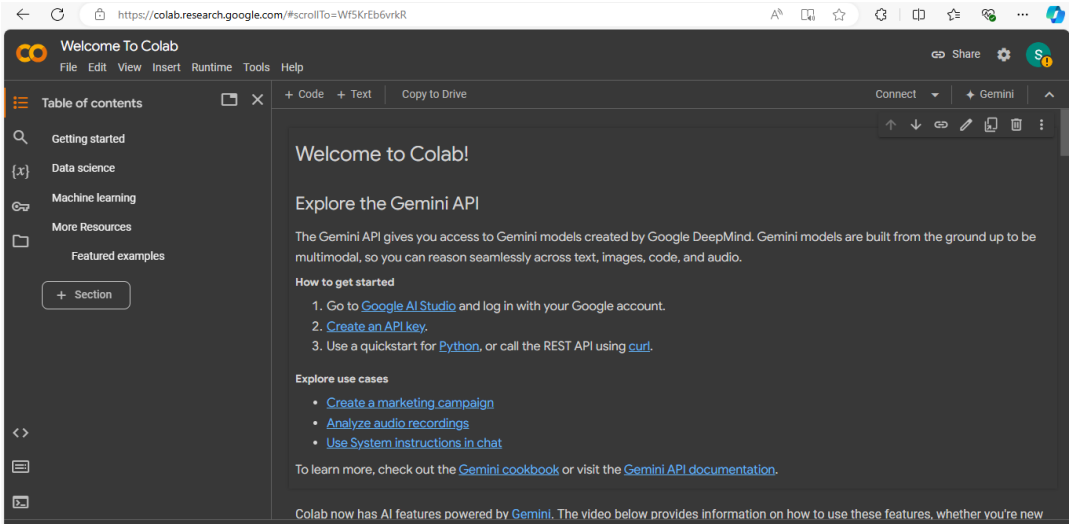
Tertanda,

Jerico Christianto

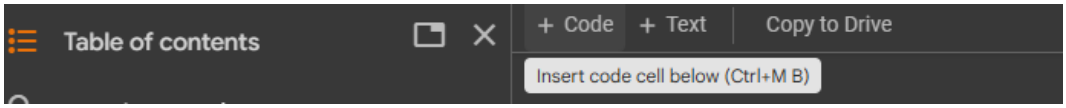
2220501082

A. Tujuan Praktikum
   1. Memahami konsep dasar algoritma pencarian
   2. Menerapkan algoritma pencarian dalam bahasa pemrograman Python
   3. Menerapkan algoritma pencarian Blind Search dalam Python

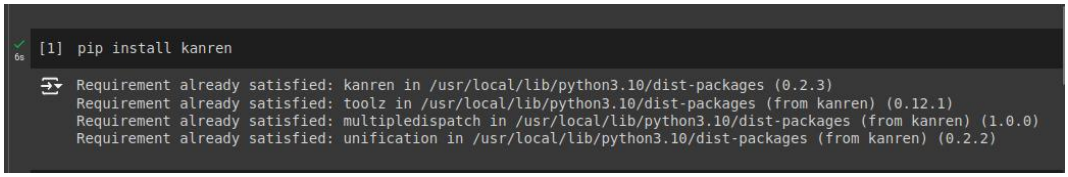
B. Langkah Praktikum
   1. Membuka Google Colab



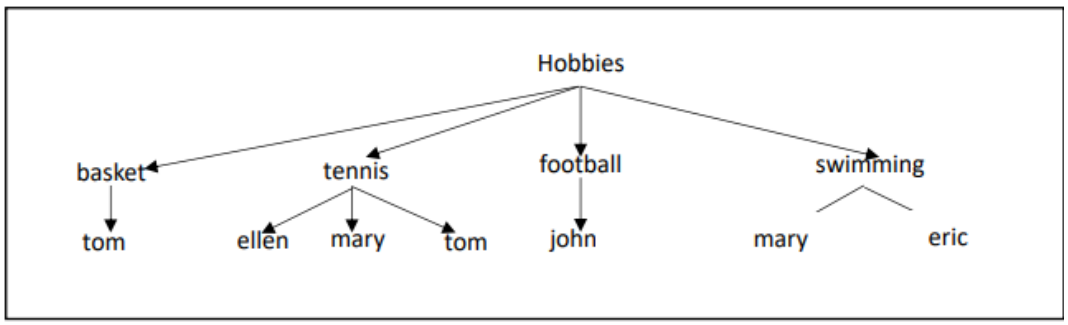
   2. Membuat kode baru



   3. Jalankan perintah “pip install kanren” untuk menginstall kanren



   4. Terdapat sebuah ilustrasi kasus pencarian hobi seperti berikut:



5. Tulis kode program yang digunakan untuk melakukan pencarian DFS pada kasus tersebut seperti berikut:

```
from kanren.facts import Relation, facts
from kanren.core import var, run
from collections.abc import Iterable
suka = Relation()
facts(suka, ("ellen", "tenis"),
             ("john", "football"),
             ("mary", "swimming"),
             ("tom", "tenis"),
             ("tom", "basket"),
             ("eric", "swimming"),
             ("mary", "tenis"))
x = var()
tom_hobbies = run(0, x, suka("tom",x))
print("Tom: ", tom_hobbies)

for hobby in tom_hobbies:
    facts(suka,("bill"), hobby)
bill_hobbies = run(0,x,suka("bill",x))
print("bill:", bill_hobbies)

mary_hobbies = run(0,x,suka("mary",x))
print("mary:", mary_hobbies)
```

```
    facts(suka,("bill"), hobby)
bill_hobbies = run(0,x,suka("bill",x))
print("bill:", bill_hobbies)

mary_hobbies = run(0,x,suka("mary",x))
print("mary:", mary_hobbies)

for hobby in mary_hobbies:
    facts(suka, ("ann"), hobby)
ann_hobbies = run(0,x,suka("ann",x))
print("Ann: ", ann_hobbies)
```

6. Modifikasi kode program menjadi seperti berikut:

```
from kanren.facts import Relation, facts, fact
from kanren.core import var, run
from kanren.goals import membero
from collections.abc import Iterable
suka = Relation()
facts(suka, ("ellen", "tenis"),
             ("john", "football"),
             ("mary", "swimming"),
             ("tom", "tenis"),
             ("tom", "basket"),
             ("eric", "swimming"),
             ("mary", "tenis"))
x = var()
tom_hobbies = run(0, x, suka("tom",x))
print("Tom: ", tom_hobbies)

for hobby in tom_hobbies:
    fact(suka,("bill"), hobby)
bill_hobbies = run(0,x,suka("bill",x))
print("bill:", bill_hobbies)

mary_hobbies = run(0,x,suka("mary",x))
print("mary:", mary_hobbies)

for hobby in mary_hobbies:
    fact(suka, ("ann"), hobby)
ann_hobbies = run(0,x,suka("ann",x))
print("Ann: ", ann_hobbies)
```

C. Hasil Praktikum

1. Melakukan Pencarian DFS Pada Sebuah Kasus

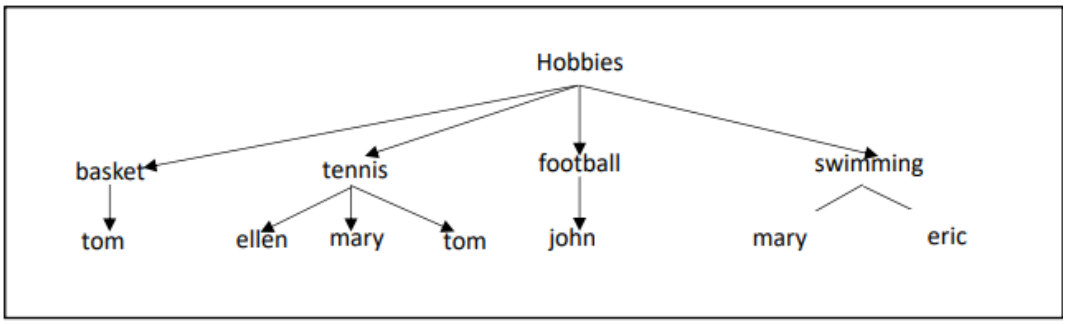


Diagram diatas menggambarkan sebuah pohon (tree) yang mewakili hubungan antara hobi dan orang yang memiliki hobi tersebut. Setiap cabang dari pohon merepresentasikan sebuah hobi, dan daun-daun pada pohon merepresentasikan orang-orang yang menyukai hobi tersebut.

Depth First Search (DFS) adalah sebuah algoritma pencarian pada struktur data seperti pohon atau graf. Prinsip kerjanya adalah menjelajahi sejauh mungkin pada satu cabang sebelum beralih ke cabang lainnya. Dalam konteks diagram tersebut, DFS akan mengunjungi setiap anak dari sebuah node (hobi) sebelum kembali ke node induknya.

2. Kode program yang digunakan untuk melakukan pencarian DFS pada kasus tersebut:

```
from kanren.facts import Relation, facts
from kanren.core import var, run
from collections.abc import Iterable
suka = Relation()
facts(suka, ("ellen", "tenis"),
             ("john", "football"),
             ("mary", "swimming"),
             ("tom", "tenis"),
             ("tom", "basket"),
             ("eric", "swimming"),
             ("mary", "tenis"))
x = var()
tom_hobbies = run(0, x, suka("tom",x))
print("Tom: ", tom_hobbies)

for hobby in tom_hobbies:
    facts(suka,("bill"), hobby)
bill_hobbies = run(0,x,suka("bill",x))
print("bill:", bill_hobbies)

mary_hobbies = run(0,x,suka("mary",x))
print("mary:", mary_hobbies)

for hobby in mary_hobbies:
    facts(suka, ("ann"), hobby)
ann_hobbies = run(0,x,suka("ann",x))
print("Ann: ", ann_hobbies)
```

Kita membuat sebuah dictionary hobbies untuk merepresentasikan pohon. Kunci dari dictionary adalah nama hobi, dan nilainya adalah list dari orang-orang yang menyukai hobi tersebut.

- Mendefinisikan variabel ‘suka’ sebagai relasi

```
suka = Relation()
```

- Menambahkan fakta ‘suka’ dengan nama dan hobi

```
facts(suka, ("ellen", "tenis"),
            ("john", "football"),
            ("mary", "swimming"),
            ("tom", "tenis"),
            ("tom", "basket"),
            ("eric", "swimming"),
            ("mary", "tenis"))
```

- Mencari relasi fakta ‘suka’ dengan “tom” yg dimasukkan ke variabel tom_hobbies

```
x = var()
tom_hobbies = run(0, x, suka("tom",x))
print("Tom: ", tom_hobbies)
```

- Menambahkan fakta hobi dari “Bill” yang nilainya akan sama dengan setiap hobi di tom_hobbies dengan facts

```
for hobby in tom_hobbies:
    facts(suka,("bill"), hobby)
bill_hobbies = run(0,x,suka("bill",x))
print("bill:", bill_hobbies)
```

- Mencari relasi fakta ‘suka’ dengan “mary” yg dimaskukkan ke variabel mary_hobbies

```
mary_hobbies = run(0,x,suka("mary",x))
print("mary:", mary_hobbies)
```

- Menambahkan fakta hobi dari “Ann” yang nilainya akan sama dengan setiap hobi di mary_hobbies dengan facts

```
for hobby in mary_hobbies:
    facts(suka, ("ann"), hobby)
ann_hobbies = run(0,x,suka("ann",x))
print("Ann: ", ann_hobbies)
```

3. Hasil Eksekusi Kode Program

```
for hobby in mary_hobbies:
    facts(suka, ("ann"), hobby)
ann_hobbies = run(0,x,suka("ann",x))
print("Ann: ", ann_hobbies)
```

```
Tom:  ('basket', 'tenis')
bill: ()
mary: ('swimming', 'tenis')
Ann:  ()
```

Berdasarkan hasil eksekusi kode, dapat disimpulkan bahwa node Tom terhubung dengan node tenis dan basket, serta node Mary terhubung dengan node swimming dan tenis pada struktur data pohon yang telah didefinisikan. Sementara itu, tidak ada edge yang menghubungkan node Bill dan Ann dengan node-node hobi lainnya.

4. Modifikasi Kode Program

```
from kanren.facts import Relation, facts, fact
from kanren.core import var, run
from kanren.goals import membero
from collections.abc import Iterable
suka = Relation()
facts(suka, ("ellen", "tenis"),
             ("john", "football"),
             ("mary", "swimming"),
             ("tom", "tenis"),
             ("tom", "basket"),
             ("eric", "swimming"),
             ("mary", "tenis"))
x = var()
tom_hobbies = run(0, x, suka("tom",x))
print("Tom: ", tom_hobbies)

for hobby in tom_hobbies:
    fact(suka,("bill"), hobby)
bill_hobbies = run(0,x,suka("bill",x))
print("bill:", bill_hobbies)

mary_hobbies = run(0,x,suka("mary",x))
print("mary:", mary_hobbies)

for hobby in mary_hobbies:
    fact(suka, ("ann"), hobby)
ann_hobbies = run(0,x,suka("ann",x))
print("Ann: ", ann_hobbies)
```

5. Hasil Eksekusi Kode Program

```
print("mary:", mary_hobbies)

for hobby in mary_hobbies:
    fact(suka, ("ann"), hobby)
ann_hobbies = run(0,x,suka("ann",x))
print("Ann: ", ann_hobbies)

Tom:  ('basket', 'tenis')
bill: ('tenis', 'basket')
mary: ('swimming', 'tenis')
Ann:  ('swimming', 'tenis')
```

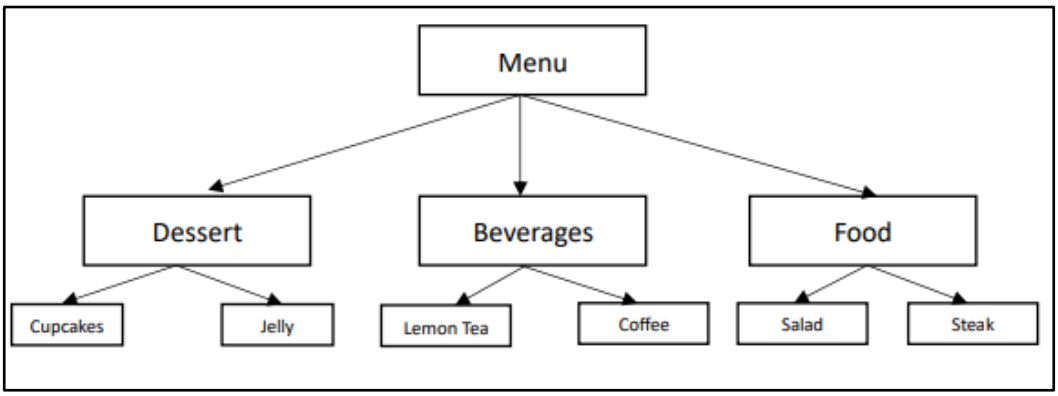
Setelah dilakukan modifikasi pada kode program, informasi mengenai hobi Bill dan Ann berhasil ditampilkan. Ternyata, Bill memiliki hobi yang sama dengan Tom, sedangkan Ann memiliki hobi yang sama dengan Mary. Hal ini dapat terjadi karena digunakannya sintaks fact untuk menambahkan data baru. Sintaks fact berfungsi untuk mengintegrasikan fakta baru (hobi Bill dan Ann) ke dalam fakta yang sudah ada sebelumnya (hobi Tom dan Mary).

D. Kendala Praktikum

Terdapat error saat menjalankan kode program pada jupyterlab sehingga digunakan Google Colab, sempat terdapat error yang berbunyi ImportError: cannot import name 'Iterable' from 'collections' , solusinya yaitu dengan mengganti nama collection.abc pada file header

E. Studi Kasus

1. Implementasikan pencarian Blind Search pada kasus berikut:



2. Kode Program untuk Pencarian Blind Search

```
from kanren.facts import Relation, facts, fact
from kanren.core import var, run
from kanren.goals import membero
from collections.abc import Iterable

jenis = Relation()
facts(jenis, ("menu", "dessert"),
             ("menu", "beverages"),
             ("menu", "food"),
             ("dessert", "cupcakes"),
             ("dessert", "jelly"),
             ("beverages", "lemon tea"),
             ("beverages", "coffee"),
             ("food", "salad"),
             ("food", "steak"))

x = var()
menu = run(0, x, jenis("menu",x))
print("menu: ", menu)

dessert_menu = run(0, x, jenis("dessert",x))
print("dessert: ", dessert_menu)

beverages_menu = run(0, x, jenis("beverages",x))
print("beverages: ", beverages_menu)

food_menu = run(0, x, jenis("food",x))
print("food: ", food_menu)

for semua_menu in dessert_menu, beverages_menu, food_menu:
    fact(jenis,("semua menu"), semua_menu)
semua_menu = run(0, x, jenis("semua menu",x))
print("semua menu: ", semua_menu)
```

```
menu :  ('beverages', 'food', 'dessert')
dessert  :  ('cupcakes', 'jelly')
beverages  :  ('lemon tea', 'coffee')
food :  ('salad', 'steak')
semua menu :  (('salad', 'steak'), ('lemon tea', 'coffee'), ('cupcakes', 'jelly'))
```

- Kode memulai dengan mendefinisikan relasi "jenis" yang menghubungkan antara konsep yang lebih umum (misalnya, "menu") dengan konsep yang lebih spesifik (misalnya, "dessert"). Kemudian, sejumlah fakta ditambahkan ke basis pengetahuan. Fakta-fakta ini menyatakan hubungan antara berbagai konsep.

Misalnya, fakta "(menu", "dessert")" menyatakan bahwa "dessert" adalah jenis dari "menu".

```
facts(jenis, ("menu", "dessert"),
             ("menu", "beverages"),
             ("menu", "food"),
             ("dessert", "cupcakes"),
             ("dessert", "jelly"),
             ("beverages", "lemon tea"),
             ("beverages", "coffee"),
             ("food", "salad"),
             ("food", "steak"))
```

- Setelah basis pengetahuan dibangun, kode melakukan beberapa pencarian menggunakan fungsi run.
    - Pencarian Pertama: Kode mencari semua nilai x yang memenuhi relasi "jenis" di mana konsep pertama adalah "menu". Hasilnya adalah semua jenis menu yang ada dalam basis pengetahuan.
    - Pencarian Selanjutnya: Kode melakukan pencarian serupa untuk mencari semua jenis "dessert", "beverages", dan "food".

```
x = var()
menu = run(0, x, jenis("menu",x))
print("menu: ", menu)

dessert_menu = run(0, x, jenis("dessert",x))
print("dessert: ", dessert_menu)
dessert_menu = run(0, x, jenis("dessert",x))
print("dessert: ", dessert_menu)

beverages_menu = run(0, x, jenis("beverages",x))
print("beverages: ", beverages_menu)

food_menu = run(0, x, jenis("food",x))
print("food: ", food_menu)
```

- Hasil dari pencarian jenis "dessert", "beverages", dan "food" kemudian digabungkan menjadi satu list yang disebut "semua menu". Ini menunjukkan bahwa semua jenis "dessert", "beverages", dan "food" sebenarnya adalah bagian dari "menu" yang lebih umum.

```
for semua_menu in dessert_menu, beverages_menu, food_menu:
    fact(jenis,("semua menu"), semua_menu)
semua_menu = run(0, x, jenis("semua menu",x))
print("semua menu: ", semua_menu)
```

3. Hasil

```
menu :  ('beverages', 'food', 'dessert')
dessert  :  ('cupcakes', 'jelly')
beverages  :  ('lemon tea', 'coffee')
food :  ('salad', 'steak')
semua menu :  (('salad', 'steak'), ('lemon tea', 'coffee'), ('cupcakes', 'jelly'))
```

Hasil pencariannya dapat diuraikan sebagai berikut:

- menu: Ini adalah kategori paling umum. Menu ini terdiri dari tiga jenis item: minuman (beverages), makanan (food), dan makanan penutup (dessert).
- dessert: Jenis makanan penutup yang tersedia adalah cupcakes dan jelly.
- beverages: Jenis minuman yang tersedia adalah lemon tea dan coffee.
- food: Jenis makanan utama yang tersedia adalah salad dan steak.
- semua menu: Ini adalah gabungan dari semua jenis makanan dan minuman yang ada, yaitu salad, steak, lemon tea, coffee, cupcakes, dan jelly.