|  |  |
| --- | --- |
| Trường Đại học Khoa học Tự Nhiên  **ĐHQG.TPHCM** | **CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM**  **Độc lập - Tự do - Hạnh phúc**  *TPHCM,ngày 24 tháng 10 năm 2024* |

**BÁO CÁO THỰC HÀNH – NHẬP MÔN TRÍ TUỆ NHÂN TẠO – LẦN 1**

**Họ và tên: Nguyễn Thanh Kiên.**

**MSSV: 22110092.**

1. **Nhắc lại Lý thuyết.**

|  |  |
| --- | --- |
| BFS: Breadth First Search là thuật toán duyệt đồ thị ưu tiên chiều rộng để tìm kiếm đường đi ngắn nhất từ một đỉnh gốc tới tất cả các đỉnh khác. | A screenshot of a computer  Description automatically generated |
| DFS: Depth First Search là thuật toán tìm kiếm theo chiều sâu xuất phát từ một đỉnh gốc và đi xa nhất có thể theo một nhánh trước khi quay lui. | A white paper with black text  Description automatically generated |
| UCS: Thuật toán UCS là một thuật toán tìm kiếm trên một cấu trúc cây hoặc đồ thị có trọng số (chi phí). Việc tìm kiếm bắt đầu tại nút gốc và tiếp tục bằng cách duyệt các nút tiếp theo với trọng số hay chi phí thấp nhất tính từ nút gốc. UCS sử dụng một hàng đợi ưu tiên (Priority Queue– PQ) để lưu trữ và duyệt các trạng thái trên đường đi. | A black and white text  Description automatically generated |

1. **Thực hiện bài tập.**
2. **Chạy tay thuật toán BFS, DFS và UCS.**

Cho đồ thị như hình vẽ bên dưới Tìm đường đi ngắn nhất từ trường Đại học Khoa học Tự nhiên (V1) tới sân bay Tân Sơn Nhất (V18) dùng các thuật toán sau: BFS – DFS -UCS.

A diagram of a constellation

Description automatically generated

**. BFS**

1. *L* = [V1] (trạng thái ban đầu).
2. *Node* = V1, *L* = [V2, V3, V4], *father*[V2, V3, V4] = V1.
3. *Node* = V2, *L* = [V3, V4, V7], *father*[V7] = V2.
4. *Node* = V3, *L* = [V4, V7, V6, V8], *father*[V6, V8] = V3.
5. *Node* = V4, *L* = [V7, V6, V8, V5], *father*[V5] = V4.
6. *Node* = V7, *L* = [V6, V8, V5]. (V3 kề với V7 nhưng đã tồn tại trong L nên không thêm vào), *father*[V3] = V7.
7. *Node* = V6, *L* = [V8, V5, V12], *father*[V12] = V6.
8. *Node* = V8, *L* = [V5, V12, V9, V10], *father*[V9, V10] = V8.
9. *Node* = V5, *L* = [V12, V9, V10, V14], *father*[V14] = V5.
10. *Node* = V12, *L* = [V9, V10, V14, V13], *father*[V13] = V12.
11. *Node* = V9, *L* = [V10, V14, V13, V11], *father*[V11] = V9.
12. *Node* = V10, *L* = [V14, V13, V11], *father*[V11] = V9 (không có đường đi từ V10 tới bất kỳ cạnh nào khác, *father* vẫn giữ nguyên như cũ).
13. *Node* = V14, *L* = [V13, V11]. (V11 kề với V14 nhưng đã tồn tại trong L nên không thêm vào), *father*[V11] = V14.
14. *Node* = V13, *L* = [V11]. (V14 kề với V13 nhưng đã tồn tại trong L nên không thêm vào), *father*[V14] = V13.
15. *Node* = V11, *L* = [V15, V10], *father*[V15, V10] = V11.
16. *Node* = V15, *L* = [V10,V16], *father*[V16] = V15.
17. *Node* = V10, *L* = [V16], *father*[V16] = V15 (không có đường đi từ V10 tới bất kỳ cạnh nào khác, *father* vẫn giữ nguyên như cũ).
18. *Node* = V16, *L* = [V17], *father*[V17] = V16.
19. *Node* = V17, *L* = [V18], *father*[V18] = V17.
20. *Node* = V18 (trạng thái kết thúc) -> Dừng.

**Vậy đường đi từ đỉnh V1 tới đỉnh V18 là:**

|  |  |
| --- | --- |
| Đường đi 1: V1 -> V4 -> V5 -> V14 -> V11 -> V15 -> V16 -> V17 -> V18. | Đường đi 1: ĐHKHTN -> Nguyễn Trãi -> CMT8 -> Lý Chính Thắng -> Nam kỳ Khởi nghĩa -> Nguyễn Văn Trỗi -> Phan Đình Giót -> Trường Sơn -> Sân bay TSN. |
| Đường đi 2: V1 -> V3 -> V6 -> V12 -> V13 -> V14 -> V11 -> V15 -> V16 -> V17 -> V18. | Đường đi 2: ĐHKHTN -> Ngã 6 -> Nguyễn Thị Minh Khai -> Cao Thắng -> 3/2 -> Lý Chính Thắng -> Nam kỳ Khởi nghĩa -> Nguyễn Văn Trỗi -> Phan Đình Giót -> Trường Sơn -> Sân bay TSN. |
| Đường đi 3: V1 -> V2 -> V7 -> V3 -> V8 -> V9 -> V11 -> V15 -> V16 -> V17 -> V18. | Đường đi 3: ĐHKHTN -> An Dương Vương -> Trần Phú -> Ngã 6 -> Lý Thái Tổ -> Điện Biên Phủ -> Nam kỳ Khởi nghĩa -> Nguyễn Văn Trỗi -> Phan Đình Giót -> Trường Sơn -> Sân bay TSN. |
| Đường đi 4: V1 -> V2 -> V7 -> V3 -> V6 -> V12 -> V13 -> V14 -> V11 -> V15 -> V16 -> V17 -> V18. | Đường đi 4: ĐHKHTN -> An Dương Vương -> Trần Phú -> Ngã 6 -> Nguyễn Thị Minh Khai -> Cao Thắng -> 3/2 -> Lý Chính Thắng -> Nam kỳ Khởi nghĩa -> Nguyễn Văn Trỗi -> Phan Đình Giót -> Trường Sơn -> Sân bay TSN. |
| Đường đi 5: V1 -> V3 -> V8 -> V9 -> V11 -> V15 -> V16 -> V17 -> V18. | Đường đi 5: ĐHKHTN -> Ngã 6 -> Lý Thái Tổ -> Điện Biên Phủ -> Nam kỳ Khởi nghĩa -> Nguyễn Văn Trỗi -> Phan Đình Giót -> Trường Sơn -> Sân bay TSN. |

**.DFS**

1. *L* = [V1] (trạng thái ban đầu).
2. *Node* = V1, *L* = [V4, V3, V2], *father*[V4, V3, V2] = V1.
3. *Node* = V4, *L* = [V5, V3, V2], *father*[V5] = V4.
4. *Node* = V5, *L* = [V14,V3, V2], *father*[V14] = V5.
5. *Node* = V14, *L* = [V11,V3, V2], *father*[V11] = V14.
6. *Node* = V11, *L* = [V15,V3, V2], *father*[V15] = V11.
7. *Node* = V15, *L* = [V16,V3, V2], *father*[V16] = V15.
8. *Node* = V16, *L* = [V17,V3, V2], *father*[V17] = V16.
9. *Node* = V17, *L* = [V18,V3, V2], *father*[V18] = V17.
10. *Node* = V18 (trạng thái kết thúc) -> Dừng.

**Vậy đường đi từ đỉnh V1 tới đỉnh V18 là:**

V1 -> V4 -> V5 -> V14 -> V11 -> V15 -> V16 -> V17 -> V18 hay ĐHKHTN -> Nguyễn Trãi -> CMT8 -> Lý Chính Thắng -> Nam kỳ Khởi nghĩa -> Nguyễn Văn Trỗi -> Phan Đình Giót -> Trường Sơn -> Sân bay TSN.

**. UCS**

1. *PQ* = {(*V1*, 0)}. (PQ là Priority Queue).
2. *PQ* = {(*V2*, 50), (*V3*, 350), (*V4*, 300)}.
3. *PQ* = {(*V3*, 350), (*V4*, 300), (*V7*, 650)}.
4. *PQ* = {(*V4*, 300), (*V6*, 450), (*V7*, 650), (*V8*, 1250)}.
5. *PQ* = {(*V6*, 450), (*V7*, 650), (*V8*, 1250), (*V5*, 1600)}.
6. *PQ* = {(*V7*, 650), (*V12*, 1150), (*V8*, 1250), (*V5*, 1600)}.
7. *PQ* = {(*V8*, 1250), (*V5*, 1600), (*V12*, 2110)}.
8. *PQ* = {(*V10*, 1550), (*V5*, 1600), {(*V9*, 2040), (*V12*, 2110)}.
9. *PQ* = {(*V5*, 1600), (*V9*, 2040), (*V12*, 2110)}.
10. *PQ* = {(*V9*, 2040), (*V12*, 2110), (*V14*, 3000)}.
11. *PQ* = {(*V12*, 2110), (*V14*, 3000), (*V11*, 3240)}.
12. *PQ* = {(*V14*, 3000), (*V13*, 3060), (*V11*, 3240)}.
13. *PQ* = {(*V14*, 3000), (*V13*, 3060), (*V11*, 3240)}.
14. *PQ* = {(*V13*, 3060), (*V11*, 3240)}.
15. *PQ* = {(*V11*, 3240)}.
16. *PQ* = {(*V15*, 3640)}.
17. *PQ* = {(*V16*, 4940)}.
18. *PQ* = {(*V17*, 5710)}.
19. *PQ* = {(*V18*, 6910)}.

Vậy đường đi ngắn nhất từ V1 tới V18 hay đường đi ngắn nhất từ ĐHKHTN tới Sân Bay TSN là: V1 -> V3 -> V8 -> V9 -> V11 -> V15 -> V16 -> V17 -> V18 hay ĐHKHTN -> Ngã 6 -> Lý Thái Tổ -> Điện Biên Phủ -> Nam kỳ Khởi nghĩa -> Nguyễn Văn Trỗi -> Phan Đình Giót -> Trường Sơn -> Sân bay TSN với chi phí là 6910 (đvtt).

1. **Tính đúng đắn của các thuật toán đã cho sẵn code như trên.**
2. **Tài nguyên:**

Đồ thị vô hướng:

A black background with white dots

Description automatically generated

Tuy vậy, đồ thị từ Trường ĐHKHTN tới Sân bay TSN lại là đồ thị có hướng và trọng số. Vì vậy, ta sẽ dùng đồ thị có hướng và trọng số = 0 để sử dụng cho 2 thuật toán BFS và DFS.

A screenshot of a computer

Description automatically generated

Và đồ thị có hướng, có trọng số để sử dụng cho UCS là:

A screenshot of a computer

Description automatically generated

Dữ liệu txt cho BFS và DFS Dữ liệu txt cho DFS.

A screenshot of a computer code

Description automatically generatedA screenshot of a computer screen

Description automatically generated

1. **Cài đặt và chạy chương trình.**

Sau khi đã cài đặt chương trình giống như trong file mẫu của giáo viên hướng dẫn, chương trình không chạy được và bị báo lỗi ở line 10.

A screenshot of a computer

Description automatically generated

Ở đây, đã có 1 Exception đã được Throw ra: invalid literal for int() with base 10: '\n'

Điều này có nghĩa là, có một dòng trống hoặc dòng có ký tự xuống dòng đã được đọc, và python không thể chuyển đổi nó thành số nguyên. Vì vậy, ta phải sử dụng cấu trúc .strip() để loại bỏ khoảng trắng và kiểm tra dòng trước khi chuyển đổi thành số nguyên

A computer screen shot of a program

Description automatically generated

ở đây:

line = line.strip(): loại bỏ khoảng trắng ở đầu dòng và cuối dòng.

Điều kiện if line: chỉ xử lý dòng nếu không rỗng.

Như vậy ta đã hoàn thành việc đọc data.

Các phần chạy chương trình đều không xuất hiện lỗi.

**Tính đúng đắn của các thuật toán:**

1. BFS là thuật toán tìm kiếm theo chiều rộng, phù hợp với việc tìm các đường đi không đảm bảo ngắn nhất trong đồ thị phi trọng số hay đồ thị không có chi phí. Trong đoạn code, BFS sử dụng 1 hàng đợi Queue cho các Node chờ duyệt vời một từ điển Parent để theo dõi cha của mỗi Node trên đường đi. Tuy nhiên, đoạn mã hiện tại đang thêm các node vào Visited trước khi triểm tra, nên mỗi Node đưa vào hàng đợi nhiều lần và duyệt qua nhiều lần. Vì vậy, ta loại bỏ dòng visited.append(current\_node) trong vòng lặp while và chỉ thêm node vào visited khi duyệt các Node con, giúp tránh trùng lặp và tối ưu bài toán.
2. DFS là thuật toán tìm kiếm theo chiều sâu, cũng không đảm bảo tìm thấy đường đi ngắn nhất như BFS. Trong đoạn code, BFS sử dụng một ngăn xếp frontier cho các Node chờ duyệt và cũng sử dụng từ điển Parent giúp xây dựng đường đi. Giống như BFS, ta có thể loại bỏ dòng visited.append(current\_node) trong vòng lặp While và chỉ thêm vào các node con để tránh trùng lặp và tối ưu bài toán.
3. ÚC là thuật toán tìm kiếm cho đồ thị có trọng số, dùng để tìm kiếm đường đi có chi phí thấp nhất, với việc sử dụng PriorityQueue cho các Node chờ duyệt và parent cùng cost\_so\_far để lưu trữ chi phí thấp nhất cho mỗi node đến thời điểm được duyệt. Visited không cần thiết trong đoạn code này vì cost\_so\_far sẽ đảm bảo chỉ thêm node nếu nó có chi phí thấp hơn chi phí đã ghi nhận.
4. Ta có thể xây dựng hàm để tối ưu code với hàm build\_path giúp tìm kiếm đường đi từ node Start đến Node End dựa vào từ điển parent.

A computer screen with white text

Description automatically generatedA computer screen with white text

Description automatically generated

A computer screen shot of a program code

Description automatically generated

A computer code with white text

Description automatically generated

1. **Kết quả sau khi chạy:**

Sau khi chạy chương trình, kết quả ta thu được như sau:

A screenshot of a computer program

Description automatically generated

Khi sử dụng code và chạy tay, ta thấy kết quả DFS và UCS đều giống nhau. Tuy vậy, đoạn code chỉ in ra 1 đường đi theo thuật toán BFS thay vì 5 đường đi như khi ta chạy tay.

Lí do là vì, khi triển khai thuật toán BFS cho code, ta chỉ tìm kiếm 1 đường đi nông nhất từ đỉnh bắt đầu đến đỉnh kết thúc, bởi vì khi tìm được đường đi đến đỉnh End, thuật toán lập tức thoát khỏi vòng lặp (if current\_node == end:

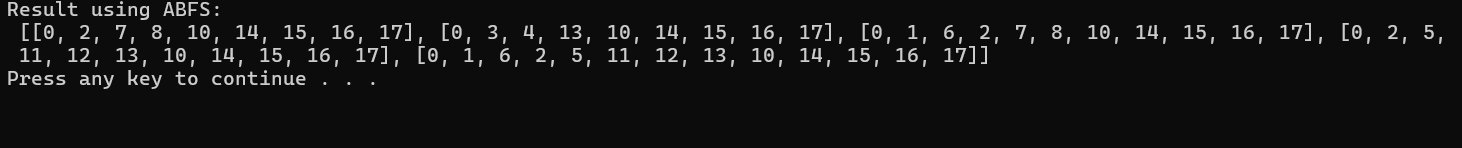
break) và trả về đường đi này.

Vì vậy để in được 5 đường đi cho BFS, ta sửa lại đoạn code như sau.

A computer screen with white and blue text

Description automatically generated

Kết quả



Hơn nữa, ta xây dựng lại tất cả thuật toán bằng cách sử dụng lập trình hướng đối tượng, sử dụng các thuộc tính kế thừa, đa hình, … , tạo các lớp…

A computer screen shot of a program

Description automatically generated

A computer screen shot of white and blue text

Description automatically generated

A computer screen shot of a program code

Description automatically generated

A computer screen with white and blue text

Description automatically generated

A computer screen shot of a program

Description automatically generated

A computer screen with white text

Description automatically generated

A screen shot of a computer program

Description automatically generated

Người báo cáo

Nguyễn Thanh Kiên