Report

January 6, 2020

0.1 Group members:

- Do Duy Huy Hoang
- Pham Thi Ngoc Mai
- Doan Lien Huong
- Le Nhu Chu Hiep
- Pham Phan Bach
- Vu Khanh Huyen

[76]:
```
# Enable auto-reloading modules
%reload_ext autoreload
%autoreload 2
```

[77]:
```
# Import necessary packages
from graph import *
from tree import *
from pptree import print_tree
```

0.2 1. Overview

- Our code is organized in two files:
  - `graph.py`: definition of the class Graph along with graph-related functions
  - `tree.py`: definition of the class Tree along with tree traversal functions
- Implemented functions:
  1. `degree`
  2. `neighbors`
  3. `components`
  4. `path`
  5. `spanning_tree`
  6. `prim`
  7. `kruskal`
  8. `dijkstra`
  9. `shortest_path`
  10. `preorder`
  11. `postorder`
- We also borrowed some code online:
  - `priority_dict.py`: an implementation of priority queues, which is used for Prim, Kruskal and Dijkstra
  - `unionfind.py`: an implementation of disjoint sets, which is used for Kruskal

0.3 2. Examples

- Graphs are represented using an adjacency matrices
- We used the provided graph of Vietnamese cities. There are 98 vertices, each is assigned with a numeric unique ID

[78]:
```
G = Graph("graphs/Vietnam_Distances.txt")
print(G.vert_name_2_id)
```

```
{'Ho Chi Minh City': 0, ' Hanoi': 1, 'Da Nang': 2, ' Haiphong': 3, 'Bien Hoa':
4, ' Hue': 5, 'Nha Trang': 6, ' Can Tho': 7, 'Rach Gia': 8, ' Qui Nhon': 9, 'Da
Lat': 10, ' Thanh Pho Nam Dinh': 11, 'Vinh': 12, ' Phan Thiet': 13, 'Long
Xuyen': 14, ' Thanh Pho Ha Long': 15, 'Buon Ma Thuot': 16, ' Thanh Pho Thai
Nguyen': 17, 'My Tho': 18, ' Soc Trang': 19, 'Pleiku': 20, ' Thanh Hoa': 21, 'Ca
Mau': 22, ' Thanh pho Bac Lieu': 23, 'Thanh Pho Hoa Binh': 24, ' Vinh Long': 25,
'Yen Bai': 26, ' Viet Tri': 27, 'Phan Rang-Thap Cham': 28, ' Thu Dau Mot': 29,
'Tuy Hoa': 30, ' Tan An': 31, 'Cao Lanh': 32, ' Ben Tre': 33, 'Tam Ky': 34, '
Thanh Pho Hai Duong': 35, 'Tra Vinh': 36, ' Thanh Pho Lang Son': 37, 'Bac
Giang': 38, ' Thanh Pho Thai Binh': 39, 'Kon Tum': 40, ' Bac Ninh': 41, 'Thanh
Pho Cao Bang': 42, ' Dien Bien Phu': 43, 'Hung Yen': 44, ' Thanh Pho Ninh Binh':
45, 'Lao Cai': 46, ' Tay Ninh': 47, 'Thanh Pho Tuyen Quang': 48, ' Quang Ngai':
49, 'Thanh Pho Ha Giang': 50, ' Thanh Pho Phu Ly': 51, 'Quang Binh': 52, ' Ha
Tinh': 53, 'Vi Thanh': 54, ' Don Luan': 55, 'Son La': 56, ' Vinh Yen': 57, 'Bac
Kan': 58, ' Dong Ha': 59, ' Ho Chi Minh City': 60, 'Haiphong': 61, 'Hue': 62,
'Can Tho': 63, 'Qui Nhon': 64, 'Thanh Pho Nam Dinh': 65, 'Phan Thiet': 66,
'Thanh Pho Ha Long': 67, 'Thanh Pho Thai Nguyen': 68, 'Soc Trang': 69, 'Thanh
Hoa': 70, 'Thanh pho Bac Lieu': 71, 'Vinh Long': 72, 'Viet Tri': 73, 'Thu Dau
Mot': 74, 'Tan An': 75, 'Ben Tre': 76, 'Thanh Pho Hai Duong': 77, 'Thanh Pho
Lang Son': 78, 'Thanh Pho Thai Binh': 79, 'Bac Ninh': 80, 'Dien Bien Phu': 81,
'Thanh Pho Ninh Binh': 82, 'Tay Ninh': 83, 'Quang Ngai': 84, 'Thanh Pho Phu Ly':
85, 'Ha Tinh': 86, 'Don Luan': 87, 'Vinh Yen': 88, 'Dong Ha': 89, 'Hanoi': 90,
'Cat Ba': 91, 'Can Giuoc': 92, 'Cam Ranh': 93, 'Cho Doc': 94, ' Yen Bai': 95, '
Da Lat': 96, ' Tuy Hoa': 97}
```

0.3.1 degree(G, v): returns the degree of v in G

[79]:
```
hcmc_id = G.vert_name_2_id['Ho Chi Minh City']
hp_id = G.vert_name_2_id['Da Lat']
print(degree(G, hcmc_id))
```

```
58
```

0.3.2 neighbors(G, v): returns the list of neighbors of v in graph G

[80]:
```
print([G.vert_id_2_name[u] for u in neighbors(G, hcmc_id)])
```

```
[' Hanoi', 'Bien Hoa', 'Nha Trang', 'Rach Gia', 'Da Lat', 'Vinh', 'Long Xuyen',
'Buon Ma Thuot', 'My Tho', 'Pleiku', 'Ca Mau', 'Thanh Pho Hoa Binh', 'Yen Bai',
'Phan Rang-Thap Cham', 'Tuy Hoa', 'Cao Lanh', 'Tam Ky', 'Tra Vinh', 'Bac Giang',
'Kon Tum', 'Thanh Pho Cao Bang', 'Hung Yen', 'Lao Cai', 'Thanh Pho Tuyen Quang',
'Thanh Pho Ha Giang', 'Quang Binh', 'Vi Thanh', 'Son La', 'Bac Kan', 'Haiphong',
'Hue', 'Can Tho', 'Qui Nhon', 'Thanh Pho Nam Dinh', 'Phan Thiet', 'Thanh Pho Ha
Long', 'Thanh Pho Thai Nguyen', 'Soc Trang', 'Thanh Hoa', 'Thanh pho Bac Lieu',
'Vinh Long', 'Viet Tri', 'Thu Dau Mot', 'Tan An', 'Ben Tre', 'Thanh Pho Hai
Duong', 'Thanh Pho Lang Son', 'Thanh Pho Thai Binh', 'Bac Ninh', 'Dien Bien
Phu', 'Thanh Pho Ninh Binh', 'Tay Ninh', 'Quang Ngai', 'Thanh Pho Phu Ly', 'Ha
Tinh', 'Don Luan', 'Vinh Yen', 'Dong Ha']
```

0.3.3 components(G): returns the list of connected components in G. Uses DFS to locate components

The graph of Vietnamese cities is connected, so it has only 1 component

[81]:
```python
component_list = components(G)
print(component_list)
print("\n")
print("Number of components:", len(component_list))
```

```
[{0: None, 1: 0, 4: 0, 5: 4, 20: 5, 11: 20, 10: 11, 36: 10, 30: 36, 31: 30, 75:
30, 77: 30, 81: 30, 84: 30, 22: 84, 23: 22, 86: 22, 50: 86, 51: 50, 95: 50, 44:
95, 45: 44, 48: 95, 49: 48, 62: 48, 26: 62, 27: 26, 58: 62, 59: 58, 63: 62, 93:
62, 2: 93, 3: 2, 60: 2, 8: 93, 9: 8, 52: 93, 53: 52, 74: 95, 71: 86, 79: 86, 76:
79, 82: 79, 96: 82, 67: 96, 91: 96, 90: 79, 28: 90, 29: 28, 68: 90, 72: 90, 83:
90, 92: 86, 32: 84, 33: 32, 94: 30, 37: 36, 40: 10, 41: 40, 64: 10, 21: 20, 97:
20, 6: 0, 7: 6, 12: 0, 13: 12, 14: 0, 15: 14, 16: 0, 17: 16, 18: 0, 19: 18, 24:
0, 25: 24, 34: 0, 35: 34, 38: 0, 39: 38, 42: 0, 43: 42, 46: 0, 47: 46, 54: 0,
55: 54, 56: 0, 57: 56, 61: 0, 65: 0, 66: 0, 69: 0, 70: 0, 73: 0, 78: 0, 80: 0,
85: 0, 87: 0, 88: 0, 89: 0}]


Number of components: 1
```

0.3.4 path(G, u, v): returns the path between u and v if it exists. Also uses DFS

[82]:
```python
print([G.vert_id_2_name[u] for u in path(G, hcmc_id, hp_id)])
```

```
['Ho Chi Minh City', 'Bien Hoa', ' Hue', 'Pleiku', ' Thanh Pho Nam Dinh', 'Da
Lat']
```

0.3.5 spanning_tree(G, v): returns an arbitrary spanning tree rooted at v, using DFS

[83]:
```python
st = spanning_tree(G, hcmc_id)
print_tree(st)
print()
print("Preorder traversal of st: ", end="")
preorder(st)
```

Hanoi
Haiphong
Thanh Pho Nam Dinh
Phan Thiet
Soc Trang
Thanh Hoa
Viet Tri
Thanh Pho Lang Son
Bac Ninh
Thanh Pho Phu Ly
Don Luan
Vinh Yen
Dong Ha
Nha Trang
Can Tho
Vinh
Phan Thiet
Long Xuyen
Thanh Pho Ha Long
Buon Ma Thuot
Thanh Pho Thai Nguyen
My Tho
Soc Trang
Thanh Pho Hoa Binh
Vinh Long
Tam Ky
Thanh Pho Hai Duong
Bac Giang
Thanh Pho Thai Binh
Thanh Pho Cao Bang
Dien Bien Phu
Lao Cai
Tay Ninh
Vi Thanh
Don Luan
Son La
Vinh Yen
Ho Chi Minh City
Bien Hoa
Hue
Thanh Hoa
Tuy Hoa
Pleiku
Thanh Pho Nam Dinh
Qui Nhon
Kon Tum
Bac
Ninh

Da Lat

Thanh Pho Lang Son

Tra Vinh

Tan An

Tan An

Thanh Pho Hai Duong

Dien Bien Phu

Cho Doc

Tuy

Hoa

Cao Lanh

Ben Tre

Quang Ngai

Thanh pho Bac Lieu

Ca Mau

Thanh pho Bac Lieu

Can Giuoc

Ben Tre

Thanh Pho Ninh Binh

Thanh Pho Ha Long

Da Lat

Cat

Ba

Thanh Pho Thai Binh

Thanh Pho Thai Nguyen

Vinh Long

Hanoi

Phan Rang-Thap Cham

Thu

Dau Mot

Tay Ninh

Ha Tinh

Thanh Pho Phu Ly

Thanh Pho Ha Giang

Thu Dau Mot

Hung Yen

Thanh Pho

Ninh Binh

Yen Bai

Quang Ngai

```
                                                          Thanh Pho Tuyen Quang

 Can Tho

 Yen Bai

         Viet Tri

 Bac Kan

         Dong Ha
 Hue
           Rach Gia
                     Qui Nhon
   Cam Ranh
                     Haiphong
           Da Nang
                     Ho Chi Minh City
           Quang Binh
                        Ha Tinh

Preorder traversal of st: Ho Chi Minh City;  Hanoi; Bien Hoa;  Hue; Pleiku;
Thanh Pho Nam Dinh; Da Lat; Tra Vinh; Tuy Hoa;  Tan An; Tan An; Thanh Pho Hai
Duong; Dien Bien Phu; Quang Ngai; Ca Mau;  Thanh pho Bac Lieu; Ha Tinh; Thanh
Pho Ha Giang;  Thanh Pho Phu Ly;  Yen Bai; Hung Yen;  Thanh Pho Ninh Binh; Thanh
Pho Tuyen Quang;  Quang Ngai; Hue; Yen Bai;  Viet Tri; Bac Kan;  Dong Ha; Can
Tho; Cam Ranh; Da Nang;  Haiphong;  Ho Chi Minh City; Rach Gia;  Qui Nhon; Quang
Binh;  Ha Tinh; Thu Dau Mot; Thanh pho Bac Lieu; Thanh Pho Thai Binh; Ben Tre;
Thanh Pho Ninh Binh;  Da Lat; Thanh Pho Ha Long; Cat Ba; Hanoi; Phan Rang-Thap
Cham;  Thu Dau Mot; Thanh Pho Thai Nguyen; Vinh Long; Tay Ninh; Can Giuoc; Cao
Lanh;  Ben Tre; Cho Doc;  Thanh Pho Lang Son; Kon Tum;  Bac Ninh; Qui Nhon;
Thanh Hoa;  Tuy Hoa; Nha Trang;  Can Tho; Vinh;  Phan Thiet; Long Xuyen;  Thanh
Pho Ha Long; Buon Ma Thuot;  Thanh Pho Thai Nguyen; My Tho;  Soc Trang; Thanh
Pho Hoa Binh;  Vinh Long; Tam Ky;  Thanh Pho Hai Duong; Bac Giang;  Thanh Pho
Thai Binh; Thanh Pho Cao Bang;  Dien Bien Phu; Lao Cai;  Tay Ninh; Vi Thanh;
Don Luan; Son La;  Vinh Yen; Haiphong; Thanh Pho Nam Dinh; Phan Thiet; Soc
Trang; Thanh Hoa; Viet Tri; Thanh Pho Lang Son; Bac Ninh; Thanh Pho Phu Ly; Don
Luan; Vinh Yen; Dong Ha;
```

0.3.6 prim(G): returns a minimum spanning tree using Prim's algorithm

[84]:
```python
prim_mst, w = prim(G)
print_tree(prim_mst)
print()
print("Postorder traversal of prim_mst: ", end="")
postorder(prim_mst)
print("\n")
print("Total weight: ", w)
```

```
 Hanoi
Bien Hoa
Haiphong
Thanh Pho Nam Dinh
Phan Thiet
Thanh Pho Ha Long
Soc Trang
Thanh Hoa
Thanh pho Bac Lieu
Vinh Long
Viet Tri
Thu Dau Mot
Tan An
Ben Tre
Thanh Pho Hai Duong
Thanh Pho Lang Son
Bac Ninh
Dien Bien Phu
Tay Ninh
Thanh Pho Phu Ly
Don Luan
Vinh Yen
Dong Ha
Nha Trang
           Can Tho
Rach Gia
          Qui Nhon
Vinh
      Phan Thiet
Long Xuyen
            Thanh Pho Ha Long
Buon Ma Thuot
               Thanh Pho Thai Nguyen
My Tho
        Soc Trang
Ca Mau
        Thanh pho Bac Lieu
Thanh Pho Hoa Binh
                    Vinh Long
Phan Rang-Thap Cham
                     Thu Dau Mot
Cao Lanh
          Ben Tre
Tam Ky
        Thanh Pho Hai Duong
```

Tra Vinh
Thanh Pho Lang Son
Ho Chi Minh City
Can Tho
Yen Bai
Viet Tri
Bac Kan
Dong Ha
Da Lat
Tuy Hoa
Cho Doc
Tan An
Quang Ngai
Cam Ranh
Haiphong
Da Nang
Ho Chi Minh City
Hue
Quang Ngai
Thanh Pho Tuyen Quang
Hung Yen
Thanh Pho
Ninh Binh
Yen Bai

Thanh Pho Phu Ly
Thanh Pho Ha Giang

Can Giuoc

Ha Tinh

Thanh Pho Ninh Binh

Cat Ba

Thanh Pho Thai Binh

Hanoi

Thanh Pho Thai Nguyen
Hue
Pleiku
Tuy Hoa
Thanh Hoa
Qui Nhon
Da Lat
Thanh Pho Nam Dinh

```
                  Son La
                          Vinh Yen
                  Vi Thanh
                            Don Luan
                  Quang Binh
                              Ha Tinh
                  Lao Cai
                           Tay Ninh
                  Thanh Pho Cao Bang
                                    Dien Bien Phu
                  Kon Tum
                           Bac Ninh
                  Bac Giang
                              Thanh Pho Thai Binh
```

```
Postorder traversal of prim_mst:  Hanoi; Bien Hoa;  Can Tho; Nha Trang;  Qui
Nhon; Rach Gia; Qui Nhon;  Thanh Pho Nam Dinh; Da Lat;  Phan Thiet; Vinh;  Thanh
Pho Ha Long; Long Xuyen;  Thanh Pho Thai Nguyen; Buon Ma Thuot;  Soc Trang; My
Tho;  Hue;  Thanh Hoa;  Tuy Hoa; Pleiku;  Thanh pho Bac Lieu; Ca Mau;  Vinh
Long; Thanh Pho Hoa Binh;  Thu Dau Mot; Phan Rang-Thap Cham;  Ben Tre; Cao Lanh;
Thanh Pho Hai Duong; Tam Ky;  Thanh Pho Lang Son; Tra Vinh;  Thanh Pho Thai
Binh; Bac Giang;  Bac Ninh; Kon Tum;  Dien Bien Phu; Thanh Pho Cao Bang;  Tay
Ninh; Lao Cai;  Ha Tinh; Quang Binh;  Don Luan; Vi Thanh;  Vinh Yen; Son La;
Haiphong;  Viet Tri; Yen Bai;  Thanh Pho Ninh Binh; Hung Yen; Thanh Pho Ninh
Binh; Thanh Pho Thai Nguyen; Hanoi; Cat Ba; Thanh Pho Thai Binh; Can Giuoc; Ha
Tinh;  Thanh Pho Phu Ly; Thanh Pho Ha Giang;  Yen Bai;  Quang Ngai; Thanh Pho
Tuyen Quang;  Dong Ha; Bac Kan;  Da Lat;  Tan An; Cho Doc; Tuy Hoa;  Haiphong;
Ho Chi Minh City; Da Nang; Cam Ranh; Quang Ngai; Hue; Can Tho; Thanh Pho Nam
Dinh; Phan Thiet; Thanh Pho Ha Long; Soc Trang; Thanh Hoa; Thanh pho Bac Lieu;
Vinh Long; Viet Tri; Thu Dau Mot; Tan An; Ben Tre; Thanh Pho Hai Duong; Thanh
Pho Lang Son; Bac Ninh; Dien Bien Phu; Tay Ninh; Thanh Pho Phu Ly; Don Luan;
Vinh Yen; Dong Ha; Ho Chi Minh City;

Total weight:  51975.0
```

0.3.7 kruskal(G): returns a minimum spanning tree using Kruskal's algorithm

Because it is hard to keep track of the tree's hierachical information when implementing Kruskal, the function only return **the edges belonging to the tree** instead of the actual Tree object

[85]:
```python
mst_edges, total_weight = kruskal(G)
print("Edge list: ", [(G.vert_id_2_name[u], G.vert_id_2_name[v]) for (u, v) in␣
 ↪mst_edges])
print("\n")
print("Total weight: ", total_weight)
```

```
Edge list:  [('Ho Chi Minh City', 'Thu Dau Mot'), ('Ho Chi Minh City', 'Bien
Hoa'), ('Ho Chi Minh City', 'Tan An'), ('Hung Yen', ' Thanh Pho Ninh Binh'),
('Thanh Pho Thai Binh', 'Thanh Pho Ninh Binh'), ('Thanh Pho Tuyen Quang', ' Yen
Bai'), ('Ho Chi Minh City', 'My Tho'), ('Ca Mau', ' Thanh pho Bac Lieu'), ('Yen
Bai', ' Viet Tri'), ('Ho Chi Minh City', 'Ben Tre'), ('Ho Chi Minh City', 'Tay
Ninh'), ('Ho Chi Minh City', 'Don Luan'), ('Cao Lanh', ' Ben Tre'), ('Thanh Pho
Thai Nguyen', 'Hanoi'), ('Bac Giang', ' Thanh Pho Thai Binh'), ('My Tho', ' Soc
Trang'), ('Ho Chi Minh City', 'Vinh Long'), ('Ho Chi Minh City', 'Tra Vinh'),
('Thanh Pho Thai Binh', 'Hanoi'), ('Ho Chi Minh City', 'Cao Lanh'), ('Quang
Binh', ' Ha Tinh'), ('Ho Chi Minh City', 'Can Tho'), ('Ho Chi Minh City', 'Long
Xuyen'), ('Thanh Pho Thai Binh', 'Cat Ba'), ('Ho Chi Minh City', 'Soc Trang'),
('Ho Chi Minh City', 'Phan Thiet'), ('Ho Chi Minh City', 'Vi Thanh'), ('Son La',
' Vinh Yen'), ('Ho Chi Minh City', 'Rach Gia'), ('Thanh Pho Ha Giang', ' Yen
Bai'), ('Ho Chi Minh City', 'Thanh pho Bac Lieu'), ('Hung Yen', ' Yen Bai'),
('Hue', 'Quang Ngai'), ('Pleiku', ' Tuy Hoa'), ('Ho Chi Minh City', 'Da Lat'),
('Ho Chi Minh City', 'Ca Mau'), ('Tuy Hoa', ' Da Lat'), ('Vi Thanh', ' Don
Luan'), ('Ho Chi Minh City', 'Buon Ma Thuot'), ('Phan Rang-Thap Cham', ' Thu Dau
Mot'), ('Ho Chi Minh City', 'Phan Rang-Thap Cham'), ('Thanh Pho Ha Giang', '
Thanh Pho Phu Ly'), ('Tuy Hoa', 'Quang Ngai'), ('Thanh Pho Thai Binh', 'Ha
Tinh'), ('Ho Chi Minh City', 'Nha Trang'), ('Da Lat', 'Qui Nhon'), ('Hue', 'Can
Tho'), ('Thanh Pho Cao Bang', ' Dien Bien Phu'), ('Ho Chi Minh City', 'Pleiku'),
('Ho Chi Minh City', 'Kon Tum'), (' Hue', 'Pleiku'), ('Tuy Hoa', ' Tan An'),
('Nha Trang', ' Can Tho'), ('Quang Ngai', 'Cam Ranh'), ('Da Nang', ' Haiphong'),
('Ho Chi Minh City', 'Tam Ky'), ('Da Nang', 'Cam Ranh'), ('Bac Kan', ' Dong
Ha'), ('Da Nang', ' Ho Chi Minh City'), ('Rach Gia', ' Qui Nhon'), ('Thanh Pho
Ha Giang', 'Ha Tinh'), ('Tam Ky', ' Thanh Pho Hai Duong'), ('Ho Chi Minh City',
'Dong Ha'), ('Pleiku', ' Thanh Hoa'), ('Ho Chi Minh City', 'Quang Binh'), ('Tuy
Hoa', 'Cho Doc'), ('Kon Tum', ' Bac Ninh'), ('Thanh Pho Tuyen Quang', 'Hue'),
('Yen Bai', 'Hue'), ('Bac Kan', 'Hue'), ('Thanh Pho Tuyen Quang', ' Quang
Ngai'), ('Ho Chi Minh City', 'Vinh'), ('Vinh', ' Phan Thiet'), ('Da Lat', '
Thanh Pho Nam Dinh'), ('Ho Chi Minh City', 'Thanh Hoa'), ('Buon Ma Thuot', '
Thanh Pho Thai Nguyen'), ('Ho Chi Minh City', 'Thanh Pho Nam Dinh'), ('Ho Chi
Minh City', 'Thanh Pho Phu Ly'), ('Ho Chi Minh City', 'Haiphong'), ('Ho Chi Minh
City', 'Thanh Pho Hoa Binh'), ('Ho Chi Minh City', 'Thanh Pho Hai Duong'), ('Ho
Chi Minh City', 'Thanh Pho Ha Long'), ('Ho Chi Minh City', ' Hanoi'), ('Ho Chi
Minh City', 'Bac Ninh'), ('Ho Chi Minh City', 'Bac Giang'), ('Ho Chi Minh City',
'Vinh Yen'), ('Ho Chi Minh City', 'Viet Tri'), ('Thanh Pho Hoa Binh', ' Vinh
Long'), ('Long Xuyen', ' Thanh Pho Ha Long'), ('Ho Chi Minh City', 'Son La'),
('Ho Chi Minh City', 'Thanh Pho Lang Son'), ('Ho Chi Minh City', 'Dien Bien
Phu'), ('Lao Cai', ' Tay Ninh'), ('Ha Tinh', 'Can Giuoc'), ('Ho Chi Minh City',
'Thanh Pho Cao Bang'), ('Tra Vinh', ' Thanh Pho Lang Son'), ('Ho Chi Minh City',
'Lao Cai')]


Total weight:  51975.0
```

0.3.8 `dijkstra(G, s)`: returns the shortest distances and the shortest path tree from s to all other nodes

[86]:
```
D, parent = dijkstra(G, hcmc_id)
print("Shortest distances:", D)
print("\n")
print("Shortest path tree:", parent)
```

```
Shortest distances: {0: 0, 74: 18.0, 4: 25.0, 75: 40.0, 18: 59.0, 76: 70.0, 83:
79.0, 87: 84.0, 72: 96.0, 36: 102.0, 32: 116.0, 63: 127.0, 14: 139.0, 19: 154.0,
69: 154.0, 66: 161.0, 54: 172.0, 8: 192.0, 71: 197.0, 33: 201.0, 10: 234.0, 22:
244.0, 16: 256.0, 28: 270.0, 23: 308.0, 6: 321.0, 20: 382.0, 30: 387.0, 40:
420.0, 55: 420.0, 64: 433.0, 62: 487.0, 84: 532.0, 29: 533.0, 34: 565.0, 97:
606.0, 96: 633.0, 5: 644.0, 89: 668.0, 52: 739.0, 7: 766.0, 9: 807.0, 31: 813.0,
93: 834.0, 86: 840.0, 53: 862.0, 12: 879.0, 70: 1002.0, 82: 1051.0, 65: 1070.0,
21: 1071.0, 79: 1071.0, 85: 1084.0, 44: 1094.0, 61: 1117.0, 24: 1120.0, 77:
1125.0, 67: 1127.0, 94: 1135.0, 1: 1137.0, 45: 1138.0, 80: 1154.0, 38: 1163.0,
88: 1171.0, 73: 1175.0, 90: 1175.0, 68: 1201.0, 35: 1203.0, 56: 1203.0, 41:
1207.0, 11: 1208.0, 91: 1220.0, 26: 1226.0, 78: 1226.0, 48: 1232.0, 81: 1236.0,
39: 1256.0, 58: 1262.0, 17: 1275.0, 95: 1288.0, 27: 1293.0, 42: 1317.0, 15:
1327.0, 46: 1327.0, 50: 1346.0, 57: 1378.0, 2: 1410.0, 37: 1427.0, 51: 1617.0,
43: 1680.0, 13: 1778.0, 59: 1869.0, 3: 1967.0, 60: 2018.0, 49: 2067.0, 92:
2131.0, 25: 2297.0, 47: 2590.0}


Shortest path tree: {0: None, 1: 0, 4: 0, 6: 0, 8: 0, 10: 0, 12: 0, 14: 0, 16:
0, 18: 0, 20: 0, 22: 0, 24: 0, 26: 0, 28: 0, 30: 0, 32: 0, 34: 0, 36: 0, 38: 0,
40: 0, 42: 0, 44: 0, 46: 0, 48: 0, 50: 0, 52: 0, 54: 0, 56: 0, 58: 0, 61: 0, 62:
63, 63: 0, 64: 0, 65: 0, 66: 0, 67: 0, 68: 0, 69: 0, 70: 0, 71: 0, 72: 0, 73: 0,
74: 0, 75: 0, 76: 0, 77: 0, 78: 0, 79: 0, 80: 0, 81: 0, 82: 0, 83: 0, 84: 0, 85:
0, 86: 0, 87: 0, 88: 0, 89: 0, 95: 48, 5: 4, 19: 18, 90: 79, 37: 36, 33: 32, 15:
14, 55: 54, 9: 8, 93: 8, 11: 10, 23: 22, 96: 30, 17: 16, 29: 28, 7: 6, 21: 20,
31: 30, 97: 20, 94: 30, 41: 40, 35: 34, 91: 79, 53: 52, 2: 93, 92: 86, 13: 12,
45: 44, 25: 24, 39: 38, 57: 56, 27: 26, 49: 48, 59: 58, 43: 42, 47: 46, 51: 50,
3: 2, 60: 2}
```

0.3.9 `shortest_path(G, vert_name_1, vert_name_2)`: wrapper function of `dijkstra`, returns the shortest path between two vertices and its length. Accept vertices' names as arguments for convenience

[87]:
```
p, length = shortest_path(G, "Hanoi", "Bac Ninh")
print("Shortest path: ", p)
print("Length: ", length)
```

```
Shortest path:  ['Hanoi', 'Thanh Pho Thai Binh', 'Ho Chi Minh City', 'Bac Ninh']
Length:  2329.0
```