**Học viện kỹ thuật quân sự**



**Khoa công nghệ thông tin**

# **A picture containing text, sign Description automatically generated**

**BÁO CÁO BÀI TẬP LỚN**

**MÔN HỌC TRÍ TUỆ NHÂN TẠO**

**Đề tài: Tìm hiểu giải thuật tìm kiếm theo chiều rộng (BFS)**

### Giáo viên hướng dẫn: Ngô Hữu Phúc

### Lớp: KHMT17

### Người thực hiện : Cao Văn Hoàng

### Mã sinh viên: 18150014

### - Hà Nội –

**I. Nội dung tìm hiểu**

**1.** **Nội dung tìm hiểu** :

- Phân tích phương pháp tìm kiếm theo chiều rộng

- Áp dụng cho bài toán: trên mặt phẳng đặt ngẫu nhiên 1000 điểm, giữa các điểm có thể có cạnh hoặc không có cạnh. Tìm đường đi từ một điểm đến một điểm bằng phân tích theo chiều rộng.

**2. Mục tiêu**:

- Xây dựng thuật toán phương pháp tìm kiếm theo chiều rộng

- Nắm được cơ chế hoạt động của thuật toán phương pháp tìm kiếm theo chiều rộng

- Áp dụng tìm đường đi từ 1 điếm đến 1 điểm trên mặt phẳng 1000 điểm bằng phương pháp tìm kiếm theo chiều rộng

1. **Thuật toán phương pháp tìm kiếm theo chiều rộng**
2. **Thuật toán tìm kiếm theo chiều rộng là gì ?**

* [Giải thuật](https://quantrimang.com/cau-truc-du-lieu-va-giai-thuat) tìm kiếm theo chiều rộng (Breadth First Search – viết tắt là BFS) duyệt qua một đồ thị theo chiều rộng và sử dụng hàng đợi (queue) để ghi nhớ đỉnh liền kề để bắt đầu việc tìm kiếm khi không gặp được đỉnh liền kề trong bất kỳ vòng lặp nào.

1. **Ý tưởng thuật toán**

Từ một đỉnh (nút) gốc ban đầu.

* Xác định và lần lượt duyệt các đỉnh kề xung quanh đỉnh gốc vừa xét.
  + Tiếp tục quá trình duyệt qua các đỉnh kề đỉnh vừa xét cho đến khi đạt được kết quả cần tìm hoặc duyệt qua tất cả các đỉnh.

1. **Trình bày thuật toán**

Thuật toán sử dụng một cấu trúc dữ liệu [hàng đợi](https://vi.wikipedia.org/wiki/H%C3%A0ng_%C4%91%E1%BB%A3i) để lưu trữ thông tin trung gian thu được trong quá trình tìm kiếm:

1. Chèn đỉnh gốc vào hàng đợi (đang hướng tới)
2. Lấy ra đỉnh đầu tiên trong hàng đợi và quan sát nó
   * Nếu đỉnh này chính là đỉnh đích, dừng quá trình tìm kiếm và trả về kết quả.
   * Nếu không phải thì chèn tất cả các đỉnh kề với đỉnh vừa thăm nhưng chưa được quan sát trước đó vào hàng đợi.
3. Nếu hàng đợi là rỗng, thì tất cả các đỉnh có thể đến được đều đã được quan sát – dừng việc tìm kiếm và trả về "không thấy".
4. Nếu hàng đợi không rỗng thì quay về bước 2.
5. **Ứng dụng cho thuật toán BFS**
6. **Áp dụng cho mô hình**

\*Mô tả: cho 1 ma trận 10x10 gồm 100 ô và các ô được đánh số từ 1 đến 100

Ma trận chỉ cho phép đi theo hàng ngang và hàng dọc

\*Yêu cầu : Tìm kiếm đường đi từ điểm bắt đầu đến điểm kết thúc bằng thuật toán tìm kiếm theo chiều rộng

A screenshot of a computer

Description automatically generated with medium confidence

1. **Thực hiện giải thuật**

**Bước 1:** xây dựng rào chắn (hay ô mà điểm không thể di chuyển đến)

Tương ứng các ô màu đen sẽ là rào chắn không thể di chuyển đến

A screenshot of a computer game

Description automatically generated with medium confidence

**Bước 2**: Xác định điểm bắt đầu và điểm kết thúc cho bài toán

Điểm bắt đầu được đánh dấu bởi ô màu đỏ (tưng ứng là ô thứ 14)

Điểm kết thúc được đánh dấu bởi ô màu xanh (tưng ứng là ô thứ 56)

A screenshot of a chess board

Description automatically generated with low confidence

**Bước 3:** Sử dụng thuật toán tìm kiếm theo chiều rộng để tìm kiếm đường đi từ điểm đầu đến điểm cuối

* **Đưa ra thuật toán**

while queue:

            # pop the first path from the queue

            path = queue.pop(0)

            # get the last node from the path

            node = path[-1]

            if node not in explored:

                neighbors = self.graph[node]

                # go through all neighbor nodes

                # push it into the queue

                for neighbor in neighbors:

                    new\_path = list(path)

                    new\_path.append(neighbor)

                    queue.append(new\_path)

                    print(new\_path)

                    if neighbor == self.goal:

                        print(new\_path)

                        return new\_path

                # mark node as explored

                explored.append(node)

* **Mô tả thuật toán qua bảng phía dưới:**

*Định nghĩa:*

* Node là điểm đang xét
* Neighbors là các điểm kề
* Queue là Open
* Explored là Close

*Các bước thực hiện:*

* Kiểm tra trong khi queue còn khác rỗng
* Lấy điểm cuối cùng trong queue ra
* Kiểm tra xem điểm đã được duyệt chưa ,nếu chưa thì đưa các điểm kề vào neighbors
* Kiểm tra xem các đỉnh kề có điểm kết thúc không thì kết thúc,nếu ko sẽ đưa neighbors vào queue và quay trở lại từ đầu
* **Chạy thuật toán**

Ta có đường đi từ 1 điểm đến các điểm khác:

{ 2: [3], 3: [2, 4], 4: [3, 5, 14], 5: [4, 15], 7: [8], 8: [7, 9, 18], 9: [8], 14: [15, 24, 4], 15: [14, 16, 5], 16: [15, 26], 18: [8], 20: [30], 21: [31], 23: [24, 33], 24: [23, 34, 14], 26: [27, 16], 27: [26, 37], 29: [30], 30: [29, 20], 31: [41, 21], 33: [34, 43, 23], 34: [33, 35, 24], 35: [34, 45], 37: [38, 47, 27], 38: [37, 48], 41: [42, 31], 42: [41, 43, 52], 43: [42, 53, 33], 45: [46, 35], 46: [45, 47, 56], 47: [46, 48, 57, 37], 48: [47, 49, 38], 49: [48, 59], 52: [53, 62, 42], 53: [52, 63, 43], 56: [57, 66, 46], 57: [56, 47], 59: [60, 69, 49], 60: [59, 70], 61: [62, 71], 62: [61, 63, 52], 63: [62, 64, 73, 53], 64: [63, 65], 65: [64, 66], 66: [65, 76, 56], 68: [69, 78], 69: [68, 70, 59], 70: [69, 60], 71: [81, 61], 73: [83, 63], 76: [77, 86, 66], 77: [76, 78], 78: [77, 68], 81: [71], 83: [84, 73], 84: [83, 94], 86: [76], 94: [95, 84], 95: [94], 97: [98], 98: [97] }

| **ĐIỂM ĐANG XÉT** | **CÁC ĐIỂM KỀ** | **Open** | **Close** |
| --- | --- | --- | --- |
|  |  | {14} | {} |
| 14 | {15; 24; 4} | {15; 24; 4} | {14} |
| 15 | {14; 16; 5} | {24; 4; 16; 5} | {14; 15} |
| 24 | {23; 34; 14} | {4; 16; 5; 23; 34} | {14; 15; 24} |
| 4 | {3; 5; 14} | {16; 5; 23; 34; 3} | {14; 15; 24; 4} |
| 16 | {15; 26} | {5; 23; 34; 3; 26} | {14; 15; 24; 4; 16} |
| 5 | {4; 15} | {23; 34; 3; 26} | {14; 15; 24; 4; 16; 5} |
| 23 | {24; 33} | {34; 3; 26; 33} | {14; 15; 24; 4; 16; 5; 23} |
| 34 | {33; 35; 24} | {3; 26; 33; 35; 24} | {14; 15; 24; 4; 16; 5; 23; 34} |
| 3 | {2; 4} | {26; 33; 35; 2} | {14; 15; 24; 4; 16; 5; 23; 34; 3} |
| 26 | {27; 16} | {33; 35; 2; 27} | {14; 15; 24; 4; 16; 5; 23; 34; 3; 26} |
| 33 | {34; 43; 23} | {35; 2; 27; 43} | {14; 15; 24; 4; 16; 5; 23; 34; 3; 26; 33} |
| 35 | {34; 45} | {2; 27; 43;45} | {14; 15; 24; 4; 16; 5; 23; 34; 3; 26; 33; 35} |
| 2 | {3} | {27; 43;45} | {14; 15; 24; 4; 16; 5; 23; 34; 3; 26; 33; 35; 2} |
| 27 | {26; 37} | {43; 45; 37} | {14; 15; 24; 4; 16; 5; 23; 34; 3; 26; 33; 35; 2; 27} |
| 43 | {42; 53; 33} | {45; 37; 42; 53} | {14; 15; 24; 4; 16; 5; 23; 34; 3; 26; 33; 35; 2; 27; 43} |
| 45 | {46; 35} | {37; 42; 53; 46} | {14; 15; 24; 4; 16; 5; 23; 34; 3; 26; 33; 35; 2; 27; 43; 45} |
| 37 | {38; 47; 27} | {42; 53; 46; 38; 47} | {14; 15; 24; 4; 16; 5; 23; 34; 3; 26; 33; 35; 2; 27; 43; 45; 37} |
| 42 | {43; 52} | {53; 46; 38; 47; 52} | {14; 15; 24; 4; 16; 5; 23; 34; 3; 26; 33; 35; 2; 27; 43; 45; 37; 42} |
| 53 | {52; 63; 43} | {46; 38; 47; 52; 63} | {14; 15; 24; 4; 16; 5; 23; 34; 3; 26; 33; 35; 2; 27; 43; 45; 37; 42; 53} |
| 46 | {45;47;56} | {38; 47; 52; 63} | {14; 15 ;24; 4; 16; 5; 23; 34; 3; 26; 33; 35;2; 27; 43; 45; 37; 42; 53; 46} |

Thuật toán sẽ xét duyệt tất cả các điểm : { 14; 15 ;24; 4; 16; 5; 23; 34; 3; 26; 33; 35; 2; 27; 43; 45; 37; 42; 53; 46 }

Tổng tất cả các điểm mà thuật toán xét duyệt để tìm đường đi từ điểm bắt đầu đến điểm kết thúc là 20 điểm.

[14, 24]

[14, 4]

[14, 15, 14]

[14, 15, 16]

[14, 15, 5]

[14, 24, 23]

[14, 24, 34]

[14, 24, 14]

[14, 4, 3]

[14, 4, 5]

[14, 4, 14]

[14, 15, 16, 15]

[14, 15, 16, 26]

[14, 15, 5, 4]

[14, 15, 5, 15]

[14, 24, 23, 24]

[14, 24, 23, 33]

[14, 24, 34, 33]

[14, 24, 34, 35]

[14, 24, 34, 24]

[14, 4, 3, 2]

[14, 4, 3, 4]

[14, 15, 16, 26, 27]

[14, 15, 16, 26, 16]

[14, 24, 23, 33, 34]

[14, 24, 23, 33, 43]

[14, 24, 23, 33, 23]

[14, 24, 34, 35, 34]

[14, 24, 34, 35, 45]

[14, 4, 3, 2, 3]

[14, 15, 16, 26, 27, 26]

[14, 15, 16, 26, 27, 37]

[14, 24, 23, 33, 43, 42]

[14, 24, 23, 33, 43, 53]

[14, 24, 23, 33, 43, 33]

[14, 24, 34, 35, 45, 46]

[14, 24, 34, 35, 45, 35]

[14, 15, 16, 26, 27, 37, 38]

[14, 15, 16, 26, 27, 37, 47]

[14, 15, 16, 26, 27, 37, 27]

[14, 24, 23, 33, 43, 42, 41]

[14, 24, 23, 33, 43, 42, 43]

[14, 24, 23, 33, 43, 42, 52]

[14, 24, 23, 33, 43, 53, 52]

[14, 24, 23, 33, 43, 53, 63]

[14, 24, 23, 33, 43, 53, 43]

[14, 24, 34, 35, 45, 46, 45]

[14, 24, 34, 35, 45, 46, 47]

[14, 24, 34, 35, 45, 46, 56]

Kết quả màn hình sẽ chạy ra đường đi mà thuật toán BFS tìm được:

[14, 24, 34, 35, 45, 46, 56]

A screenshot of a game

Description automatically generated with low confidence

## **Nhận xét**

## **Ưu điểm**

* Xét duyệt tất cả các đỉnh để trả về kết quả.
* Nếu số đỉnh là hữu hạn, thuật toán chắc chắn tìm ra kết quả.

**Khuyết điểm**

* Mang tính chất vét cạn, không nên áp dụng nếu duyệt số đỉnh quá lớn.
* Mang tính chất mù quáng, duyệt tất cả đỉnh, không chú ý đến thông tin trong các đỉnh để duyệt hiệu quả, dẫn đến duyệt qua các đỉnh không cần thiết.