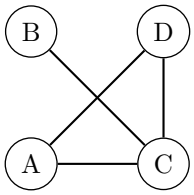
CAB203
Tutorial 7

1 Graphs

1. For each of the following graphs, determine the neighbourhoods and degrees of each vertex. Verify that the handshaking lemma holds for each graph.

   (a)



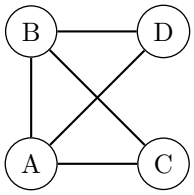
Solution:

$$\begin{aligned} N_A &= \{C, D\} \\ N_B &= \{C\} \\ N_C &= \{A, B, D\} \\ N_D &= \{A, C\} \end{aligned}$$

$$\begin{aligned} d(A) &= 2 \\ d(B) &= 1 \\ d(C) &= 3 \\ d(D) &= 2 \end{aligned}$$

There are two (an even number) of vertices of odd degree (B and C.)

   (b)



Solution:

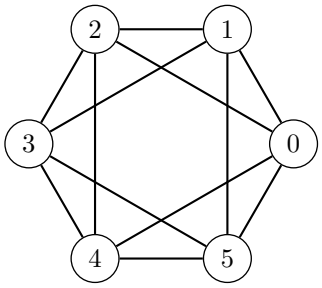
$$\begin{aligned} N_A &= \{B, C, D\} \\ N_B &= \{A, C, D\} \\ N_C &= \{A, B\} \\ N_D &= \{A, B\} \end{aligned}$$

$$\begin{aligned} d(A) &= 3 \\ d(B) &= 3 \\ d(C) &= 2 \\ d(D) &= 2 \end{aligned}$$

There are two (an even number) of vertices of odd degree (A and B.)

2. Draw the graph on the vertices $\{0, 1, 2, 3, 4, 5\}$ where u is adjacent to v if $u \equiv v + 1 \pmod 6$ or $u \equiv v + 2 \pmod 6$.

Solution:



3. For each of the following graphs and answer these questions:

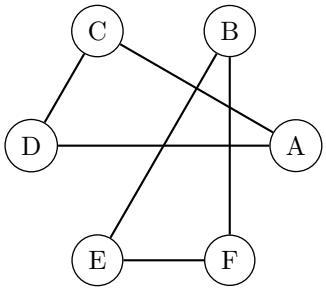
   i. Is A, B, C a path?

   ii. Is A, B, C, D, A a cycle?

   iii. What is the length of the longest path?

   iv. What are the connected components?

   (a)



Solution:

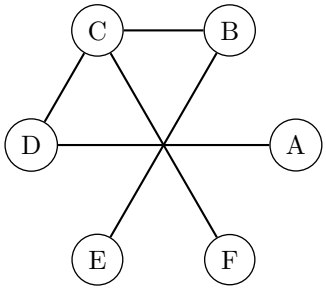
i. A, B, C is not a path.

ii. A, B, C, D, A is not a cycle.

iii. The longest path has length 2. There are several, such as A, C, D.

iv. The connected components are $\{A, C, D\}$ and $\{B, E, F\}$.

(b)



Solution:

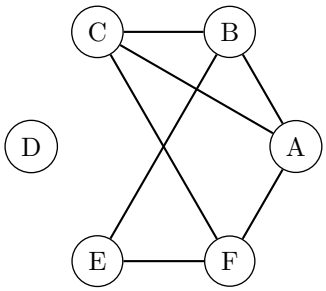
i. A, B, C is not a path.

ii. A, B, C, D, A is not a cycle.

iii. The longest path has length 4. There are several, such as A, D, C, B, E.

iv. The single connected component is $\{A, B, C, D, E, F\}$.

(c)



Solution:

i. A, B, C is a path.

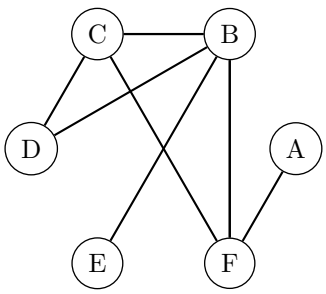
ii. A, B, C, D, A is not a cycle.

iii. The longest path has length 4. There are several, such as A, B, C, F, E.

iv. The connected components are $\{A, B, C, E, F\}$ and $\{D\}$.

4. Let $D(u, v)$ be the distance between vertices u and v. For the following graphs, form find the distance classes from vertex A, $D_j = \{v \in V : D(A, v) = j\}$.

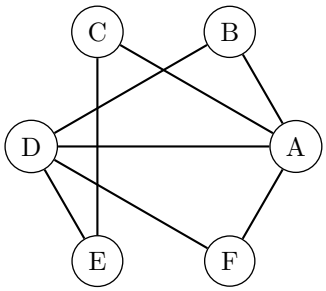
(a)



Solution:

$$\begin{aligned} G_0 &= \{A\} \\ G_1 &= \{F\} \\ G_2 &= \{B, C\} \\ G_3 &= \{D, E\} \end{aligned}$$

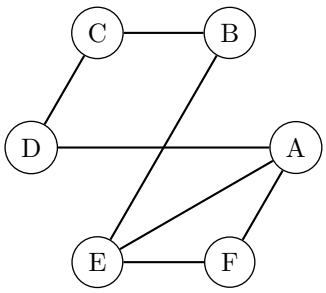
(b)



Solution:

$$\begin{aligned} G_0 &= \{A\} \\ G_1 &= \{B, C, D, F\} \\ G_2 &= \{E\} \end{aligned}$$

(c)



Solution:

$$\begin{aligned} G_0 &= \{A\} \\ G_1 &= \{D, E, F\} \\ G_2 &= \{C, B\} \end{aligned}$$

5. For each problem, describe a graph and a graph theoretic problem that describes the original problem. Describe an approach to solving the graph theoretic problem.

   (a) A travel company wants to develop a website that allows users to select an airport and lists the minimum number of stops-overs required to fly to all other airports.

Solution: The graph has airports as vertices, and two airports are adjacent if there is a direct flight between the airports. The problem is to find the minimum path length from the selected airport to all other airports. The number of stopovers for a flight is equal to the length of a path minus 1. This problem could be solved by using a breadth first traversal type algorithm starting at the selected airport.

(b) A robot is being programmed to solve a maze. The maze consists of several intersections with corridors between them, plus an entrance and an exit. The robot is able to follow corridors and remember intersections and corridors in the maze that it has visited. The goal of the robot is to travel from the entrance to the exit.

Solution: The vertices of the graph are the entrance, exit, and all intersections. Two vertices are adjacent if there is a corridor connecting them (directly without intersecting any other corridors). The problem is to traverse the graph until the exit vertex is found. Since the robot can only move easily between adjacent intersections, a depth first traversal is called for, starting at the entrance.

(c) A student is studying Rubik's cubes. She wants to write a program that, given a starting configuration of the cube, determines the minimum number of moves required to solve the cube, or determines that it is impossible (perhaps someone has moved the stickers!)

Solution: The vertices are the different configurations of the Rubik's cube. Two vertices are adjacent if there is a single move that takes one configuration to the other. Determining the minimum number of moves corresponds to finding a shortest path from the starting configuration to the winning configuration. This could be done using a breadth first search starting at the initial configuration.

6. Modify the `distanceClasses` function from the lecture to return the distance between two points rather than returning the distance classes from a particular vertex.

Solution: We first need to take a second vertex as a parameter in both functions. We'll use `v`. We just need to check whether we've found `v` and return the current distance rather than the distance classes.

This code is not very efficient. In particular, we don't need to remember all the distance classes, only the most recent one. However, this suffices.

One special case that is required is if `v` is not reachable from `u`. To indicate this we return -1.

```
def distance(V, E, u, v):
   V0 = V
   D = [ {u} ]
   return distanceR(V0, E, D, v)

def distanceR(V, E, D, v):
   Vnew = V - D[-1]
   if len(Vnew) == 0:
      return -1
   Dnew = D + [ NS(Vnew, E, D[-1]) ]
   if v in Dnew[-1]:
      return len(Dnew) - 1
   return distanceR(Vnew, E, Dnew, v)
```