Nama : Elviyani Mawarni

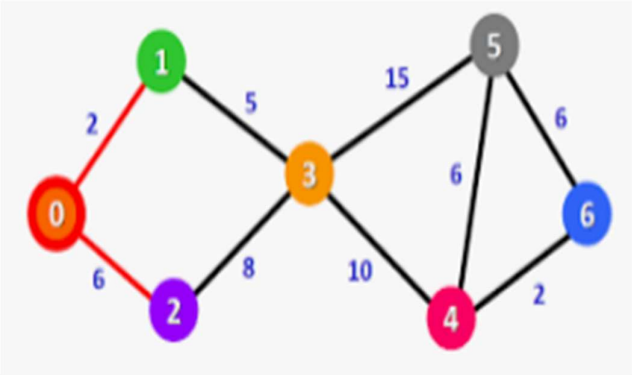
NIM : G.231.22.0077

Mata Kuliah : Struktur Data

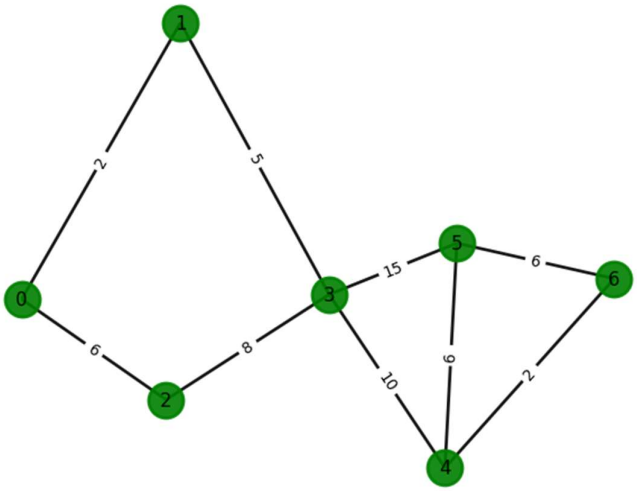
PRAK 8. UAS PRAKTIKUM

ALGORITMA C. DIJKSTRA SHORTEST PATH 2

1. Soal :



2. Hasil Visualisasi coding



3. Algoritma Dijkstra shortest 0-6
   - Membuat table untuk menentukan lintasan / jalur terpendek.

V	'0'	'1'	'2'	'3'	'4'	'5'	'6'

- Melihat node (v) 0 terhubung ke 2 node : 1 dan 2 kemudian mengisikan di kolom ‘0’ = 0, ‘1’ = 2, ‘2’=6.

V	‘0’	‘1’	‘2’	‘3’	‘4’	‘5’	‘6’
‘0’	0	2 (0-1)	6 (0-2)	∞	∞	∞	∞

- Antara node ‘1’ dan ‘2’ lebih pendek ‘1’ maka akan ditulis ‘1’ dibawahnya. Dan diisi dengan jarak 0-1 = 2, 0-2 = 6, dan 0-1-3 = 7.

V	‘0’	‘1’	‘2’	‘3’	‘4’	‘5’	‘6’
‘0’	0	2 (0-1)	6 (0-2)	∞	∞	∞	∞
‘1’	0	2 (0-1)	6 (0-2)	7 (0-1-3)	∞	∞	∞

- Kemudian dituliskan node 3 dan menuliskan jarak lintasannya seperti table dibawah ini :

V	‘0’	‘1’	‘2’	‘3’	‘4’	‘5’	‘6’
‘0’	0	2 (0-1)	6 (0-2)	∞	∞	∞	∞
‘1’	0	2 (0-1)	6 (0-2)	7 (0-1-3)	∞	∞	∞
‘3’	0	2 (0-1)	6 (0-2)	7 (0-1-3)	17 (0-1-3-4)	22 (0-1-3-5)	∞

- Kemudian lihat table diatas, node 4 dan node 5 lebih kecil nilai di node 4 maka node 4 akan dituliskan dibawahnya. Lalu tuliskan jarak node seperti table dibawah ini :

V	‘0’	‘1’	‘2’	‘3’	‘4’	‘5’	‘6’
‘0’	0	2 (0-1)	6 (0-2)	∞	∞	∞	∞
‘1’	0	2 (0-1)	6 (0-2)	7 (0-1-3)	∞	∞	∞
‘3’	0	2 (0-1)	6 (0-2)	7 (0-1-3)	17 (0-1-3-4)	22 (0-1-3-5)	∞
‘4’	0	2 (0-1)	6 (0-2)	7 (0-1-3)	17 (0-1-3-4)	23 (0-1-3-4-5)	19 (0-1-3-4-6)

- Dari table diatas dapat diambil kesimpulan Jarak terpendek lintasan dari 0-6 yaitu melewati lintasan : 0-1-3-4-6 dengan panjang lintasan 19.

4. Coding dan Hasil Coding

```
#membuat fungsi untuk mendapatkan panjang lintasan
def get_path_weight(path):
  #inisialisasi awal 0
  path_weight = 0
  # path akan di looping digunakan untuk mencari berapa beban
dari lintasan yang nantinya akan ditambahkan pada path_weight
  for index, value in enumerate(path):
    try:
      for j in graph[value]:
        if j['v'] == path[index + 1]:
            path_weight += j['w']
    except:
      break

  return path_weight

#membuat fungsi untuk mencari jarak terpendek.
def findShortpath(graph, start, end, path =[]):
  path = path + [start]
  shortest = None
  weights = None

  if start == end: return path

#mencari relasi dari node.
  for node in graph[start]:
      if node['v'] not in path:
        #menampung semua kemungkinan lintasan
          newpath = findShortpath(graph, node['v'], end, path)
          #jika ada lintasan baru akan disimpan di new weight
          if newpath:
            new_weight = get_path_weight(newpath)

            #jika tdk ada weight / new weight lebiH kecil dari
weights maka lintasan terpendek (shortest) akan diganti ke
newpath dan beban (weights) akan diganti ke new_weight.
            if not weights or new_weight < weights:
              shortest = newpath
              weights = new_weight
  return shortest

#memanggil fungsi findShortpath dari titik 0 ke 6 dan mencari
panjang lintasan.
lintasan_terpendek = findShortpath(graph, '0', '6')
```

```
panjang_lintasan = get_path_weight(lintasan_terpendek)

print('Lintasan Terpendek :', lintasan_terpendek )
print('Panjang lintasan :', panjang_lintasan)
```

Hasil :

```
Lintasan Terpendek : ['0', '1', '3', '4', '6']
Panjang lintasan : 19
```