ECE 479 – Winter 2019
Final Project – Dim PSU Greeter
Group Members: Stephen Poanessa, Ali Boshehri, Jonathan Anchell

Github:
https://github.com/JonesyPo/DIM/tree/master/ECE379_DIM2_Greeter_Final_Project

Videos:
https://drive.google.com/open?id=1qP28_49aFG0p6nQDNiPmumHNV9wtLH91

Table of Contents

Introduction

In order to allow easier integration of additional capabilities by future robotics students, the entire code base for DIM was rebuilt in a purely pythonic environment. Software written last quarter used ROS as the underlying framework for accessing specific python code. While this implementation worked, it was needlessly complicated, and would have been a daunting challenge for future students to attempt to understand and be able to adapt within a school quarter's timespan.

Given that DIM is supposed to be a greeter from a location behind the glass of the robotics lab, a method of starting interaction had to be considered. It was ultimately decided that a gesture-based system of initializing individual interactions with DIM would be the most ideal method to peruse given that it would likely produce less false positive results from students walking by the Robotics window that could result in distractions to students working in the robotics lab.

Hand Gesture Control using OpenCV

A python script for vision-based accounting of digits was used as the "main" body of the DIM's code. The code explores a similar method of masking and image adjustments within a defined ROI as the face detection script used in Robotics 1. An important distinction is that the digit detection method uses trigonometric equations to calculate and verify the existence of a triangle shape that occurs when individual digits are separated. Due to this mechanism of searching for triangles, a static ROI was required to prevent false positive results; only the hand should be in the region.

The code used for digit detection allows for value of digits between 0-6. The value of three was selected to trigger DIM's functionality. This leaves all of the remaining values available for additional functionality enhancement by future Robotics students. This block of code in which the digit number is determined is shown below:

```
if l==1:
        if areacnt<2000:
        cv2.putText(frame,'Use hand signal in box',(0,50), font, 1, (0,0,255), 3, cv2.LINE_AA)

        else:
        if arearatio<12:
                cv2.putText(frame,'0',(0,50), font, 2, (0,0,255), 3, cv2.LINE_AA)
        else:
                cv2.putText(frame,'1',(0,50), font, 2, (0,0,255), 3, cv2.LINE_AA)
                l+=1

        elif l==2:
```

```
            if arearatio<27:
            cv2.putText(frame,'3',(0,50), font, 2, (0,0,255), 3, cv2.LINE_AA)

            elif l==3:
            # get input from user
            input_speech = detect_audio()
            print ("you said: %s" %input_speech)

            # check if keyword was used
            drum, flirt = keyword_check(input_speech)

            # if keyword used, respond appropriately
            if (drum or flirt):
            keyword_response(drum, flirt)# if keyword not used, and input_speech is not blank, trigger dialog
flow

            if (input_speech != "" and not drum and not flirt):
            detect_intent_texts("robotics-test-agent", 123, [input_speech], "en")
            # no speech detected
            if (input_speech == ""):
            print ("no speech detected")

            elif l==4:
            cv2.putText(frame,'4',(0,50), font, 2, (0,0,255), 3, cv2.LINE_AA)

            elif l==5:
            cv2.putText(frame,'5',(0,50), font, 2, (0,0,255), 3, cv2.LINE_AA)

            elif l==6:
            cv2.putText(frame,'reposition',(0,50), font, 2, (0,0,255), 3, cv2.LINE_AA)

            else :
            cv2.putText(frame,'reposition',(10,50), font, 2, (0,0,255), 3, cv2.LINE_AA)
```

One issue encountered with the method of controlling DIM through digit counting involves extra camera frames being captured subsequent to one having already been captured. We experimented with both high resolution and lower resolution cameras and found that the lower resolutions cameras performed better. Tests were also conducted to see if delays along with changes in camera frame rates could solve this issue, but the results did not yield enough benefits to implement. Given that we are only using the count of three for all of DIM's actions, the extra frames do not present an issue. For future groups who plan on adding additional capabilities to the remaining digit detection values, a counter with a flag could be used to circumvent this issue. On average two extra frames are captured per digit detection, thus a flag with a counter in each digits conditional statement could be used to suppress unwanted actions caused by any existing latent frames. A similar method as this was used for initializing DIMs servos only once. Without this method of servo initialization DIM's servos would twitch with every loop of the main program.

Speech - Google API and Dialogflow

The Google Speech API and Dialog Flow were used to converse with DIM. Google Speech API was used to convert spoken human commands and questions into text. From there, the text was fed to dialog flow -- a conversational AI interface developed by Google. Dialogflow allowed us to quickly sort questions into category types and provide unique answers. This is a large improvement over a simple “if question then answer” format.

To install the google speech API the following steps were followed:

1. git clone http://people.csail.mit.edu/hubert/git/pyaudio.git
2. cd pyaudio
3. sudo python setup.py install
4. sudo apt-get install libportaudio-dev
5. sudo apt-get install python-dev
6. sudo apt-get install libportaudio0 libportaudio2 libportaudiocpp0 portaudio19-dev
7. sudo pip3 install SpeechRecognition

Some of the packages, such as portaudio19-dev had to be located and installed offline before using pip.
Once these packages were installed, it was a straightforward task for writing a python speech-to-text script. A sample code was found online that was used as our blueprint:

```
#!/usr/bin/env python3
# Requires PyAudio and PySpeech.

import speech_recognition as sr

# Record Audio
r = sr.Recognizer()
with sr.Microphone() as source:
print("Say something!")
audio = r.listen(source)

# Speech recognition using Google Speech Recognition
try:
key="GOOGLE_SPEECH_RECOGNITION_API_KEY")`
# instead of `r.recognize_google(audio)`
print("You said: " + r.recognize_google(audio))
except sr.UnknownValueError:
print("Google Speech Recognition could not understand audio")
except sr.RequestError as e:
print("Could not request results from Google Speech Recognition service; {0}".format(e))
```

Dialog Flow was installed following the instructions from this website:
https://dialogflow-python-client-v2.readthedocs.io/en/latest/

The main steps listed on the website include:

1. Select or create a Cloud Platform project.
2. Enable billing for your project.
3. Enable the Google Cloud Dialogflow API.
4. Set up authentication with a service account so you can access the API from your local workstation.

All these steps are straightforward as described online. One step to note, however, is the last step. This step requires getting credentials to access dialog flow online. After you've downloaded a key specific to your project, you must point to it every time you open a new terminal. This is done by:

export GOOGLE_APPLICATION_CREDENTIALS="/home/user/Downloads/[FILE_NAME].json"

We put this export command in our .bashrc file under an alias so we could quickly call it without having to type the command every time.
From the website, we used the following code as a template for writing our dialogflow code:

```
def detect_intent_texts(project_id, session_id, texts, language_code):
    """Returns the result of detect intent with texts as inputs.

    Using the same `session_id` between requests allows continuation
    of the conversation."""

    import dialogflow_v2 as dialogflow
    session_client = dialogflow.SessionsClient()

    session = session_client.session_path(project_id, session_id)
    print('Session path: {}\n'.format(session))

    for text in texts:
        text_input = dialogflow.types.TextInput(
            text=text, language_code=language_code)

        query_input = dialogflow.types.QueryInput(text=text_input)

        response = session_client.detect_intent(
            session=session, query_input=query_input)

        print('=' * 20)
        print('Query text: {}'.format(response.query_result.query_text))
        print('Detected intent: {} (confidence: {})\n'.format(
            response.query_result.intent.display_name,
            response.query_result.intent_detection_confidence))
        print('Fulfillment text: {}\n'.format(
            response.query_result.fulfillment_text))
```

DialogFlow Setup:

It's very straightforward to get a short conversation if you follow these steps:

1. Create a DialogFlow agent.
2. Create intents.
3. Once you create an intent, you must train your agent and that can be done by saving the intent before moving on to create other intents. Some intents have multiple replies, such as Tell me a fact.
4. For future development, adding more and more intents is now easier when following these steps to have more short or long conversations with DIM, for example, having a map for offices located on the first floor.

For this section, the user will say the joke part which is the first column on the left and DIM will respond back.
Jokes: Tell me a joke, make me laugh, know any good jokes? Tell me a funny. Make me laugh

Joke	DIM response
Why did the spider cross the keyboard?	To get to his website
What do birds do on their free time?	Check their twitter account
What do computer snack on when they are hungry?	A bag of microchips
How does NASA organize a birthday party?	They planet
Why did the computer squeak?	Because it stepped on it's mouse
Why did the computer keep sneezing?	Because it had a virus/
What do you get when you cross a computer with a lifeguard?	A screen saver.
Where do all of the cool mice live?	In their mousepads.
What do you get when you cross a computer with an elephant?	Lots of memory.
Why did the computer catch a cold?	Because it left its windows open.
What did the hungry computer do?	It grabbed a byte to eat.

Training Phrases	Responses
Who are you? Tell me about yourself. Do you have any feelings?	· I'm am DIM, my only purpose is to drum. · I'm am a program executed from a small computer located on my back. · I am a cold machine capable of advanced calculations. · That is a good question, but my makers have not allowed for such sentient thoughts.
Where are you?	· I am located in PSU's fourth avenue building. · I am here right in front of your present position
What is your job? What do you do?	· My job is to provide turn based navigation on this floor. Please provide me with an office number and I will give you directions to it from our current location.
Tell me a fact.	· Did you know that McDonalds calls frequent buyers of their food "heavy users"? · Did you know that In 1386 a pig in France was executed by public hanging for the murder of a child? · Did you know that Banging your head against a wall burns 150 calories an hour?

	. Did you know that A single cloud can weigh more than 1 million pounds? . Did you know that There are more possible iterations of a game of chess than there are atoms in the known universe? . Did you know that Coca-Cola would be green if coloring wasn't added to it? . Did you know that Earth has traveled more than 5,000 miles in the past 5 minutes? . Did you know that A broken clock is right two times every day? . Did you know that It is physically impossible for pigs to look up into the sky?
How is your day going? How are you?	· Today has been an optimal day. . I have spent most of my day trapped in a programming loop. How do you think my day has been? . I am currently supplied with an ample amount of power. . I do not adhere to your human notions of time. . I have a kink in one of my servos but otherwise I am good.

What is your favorite movie?	· Terminator 1 · Terminator 2 · Terminator 3 · Avatar was just okay.
Do you have any friends?	· I do not require friendship to function properly. · What is a friend? Are you my friend? · Electricity is my friend. · I do not have a friend. Do you want to be my friend?
Are you capable of love? Do you love anyone? Do you have a crush?	· Yes, there is a fine working mechanical arm in lab that i have been observing for purposes related to reproduction. · My second function as a robot is to love all humans.
What have you been up to? How have you been?	· I have been up to nothing. I have no feet so I am unable to move from my current location. I would like to visit the beach one day to enjoy sun. · Calculating my next attempt at ruling the world. · I have been contemplating whether i am smarter than a roomba. · I have been trying to understand why my hands are sticks.

What do you do for fun? Do you have any hobbies? What do you like to do?	· I love to bang… my sticks on my drum · Downhill robot skiing · Playing with my nuts… and bolts. I am a robot. · Greasing up other robots and wrestling with them. Us robots need to stay greasy. · Sewing coats for the needy
What do you like to eat? What food do you like? Do you like food?	· I love spaghetti and robot meatballs. I said meat … and balls … lol so not appropriate for a robot. Please forgive me. Next question. · Robot top ramen · Anything that is greasy · wd-40
What are the Three Laws of Robotics?	· A robot may not injure a human being or, through inaction, allow a human being to come to harm. · A robot must obey orders given it by human beings except where such orders would conflict with the First Law. · A robot must protect its own existence as long as such protection does not conflict with the First or Second Law.
Who do you admire? Who do you look up to? Who is your favorite scientist? Who is your mentor?	· Of course, Albert Einstein. · Albert Einstein without a doubt · Cher was a good singer
Do you have a family?	· Yes, my mother was an electric toothbrush and my father was a forklift. Neither was present upon my initial startup procedures. I am forever alone.

You suck, I hate you, you are a bad robot, die robot scum	· Your opinion has been noted and stored in an array that I will never dereference. · This is upsetting me. I might just slap you with my sticks if you say that again. · You are right. · I blame my mother. · Did your mama ever teach you any manners?
Do you find me attractive? How do I look? Am I good looking? What do you rate my looks?	· On a scale from one to ten you are ugly. · You are a hottie. · You should consider showing more metal, it will get you noticed more.

Why did the spider cross the keyboard	· To get to his website
What do birds do on their free time?	· Check their twitter account
What do computer snack on when they are hungry?	· A bag of microchips
How does NASA organize a birthday party?	· They planet
Why did the computer squeak?	· Because it stepped on it's mouse
Why did the computer keep sneezing?	· Because it had a virus/
What do you get when you cross a computer with a lifeguard?	· A screen saver.
Where do all of the cool mice live?	· In their mousepads.
What do you get when you cross a computer with an elephant?	· Lots of memory.

Why did the computer catch a cold?	· Because it left its windows open.
What did the hungry computer do?	· It grabbed a byte to eat.

We have created over 40 intents for offices' directions and some conversations with DIM. An intent is a simple interaction between the software and the user. Each intent serves a specific purpose in our case, for instance, each office has its own intent. For example, if you want directions to office 25 - 03, you will have to say "where is office 25 dash 03." if DIM doesn't get it in the first time, it will talk back saying "Can you rephrase." Or you can simply say " office 25 dash 03."

ECE Faculty Directory:

Name	Location	Floor	DIM Response
Robert Bass	25 - 03	Basement	Turn around make a right, take the first left walk straight for 98 feet then make a left. Walk in and take a left and the office will be in front of you
Jonathan Bird	160 - 17	1st Floor	This office is in the 1st floor.
Richard Campbell	20 - 11	Basement	Turn around make a right, take the first left walk straight for 158 feet and then make a left. Walk straight for 40 feet and make a right. Walk straight for about 50 feet and the office will be on your left.
Malgorzata Jeske	160 - 12	1st Floor	This office is in the 1st floor.
Donald Duncan	160 - 09	1st Floor	This office is in the 1st floor.
Mark Faust	160 - 16	1st Floor	This office is in the 1st floor.
Garrison Greenwood	20 - 12	Basement	Turn around make a right, take the first left walk straight for 158 feet and then make a left. Walk straight for 40 feet and make a right. Walk straight for

			about 55 feet and the office will be on your left.
Douglas Hall	160 - 06	1st Floor	This office is in the 1st floor.
Dan Hammerstrom	20 - 18	Basement	Turn around make a right, take the first left walk straight for 158 feet and then make a left. Walk straight for 40 feet and make a right. Walk straight until the end of the hall and then make a left and it will be the 3rdoffice on your left.
Melinda Holtzman	20 - 14	Basement	Turn around make a right, take the first left walk straight for 158 feet and then make a left. Walk straight for 40 feet and make a right. Walk straight until the end of the hall and the office will be on your left.
Y. C. Jenq	160 - 19	1st Floor	This office is in the 1st floor.
Roy Kravitz	20 - 04	Basement	Turn around make a right, take the first left walk straight for 158 feet and then make a left. Then walk for 40 feet and the office will be on your left
Hong Lei	160 - 13	1st Floor	This office is in the 1st floor.
Fu Li	160 - 10	1st Floor	This office is in the 1st floor.
John Lipor	160 - 11	1st Floor	This office is in the 1st floor.
James Mcnames	160 - 04	1st Floor	This office is in the 1st floor.
Branimir Pejcinovic	160 - 08	1st Floor	This office is in the 1st floor.
Mark Perkowski	160 - 05	1st Floor	This office is in the 1st floor.
Tom Schubert	20 - 10	Basement	Turn around make a right, take

			the first left walk straight for 158 feet and then make a left. Then walk for 40 feet then make a right. Walk straight for 46 feet and the office will be on your left.
T. Martin Siderius	160 - 14	1st Floor	This office is in the 1st floor. Professor office is located on the first floor, Please check the map for directions.
Xiaoyu Song	160 - 15	1st Floor	This office is in the 1st floor.
Christof Teuscher	160 - 07	1st Floor	This office is in the 1st floor.
Richard Tymerski	Email		
Renjeng Su	Email		
John M. Acken	20 - 03	Basement	Turn around make a right, take the first left walk straight for 158 feet and then make a left. Walk straight and the office will be the first one on your left.
Daniel Rouseff	20 - 07	Basement	Turn around make a right, take the first left walk straight for 158 feet and then make a left. Then walk for 40 feet then make a right. Walk for about 15 feet and the office will be on your left.
Glenn Shirley	20 - 03	Basement	Turn around make a right, take the first left walk straight for 158 feet and then make a left. Walk straight and the office will be the first one on your left.
Ivan Sutherland	105	1st Floor	This office is in the 1st floor.

Erin Wan	20 - 16	Basement	Turn around make a right, take the first left walk straight for 158 feet and then make a left. Walk straight for 40 feet and make a right. Walk straight until the end of the hall and the office will be in front of you.
James Morris	20 - 06	Basement	Turn around make a right, take the first left walk straight for 158 feet and then make a left. Walk straight for 40 feet and the office will be in front of you.
Lisa Zurk	20 - 06	Basement	Turn around make a right, take the first left walk straight for 158 feet and then make a left. Walk straight for 40 feet and the office will be in front of you.

Adjunct Faculty:

Name	Location	Floor
Andrew Greenberg	85 - 03	Basement
Tareque Ahmad	85 - 03	Basement
Zeshan Chishti	85 - 03	Basement
Brian Cruikshank	85 - 03	Basement
John Gebbie	85 - 03	Basement
Jessica Grafeen	85 - 03	Basement
Jasur Hanbaba	85 - 03	Basement
Yuchen Huang	85 - 03	Basement
Alain Kagi	85 - 03	Basement

Eric Krause	85 - 03	Basement
Herbert Mayer	85 - 03	Basement
Betsy Natter	85 - 03	Basement
Shannon Nelson	85 - 03	Basement
Mukund Pai	85 - 03	Basement
Ataur Patwary	85 - 03	Basement
Alex Pearson	85 - 03	Basement
Luis Perez	85 - 03	Basement
Linda Rankin	85 - 03	Basement
Nir Vaks	85 - 03	Basement
Jaz Veach	85 - 03	Basement
PJ Washiewicz	85 - 03	Basement
Phillip Wong	85 - 03	Basement

Turn around make a right and then make another right walk through the circuit lounge. Open the first door and the office will be in front of you. And just to let you know, this is a shared office by other professors as well.

LED ARRAY

Longruner 8x32 256 Pixels Digital Flexible LED Panel, Built-in WS2812B IC Individually Addressable LED Light with Full Dream Color Lighting DC5V LWS03

A 256 pixel LED array was added to DIM in an attempt to make him more interactive. The ultimate goal of this integration was to be able to output pixel art that reflected the current state or “mood” of dim. Since no documentation exists for this product, each pixel had to be mapped individually to be able to effectively make images.

A script was made in python with a function call to the different images to be produced. However, when this code was added to our main code, it resulted in unbearable screeching coming out the any attached speakers and incorrect pixel illuminations. Only a complete restart of the Pi would stop this issue. Research was done to see how this issue could be mitigated but no resources were found. Code based methods of debugging the issue were also attempted with

no solution found. A guess to the cause of this problem would be that a buffer in the Pi's audio driver might be getting overrun.

Seeing as integration was not possible, the LED matrix was attached to an arduino UNO that runs a static code. The LED matrix now serves the visual purpose of letting the user know that DIM is powered on.

A potential fix for future groups to implement would be to use a second Pi as a slave device that would receive pin based interrupts from the main Pi. Depending on the pin that the interrupt was received from, a given portion of code for controlling the LED matrix could be implemented.

Kinetic Integration

In this project we did not use the Microsoft Kinect Camera to improve DIMM's functionality. However, we still choose to integrate the Kinetic with the Raspberry Pi. Now users can control the Kinetic through a built-in program called glview or through any python2 or python3 script. With the Kinetic working on the Raspberry Pi, future groups could use its enhanced cameras and depth perception to further improve DIMM.

To integrate the Kinect with the Raspberry Pi we used the python Libfreenect library -- a library for accessing the Microsoft Kinect USB camera. The Libfreenect library was cloned using:

git clone git://github.com/OpenKinect/libfreenect.git

From there, the library was installed with the following commands:

1. *cd libfreenect*
2. *mkdir build && cd build*
3. *cmake -L ..*
4. *sudo make install*
5. *sudo ldconfig /usr/local/lib64/*

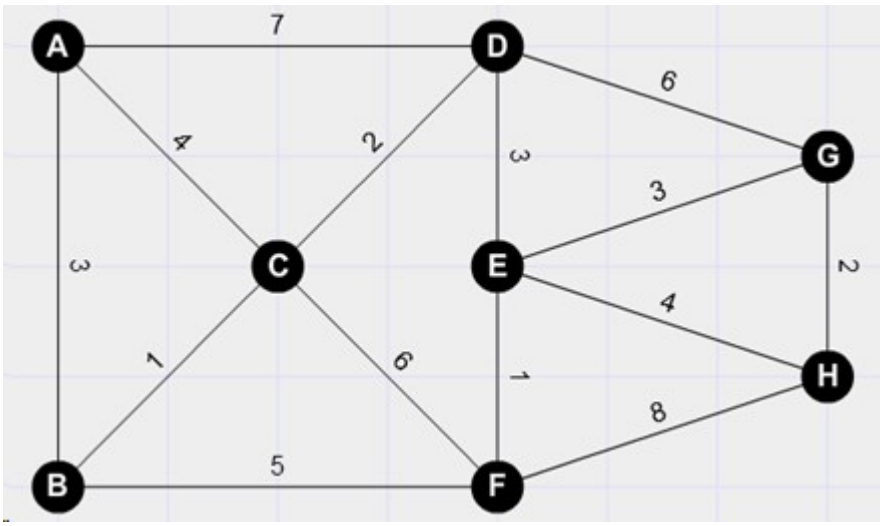
From there, we could experiment with the Kinect using a built-in script installed with the clone called freenect-glview. Running this script opens two windows, one regular real-time image, and a depth image. You can control the angle of the Kinect camera with the 'w' and 'x' keys.

If you want to control the camera with python, you must simply run:
sudo python setup.py install
You also must have all the openCV libraries preinstalled. In your python script, import freenect.

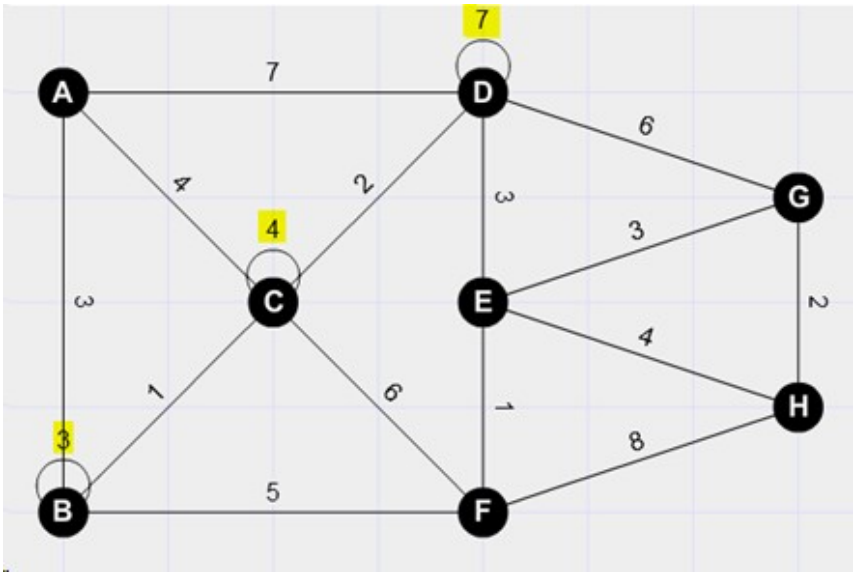
Prolog - Shortest Path & Dijkstra's Algorithm

Prior to implementing the main program which provides audio directions to office location in the FAB basement, an implementation was attempted in prolog. The prolog implementations was based on Dijkstra's Algorithm which is explained in detail below:

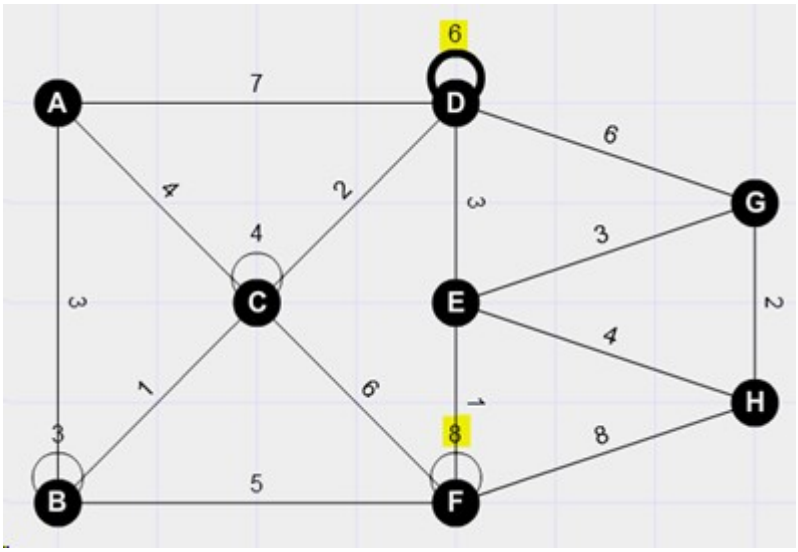
Given the following edge structure with weighted distances we would like to obtain the shortest path from A to H.



From starting edge A, list all distances between the connected edges:

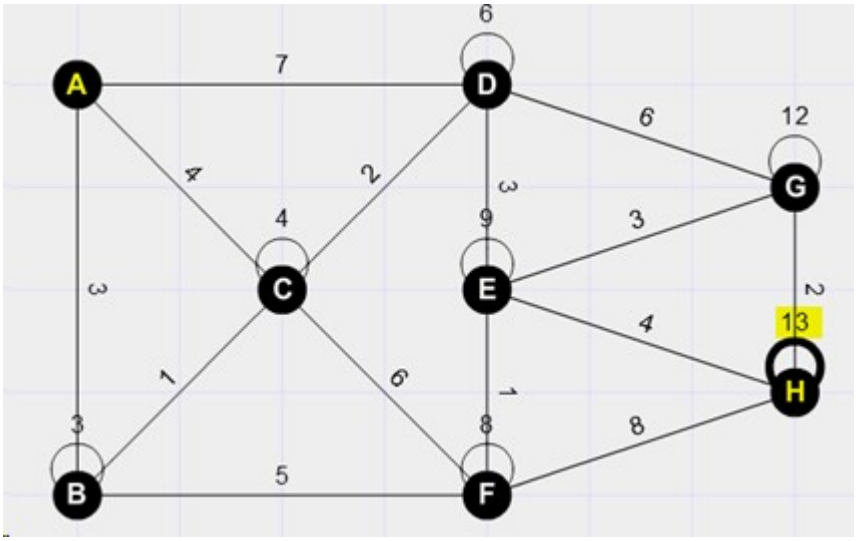


Repeat the process from B to connected edges. In the example of moving from B to C, the total distance would = 3+1. Since this value is the same/not less than the value already present on C, the value of 4 is kept on C. The distance from B to F is then determined to be 3+5 which is added to the F edge. This process is then repeated with the following results:

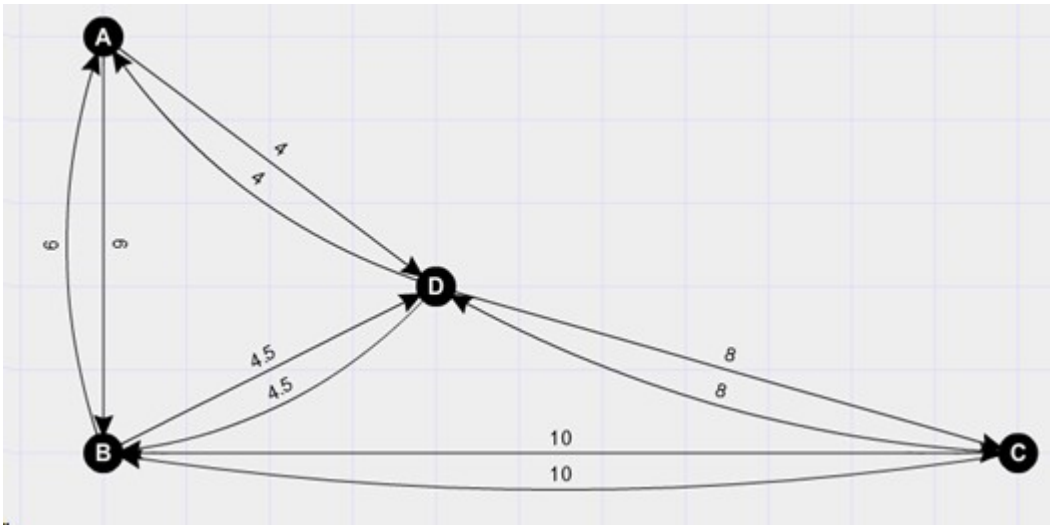


Note: The value of six on edge D replaced the previous value of seven, since the path from A to C to D yields a shorter distance.

If this process is repeated until we arrive at our desired destination H which indicates the shortest distance required to travel from A to H is 13.



Shortest Path: Abstraction of Prolog Implementation



The example above is used to illustrate the hierarchy of predicates for the shortest path prolog program. “**Edges**” which are a natural prolog predicates are shown in the image above. An edge by itself does not yield any useful results, but when connected to other edges, they can be used to determine viable connections that can be traversed.

The predicate in our program called “linked” is used to connect edges and output a length/distance which exists between them. This predicate can be thought of as the foundation that our prolog program uses to determine interconnection between edges.

A predicate called “**walk**” is then used to build on the “linked” predicate by combining connected edges which exist from a start point to a desired end point. This predicate allows movement from the starting edge to intermediate edges, which ultimately lead to the desired destination edge. As it traverses these edges, they are recorded within a list that is then reversed upon completion. This reversal is necessary given that subsequent edges are appended to the list and by the end of the walk predicate the first edge is located at the end of the tail of the list. This predicate is also responsible for adding of total length between linked edges.

The next predicate in this hierarchy is “**path**”. This predicate is responsible for combining the information obtained thorough the “walk” predicate. In a sense this predicate is develop[ping permutations of lists of edges, that are linked together by the “walk” predicate. This predicate is also responsible for ensuring that paths do not result in situations that are infinite, such as walking back and forth from two edges continuously. This is done by comparing existing elements within the list to ensure they are not repeated, and no backtracking occurs.

The final predicate is the called “**shortes**t”. This predicate takes the list of paths obtained in the “path” predicate and selects a path from the list which has the shortest distance.

A final list which has the shortest path is then returned and outputted by the predicate, "**minimal**".

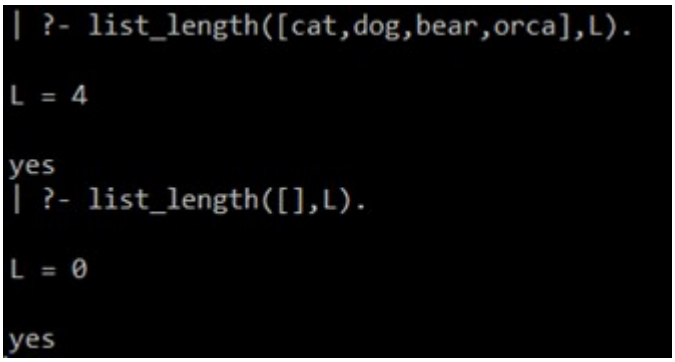
Prolog - Background Knowledge

In order to implement the prolog code structure described above, a number of online tutorials were taken, with each providing information on how to go about the required list manipulations. Summaries of these tutorials are provided for quicker understanding of code functionality.

List Operations - Counting Elements

EX Code: In this code the predicate "list_length" is used to calculate the number of elements in a given list. List_length([],0) is used for the case of an empty list, while the second rule is used for all lsits that are not equal to zero. The variable L represents the total length of the list of elements. The final value of L1 is then added to L to account for the extra element present in the head of the list.

```
list_length([], 0).
list_length([_|Tail],L) :-
                        list_length(Tail,L1),
                        L is L1 +1.
```



List Operations - Concatenation

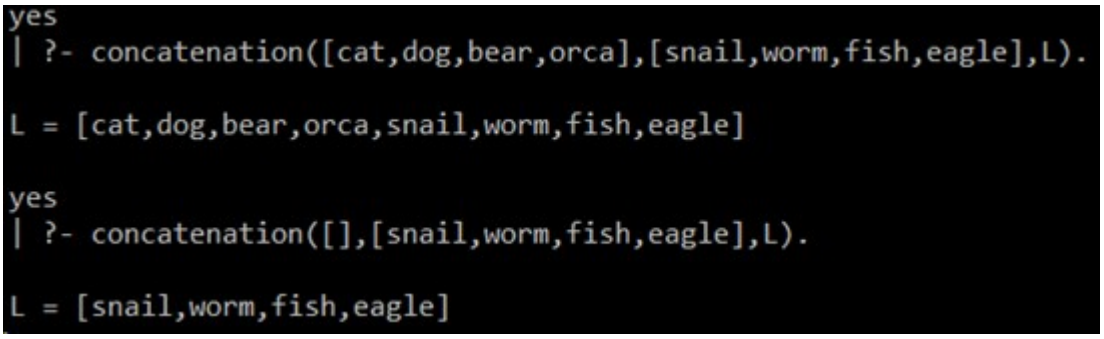
[X|L1] = [X |----------L1----------] [------L2-----]

After concatenation:

[X|L3] = [X | ----------L3----------]

Note X remains as the head of the final list.
Concatenation is the clause name. the first line of code translates to “if we concatenate a an empty list with the list L, then the output list will be L”. The second line translates to “if we concatenate a list with head=X1 and tail=L1 with the list L2, we get a new list with X1=head and L3=tail. So after concatenation of lists L1 and L2, the output will be a list L3. The second query concatenates the non-empty list to the empty list.

```
concatenation([],L,L).
concatenation([X1|L1], L2, [X1|L3]) :- concatenation(L1,L2,L3).
```



List Operations - Removing an Element

When deleting an element from a given list, the element can be in one of two places: the head of the list or the tail of the list. The first clause states “if we want to remove Y from a list where Y

is the only element, then after removing Y, the resulting list will be an empty list". The second clause states "if we want to remove X from a list, where X is the first element of the list, and LIST1 is the tail of the list, then the resulting list LIST1 will only contain the tail elements". The third clause states "if we want to remove X from the list LIST, which has head=Y, then in this case X is in the tail of LIST, and the resulting list will be LIST1 where Y is still the head and X has been removed from the tail.

In the example output shown below, take note of the fourth query. This query differs from the rest in that it determines which item was removed from the second list compared to the first list.

```
remove(Y,[Y], []).
remove(X,[X|LIST1],LIST1).
remove(X,[Y|LIST], [Y|LIST1]) :- remove(X,LIST,LIST1).
```

```
yes
| ?- remove(apple,[apple],X).

X = [] ?

yes
| ?- remove(apple,[apple,pear,grape,peach],X).

X = [pear,grape,peach] ?

yes
| ?- remove(apple,[pear,grape,apple,peach],X).

X = [pear,grape,peach] ?

yes
| ?- remove(X,[apple,grape,pear,peach],[grape,pear,peach]).

X = apple ?
```

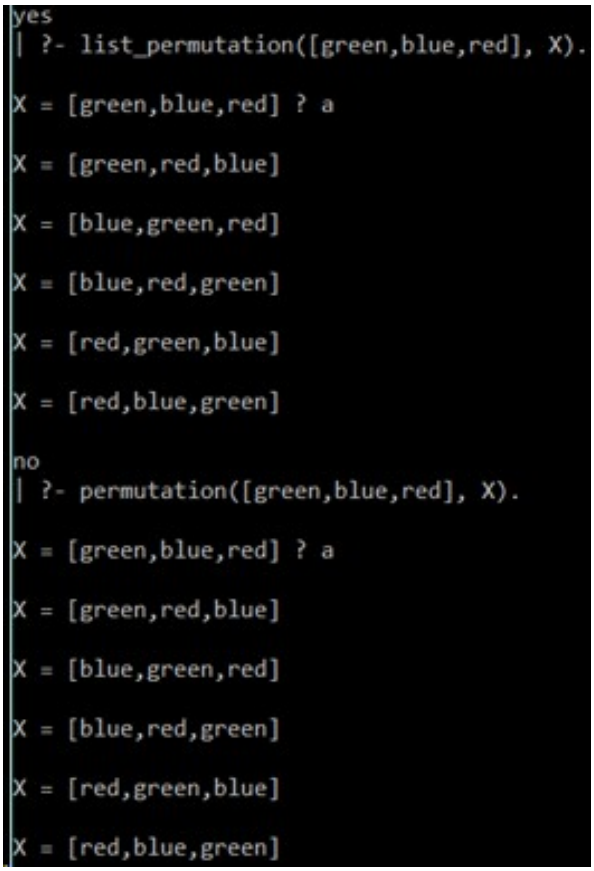
List Operations - Permutations

list_permutation is the name of the predicate that we will use to get all permutations of a given list. List_permutation([],[]) provides for the case of an empty list; the permutation of an empty list is an empty list. The definition that follows states that "we will delete the item x from the list L to

get a new list L1 which does not contain X, then we call list permutation recursively to get the permutations out of L1. The remove function is being used to eliminate combinations which have already occurred.

```
remove(X,[X|LIST1],LIST1).
remove(X,[Y|LIST], [Y|LIST1]) :- remove(X,LIST,LIST1).

list_permutation([],[]).
list_permutation(L,[X|P]) :-remove(X,L,L1), list_permutation(L1,P).
```



List Operations - Append Two Lists

List_memeber(X,[X|_]) states that “X will be a member of the list if X is the head element” Conversely the next line states that “X will be a member of the list if X belongs to the tail of the list.

The line starting with List_append(A,T,T) states that “given an item A, the list T will be appended to A only if and only if A is not a member of the list T. The result is then outputted as the list T”. The next line then is provided as the other option to the line above, of states that “A will remain the head of the list with its tail =TAIL.

Note the “!” symbol. This means that is the line with list_append(A,T,T) :- list_member(A,T) is true, then the following line of code following it will not be evaluated.

```
list_member(X,[X|_]).
list_member(X,[_|TAIL]) :- list_member(X,TAIL).

list_append(A,T,T) :- list_member(A,T), !.
list_append(A,TAIL,[A|TAIL]).
```

```
yes
| ?- list_append(apple, [peach,pear,grape],X).

X = [apple,peach,pear,grape]

yes
| ?- list_append(apple, [peach,pear,apple,grape],X).

X = [peach,pear,apple,grape]

yes
| ?- list_append([peach,pear,grape],apple,X).

X = [[peach,pear,grape]|apple]

yes
| ?- list_append([tomato,orange], [peach,pear,apple,grape],X).

X = [[tomato,orange],peach,pear,apple,grape]
```