**TRƯỜNG ĐẠI HỌC SÀI GÒN**

KHOA CÔNG NGHỆ THÔNG TIN



A blue circle with white text

Description automatically generated

**Lab 01: Search (cá nhân)**

HỌC PHẦN: TRÍ TUỆ NHÂN TẠO NÂNG CAO

**Giảng viên hướng dẫn** : Ts. Đỗ Như Tài

**Sinh viên thực hiện:**

Phạm Văn Nam 3122410251

*Thành phố Hồ Chí Minh - Tháng 09/2025*

Mục lục

[1. Đồ thị có hướng 3](#_Toc209212173)

[2. Một số kết quả cơ bản 4](#_Toc209212174)

[3. Các đường đi quan trọng 4](#_Toc209212175)

[4. Biểu diễn bằng danh sách/ma trận kề 4](#_Toc209212176)

[5. Thao tác duyệt đồ thị 5](#_Toc209212177)

[a) Liệt kê các đỉnh trong đồ thị 5](#_Toc209212178)

[b) Liệt kê các đỉnh trong đồ thị 5](#_Toc209212179)

[c) Cho hai đỉnh A, B. Hỏi hai đỉnh A, B có kề nhau không? 6](#_Toc209212180)

[d) Cho một đỉnh A. Hỏi danh sách các đỉnh kề với A 6](#_Toc209212181)

[6. Đọc và lưu đồ thị 7](#_Toc209212182)

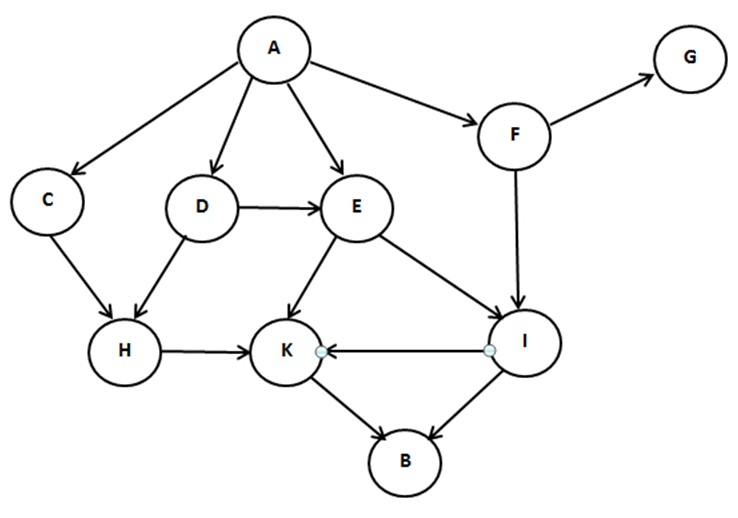
[a) Lưu đồ thị xuống tập tin 7](#_Toc209212183)

[b) Đọc đồ thị từ tập tin 8](#_Toc209212184)

[7. Tìm kiếm đường đi trên đồ thị 9](#_Toc209212185)

[a) Tìm kiếm theo chiều rộng 9](#_Toc209212186)

[b) Tìm kiếm theo chiều rộng (DFS) 10](#_Toc209212187)



# 1. Đồ thị có hướng

Các đỉnh: A, B, C, D, E, F, G, H, I, K

Các cạnh có hướng:

* A → C, A → D, A → E, A → F
* C → H
* D → E, D → H
* E → K, E → I
* F → G, F → I
* H → K
* K → I, K → B
* I → B

# 2. Một số kết quả cơ bản

* Số đỉnh (Vertices): 10
* Số cạnh (Edges): 14
* Đỉnh xuất phát (sources): A (không có cạnh đi vào)
* Đỉnh đích (sinks): G, B (không có cạnh đi ra)
* Chu trình: Có vòng phụ giữa K ↔ I.

# 3. Các đường đi quan trọng

* Từ A → F → G (dẫn đến G – sink)
* Từ A → D → E → K → B (dẫn đến B – sink)
* Nhiều đường đi khác kết hợp qua H, I, K.

# 4. Biểu diễn bằng danh sách/ma trận kề

|  |
| --- |
| ADJ = { "A" : ["C", "D", "E", "F"],  "C" : ["H"],  "D" : ["E", "H"],  "E" : ["I", "K"],  "F" : ["G", "I"],  "H" : ["K"],  "K" : ["B"],  "I" : ["K", "B"],  "B" : [],  "G" : [] } |

|  |
| --- |
| pprint.pprint(ADJ) |
| **Output:**  {'A': ['C', 'D', 'E', 'F'],  'B': [],  'C': ['H'],  'D': ['E', 'H'],  'E': ['I', 'K'],  'F': ['G', 'I'],  'G': [],  'H': ['K'],  'I': ['K', 'B'],  'K': ['B']} |

# 5. Thao tác duyệt đồ thị

## a) Liệt kê các đỉnh trong đồ thị

|  |
| --- |
| print("Các đỉnh của đồ thị: ", end = " ")  list(ADJ.keys()) |
| Output:  Các đỉnh của đồ thị:  ['A', 'C', 'D', 'E', 'F', 'H', 'K', 'I', 'B', 'G'] |

## b) Liệt kê các đỉnh trong đồ thị

|  |
| --- |
| def HienThiDoThi(G):  for i in G:  print("Đỉnh ",i, ": ", G[i])  pass  HienThiDoThi(ADJ) |
| **Output:**  Đỉnh A : ['C', 'D', 'E', 'F']  Đỉnh C : ['H']  Đỉnh D : ['E', 'H']  Đỉnh E : ['I', 'K']  Đỉnh F : ['G', 'I']  Đỉnh H : ['K']  Đỉnh K : ['B']  Đỉnh I : ['K', 'B']  Đỉnh B : []  Đỉnh G : [] |

## c) Cho hai đỉnh A, B. Hỏi hai đỉnh A, B có kề nhau không?

|  |
| --- |
| def LaKe(G, a, b):result = None  if a not in G or b not in G:  return -1  if a not in G[b]:  return 0  if b in G[a] or a in G[b]:  return 1  return result  a = "L"; b = "R"; print(f"{a} ke {b}: {LaKe(ADJ, a, b)}") a = "A"; b = "R"; print(f"{a} ke {b}: {LaKe(ADJ, a, b)}") a = "E"; b = "D"; print(f"{a} ke {b}: {LaKe(ADJ, a, b)}") a = "D"; b = "E"; print(f"{a} ke {b}: {LaKe(ADJ, a, b)}") |
| **Output**:  L ke R: -1  A ke R: -1  E ke D: 1  D ke E: 0 |

## d) Cho một đỉnh A. Hỏi danh sách các đỉnh kề với A

|  |
| --- |
| def LayKe(G, a):result = None  if a in G:  return G[a]  return result  a = "P"; print(f"Danh sach ke voi dinh {a}: {LayKe(ADJ, a)}") a = "A"; print(f"Danh sach ke voi dinh {a}: {LayKe(ADJ, a)}") a = "B"; print(f"Danh sach ke voi dinh {a}: {LayKe(ADJ, a)}") a = "D"; print(f"Danh sach ke voi dinh {a}: {LayKe(ADJ, a)}") |
| **Output**:  Danh sach ke voi dinh P: None  Danh sach ke voi dinh A: ['C', 'D', 'E', 'F']  Danh sach ke voi dinh B: []  Danh sach ke voi dinh D: ['E', 'H'] |

# 6. Đọc và lưu đồ thị

dske.txt

|  |
| --- |
| 10 # số đỉnh đồ thị  A C E F D # đỉnh A kề với C E F D  B # đỉnh B không kề với ai  C H  D E H  E K I  F I G  G  H K  I K B  K B |

## a) Lưu đồ thị xuống tập tin

|  |
| --- |
| def LuuDoThi(G, file\_path, verbose = True):  import os  file\_dir = os.path.dirname(file\_path)  if file\_dir != "" and os.path.exists(file\_path) == False:  os.makedirs(file\_dir)  if verbose: print(f"+ Tao thu muc: {file\_dir}")    with open(file\_path, "wt") as file:  file.write(str(len(G)) + "\n")  for v in G:  neighbors = G[v] if isinstance(G[v], list) else []  line = v + ("" if len(neighbors) == 0 else " " + " ".join(neighbors))  file.write(line + "\n")  if verbose: print(f"Luu do thi thanh cong xuong tap tin: {file\_path}")  pass  LuuDoThi(ADJ, "dske1.txt", verbose = True) with open("dske1.txt", "rt") as file:  lines = file.readlines()  for line in lines: print(line, end="") |
| **Output:**  Luu do thi thanh cong xuong tap tin: dske1.txt  10  A C D E F  C H  D E H  E I K  F G I  H K  K B  I K B  B  G |

## b) Đọc đồ thị từ tập tin

|  |
| --- |
| def DocDoThi(file\_path, verbose = True):  result = None  if os.path.exists(file\_path) == False:  result = None  else:  G = {}  with open(file\_path, "rt") as file:  n = int(file.readline())  for \_ in range(n):  parts = file.readline().strip().split()  if len(parts) > 0:  v = parts[0]  neighbors = parts[1:] if len(parts) > 1 else []  G[v] = neighbors  pass  result = G  return result  G1 = DocDoThi("dske1.txt", verbose = True) pprint.pprint(G1) |
| **Output:**  {'A': ['C', 'D', 'E', 'F'],  'B': [],  'C': ['H'],  'D': ['E', 'H'],  'E': ['I', 'K'],  'F': ['G', 'I'],  'G': [],  'H': ['K'],  'I': ['K', 'B'],  'K': ['B']} |

# 7. Tìm kiếm đường đi trên đồ thị

## a) Tìm kiếm theo chiều rộng

|  |
| --- |
| def BFS(G, start, goal):result = None  if G.get(start) is None or G.get(goal) is None:  result = None  else:  path = {} # path[a]=b nghia la muon di toi dinh b phai qua a  s\_open = []  s\_closed = []  s\_open.append(start)   path[start] = None  while len(s\_open)>0:  u = s\_open.pop(0)  s\_closed.append(u)  if u == goal:  break  for v in G[u]:  if v not in s\_open and v not in s\_closed:  s\_open.append(v)  if v not in path:  path[v] = u  pass  return path  path = BFS(ADJ, "A", "B") pprint.pprint(path) |
| **Output:**  {'A': None,  'B': 'I',  'C': 'A',  'D': 'A',  'E': 'A',  'F': 'A',  'G': 'F',  'H': 'C',  'I': 'E',  'K': 'E'} |

**Find path**

|  |
| --- |
| def find\_path(path, start, goal):result = []  if goal not in path:  return result  v = goal  while v is not None:  result.insert(0, v)  v = path.get(v)  if len(result) == 0 or result[0] != start:  result = []  return result  pass  find\_path(path, 'A', 'B') |
| **Output:**  ['A', 'E', 'I', 'B'] |

## b) Tìm kiếm theo chiều rộng (DFS)

|  |
| --- |
| def DFS(G, start, goal):result = None    if G.get(start) is None or G.get(goal) is None:  result = None  else:  path = {} # path[a]=b nghia la muon di toi dinh b phai qua a  s\_open = []  s\_closed = []  s\_open.append(start)   path[start] = None  while len(s\_open)>0:  u = s\_open.pop()  s\_closed.append(u)  if u == goal:  break  for v in G[u]:  if v not in s\_open and v not in s\_closed:  s\_open.append(v)  if v not in path:  path[v] = u  pass  return path  path = DFS(ADJ, "A", "B") pprint.pprint(path) |
| **Output:**  {'A': None,  'B': 'I',  'C': 'A',  'D': 'A',  'E': 'A',  'F': 'A',  'G': 'F',  'I': 'F',  'K': 'I'} |