Trường Đại học Khoa học Tự nhiên TP. HCM

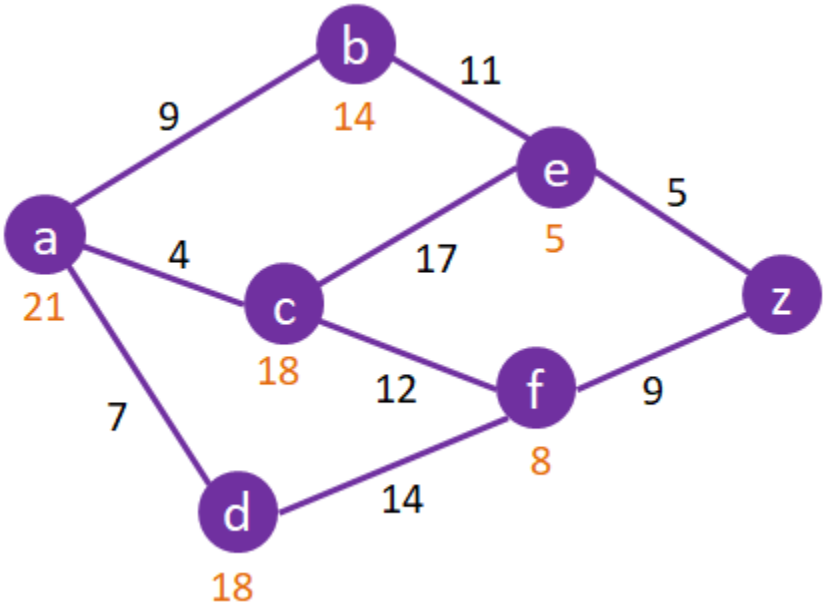
Khoa Công nghệ thông tin

Bộ môn: Khoa học máy tính

ĐỒ ÁN THỰC HÀNH 1

MÔN: CƠ SỞ TRÍ TUỆ NHÂN TẠO

SEARCH



MSSV: 18120584

Họ và tên: PHẠM ĐÌNH THỤC

Thành phố Hồ Chí Minh, ngày 31, tháng 11, năm 2020

I. THÔNG TIN NGƯỜI THỰC HIỆN.

Họ và tên: Phạm Đình Thục
MSSV: 18120584
Lớp: CS_TTNT 18_21

II. CÔNG VIỆC & ĐÁNH GIÁ MỨC ĐỘ HOÀN THÀNH

STT	CÔNG VIỆC	HOÀN THÀNH
1	Thuật toán BFS	10/10
2	Thuật toán DFS	10/10
3	Thuật toán UCS	10/10
4	Thuật toán A* (ASTAR)	10/10
5	Thuật toán GBFS	10/10
TỔNG CỘNG (Tự đánh giá)		10/10

III. LÝ THUYẾT CƠ BẢN CÁC THUẬT TOÁN

1. Thuật toán BFS:

Tìm kiếm theo chiều rộng (Breadth First Search - BFS): là một thuật toán tìm kiếm trên đồ thị vô hướng hoặc có hướng và không có trọng số, bao gồm 2 thao tác: cho trước 1 đỉnh và thêm các đỉnh kề với đỉnh vừa cho. Có thể dùng để tìm đường đi từ gốc tới đỉnh đích hay tới tất cả các đỉnh khác. Vì đồ thị này không có trọng số nên đường đi là ngắn nhất có thể.

Thuật toán bắt đầu từ đỉnh gốc (hoặc đỉnh coi như gốc) và lần lượt nhìn các đỉnh kề với đỉnh gốc. Sau đó, với mỗi đỉnh trong số đó, thuật toán lại lần lượt nhìn các đỉnh kề với nó à chưa được quan sát trước đó và lặp lại.

Độ phức tạp của thuật toán:

- Độ phức tạp thời gian: $O(b^d)$
- Độ phức tạp không gian: $O(b^d)$

(với d: độ sâu cây thuật toán)

2. Thuật toán DFS:

Tìm kiếm theo chiều sâu (Depth First Search - DFS): là một thuật toán tìm kiếm trên đồ thị vô hướng hoặc có hướng và không có trọng số. Thuật toán bắt đầu tại gốc (hoặc đỉnh coi như gốc) và phát triển xa nhất có thể theo từng nhánh.

DFS là 1 dạng tìm kiếm không đầy đủ mà quá trình tìm kiếm được phát triển tới đỉnh con đầu tiên của nút đang tìm kiếm cho tới khi gặp được đỉnh cần tìm hoặc tới một nút không có con. Khi đó giải thuật quay lui về đỉnh vừa tìm kiếm ở bước trước và bắt đầu ở nút con tiếp theo.

Độ phức tạp của thuật toán:

- Độ phức tạp thời gian: $O(b^m)$
- Độ phức tạp không gian: O(bm)

(với m: độ sâu tối đa cây thuật toán)

3. Thuật toán UCS:

Tìm kiếm chi phí đều (Uniform Cost Search - UCS) là một cách duyệt cây dùng cho việc duyệt hay tìm kiếm trên cây có trọng lượng chi phí.

Thuật toán sẽ phát triển các nút chưa xét có chi phí thấp nhất – các nút được xét theo thứ tự chi phí tăng dần.Khi đồ thi có chi phí ở mỗi bước là như nhau thì thuật toán trở thành phương pháp tìm kiếm theo chiều rông

Độ phức tạp của thuật toán:

- Độ phức tạp thời gian: $O(b^{C^*/e})$
- Độ phức tạp không gian: $O(b^{C^*/e})$

(với b: hệ số phân nhánh
C: chi phí giải pháp tối ưu*
e: chi phí một bước tối thiểu)

4. Thuật toán A* (ASTAR)

A * là một thuật toán tìm kiếm ưu tiên nhất, nghĩa là nó được xây dựng dưới dạng đồ thị có trọng số. Bắt đầu từ một nút gốc cụ thể của biểu đồ, nó nhằm mục đích tìm đường dẫn đến nút mục tiêu đã cho có giá trị nhỏ nhất chi phí (quãng đường di chuyển ít nhất, thời gian ngắn nhất, v.v.). Nó thực hiện điều này bằng cách duy trì một cây các đường dẫn bắt đầu từ nút bắt đầu và mở rộng các đường dẫn đó một cạnh cho đến khi tiêu chí kết thúc của nó được thỏa mãn. Tại mỗi lần lặp của vòng lặp chính, A * cần xác định đường dẫn nào của nó để mở rộng. Nó làm như vậy dựa trên chi phí của đường dẫn và ước tính chi phí cần thiết để mở rộng đường dẫn đến mục tiêu.

f(n) = g(n) + h(n) (với n là nút, g(n) là chí phí từ nút gốc đến n và h(n) là một heuristic ước tính chi phí nhỏ nhất từ n đến đích), h(n) được sử dụng là khoảng cách Euclidean.

Độ phức tạp của thuật toán:

- Độ phức tạp về thời gian:

  |h(x) – h*(x)| = O(log h*(x). A* phụ thuộc vào hàm heuristic h(n),

- Độ phức tạp về không gian: gần giống với các thuật toán khác.

1. Phần thêm.

Thuật toán Greedy Best First Search (GBFS)

Thuật toán greedy best first search là một chiến lược tìm kiếm với tri thức bổ sung từ việc sử dụng các tri thức cụ thể của bài toán.

Thuật toán sẽ sử dụng 1 hàm đánh giá là hàm heuristic h(n). h(n) được tính bằng khoảng cách Euclidean, hàm heuristic h(n) này sẽ đánh giá chi phí để đi từ nút hiện tại n đến nút đích (mục tiêu). phương pháp này sẽ xét các nút có vẻ gần với nút đích nhất vì nó chỉ phát triển từ những nút có ước lượng tốt nhất.

Độ phức tạp thuật toán: Đây chỉ đơn giản là tìm kiếm theo chiều rộng, nhưng với các nút được sắp xếp lại theo giá trị heuristic của chúng

Độ phức tạp về thời gian: $O(b^{d+1})$

Độ phức tạp về không gian: $O(b^{d})$

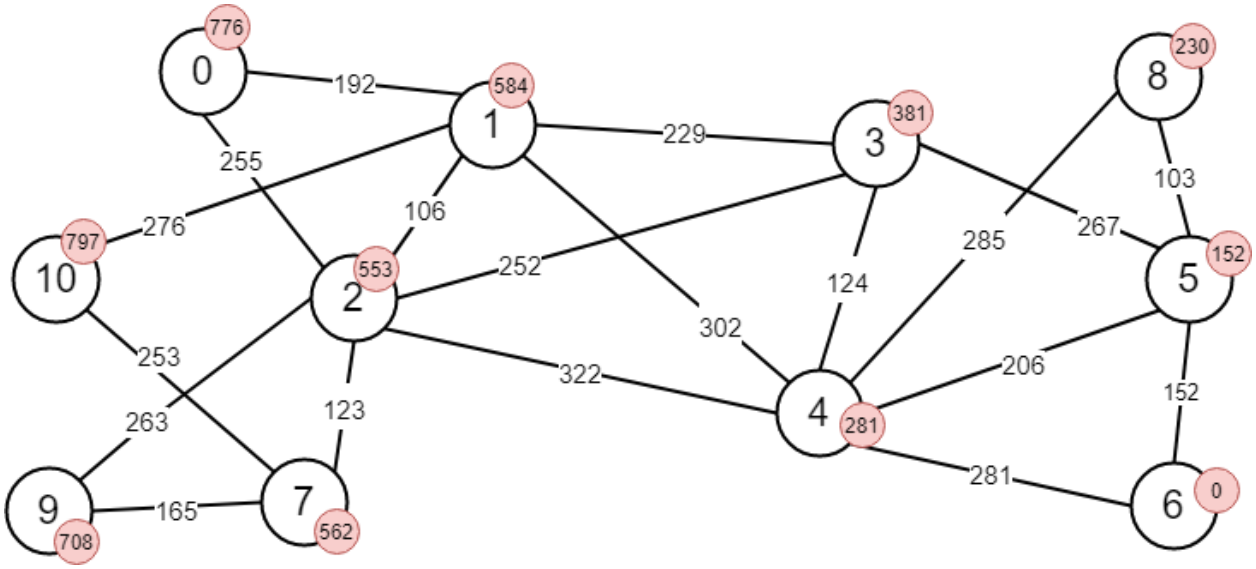
(với d: độ sâu cây thuật toán)

IV. SO SÁNH THUẬT TOÁN UCS VÀ A*:

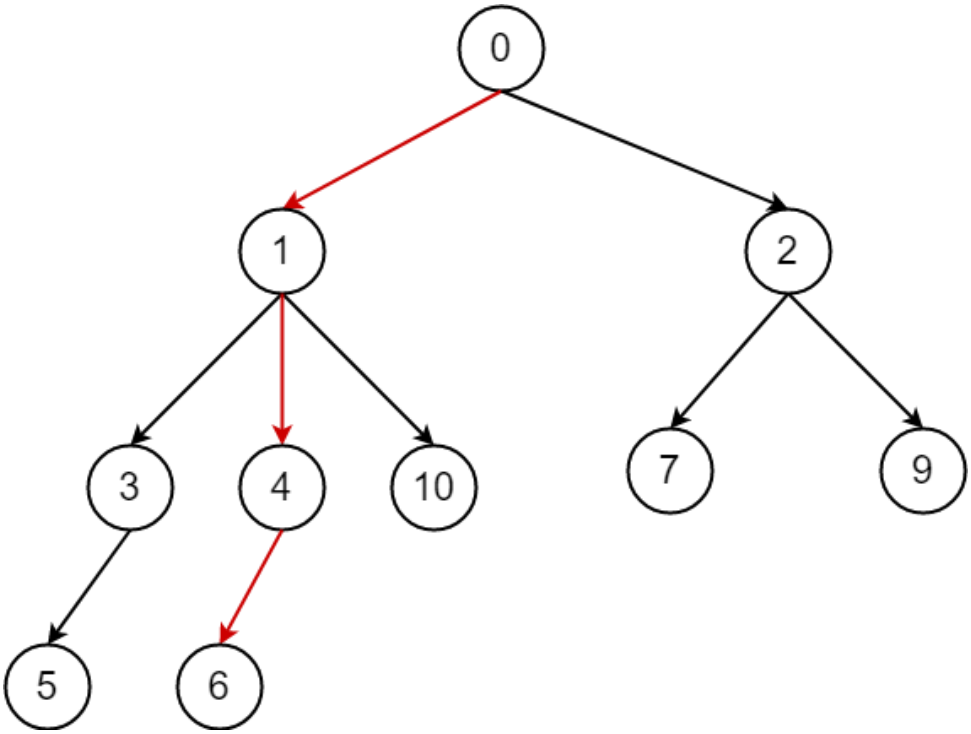
UCS	A*
Hàng đợi gồm chí phí từ gốc đến điểm nút. Là tìm kiếm không có thông tin. Nó mở rộng nút có chi phí thấp nhất và làm như vậy theo mọi hướng vì không có thông tin về đích được cung cấp. Nó được xem là 1 phần của hàm f(n) vì chỉ có g(n).	Hàng đợi gồm chí phí từ gốc đến điểm nút và chí phí ước lượng từ nút đến đích. Sự khác biệt là ở hàm heuristic. Là tìm kiếm có thông tin, sử dụng heuristic để ước lượng mức độ gần nhất của trạng thái hiện tại mới đích Nó là 1 hàm f(n) đầy đủ với g(n) là chi phí đường đi từ gốc đến điểm cần xét và h(n) là ước tính chi phí ngắn nhất từ điểm cần xét đến đích A* cố gắng giảm thiểu số lượng nút mở rộng so với UCS

V. CHI TIẾT MỖI THUẬT TOÁN VÀ CÀI ĐẶT

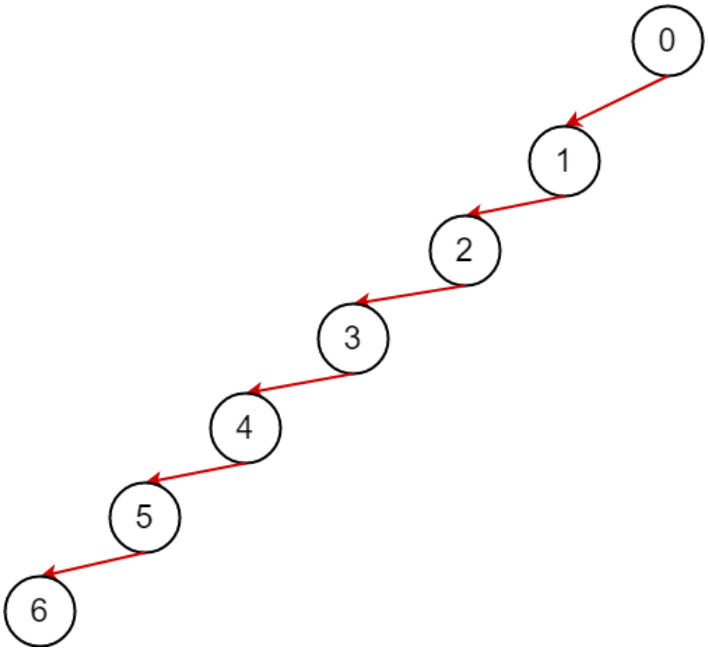
Test case 1:



Thuật toán BFS: 0 – 1 – 4 – 6



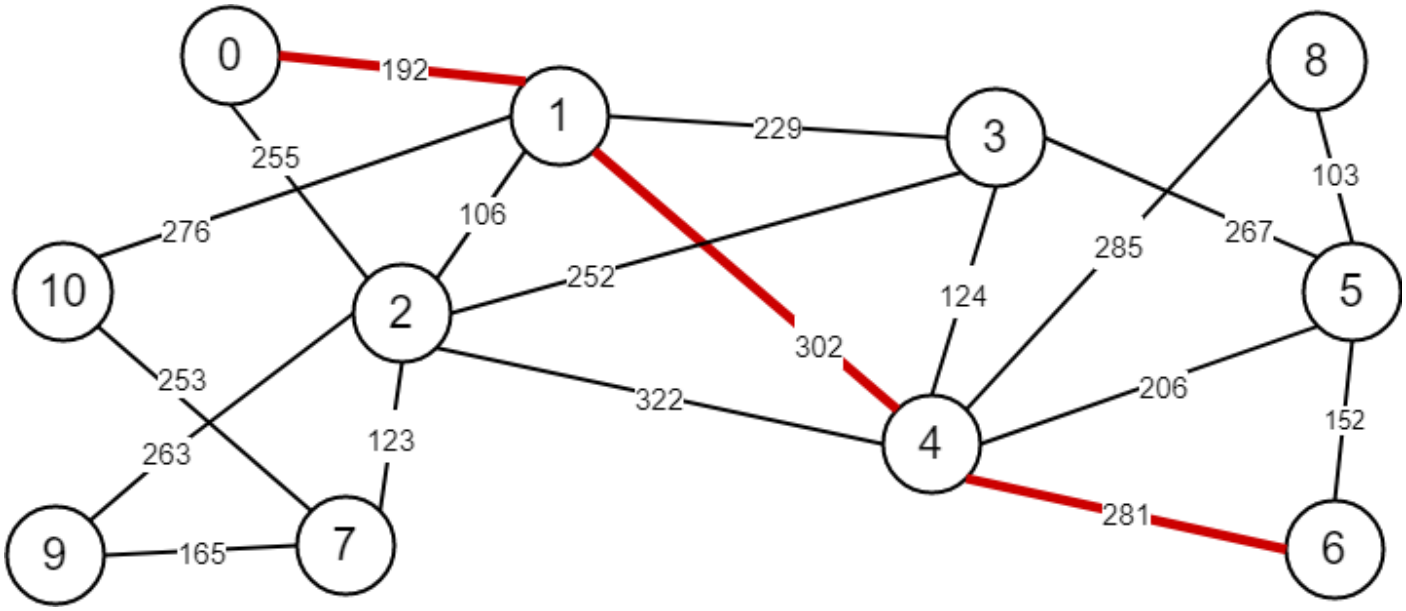
- **Thuật toán DFS: 0 – 1 – 2 – 3 – 4 – 5 – 6**



- **<u>Thuật toán UCS</u>**: Chí phí: tính theo khoảng cách Euclidean
  $g(n) = sqrt((x_1 - x_2)^2 + (y_1 - y_2)^2)$.

Các bước	Hàng đợi	Mở rộng	Đã xét
1	{(0,0)}	0	{}
2	{(1; 192), (2; 255)}	1	{0}
3	{(2; 255),(3, 421), (10;276), (4; 494)}	2	{0, 1}
4	{(7; 378), (3; 421), (10; 468), (4; 494), (9; 519)}	7	{0, 1, 2}
5	{(3; 421), (10; 468), (4, 494), (9; 519)}	3	{0, 1, 2, 7}
6	{(10; 468), (4, 494), (9; 519), (5, 688)}	10	{0, 1, 2, 7, 3}
7	{(4, 494), (9; 519), (5, 688)}	4	{0, 1, 2, 7, 3, 10}
8	{(9; 519), (5, 688), (6; 776), (8, 780)}	9	{0, 1, 2, 7, 3, 10, 4}
9	{(5, 688), (6; 776), (8, 780)}	5	{0, 1, 2, 7, 3, 10, 4, 9}
10	{(6; 776), (8, 780)}	**<u>6</u>**	{0, 1, 2, 7, 3, 10, 4, 9, 5}

Đường đi được tìm thấy: **0 – 1 – 4 – 6**



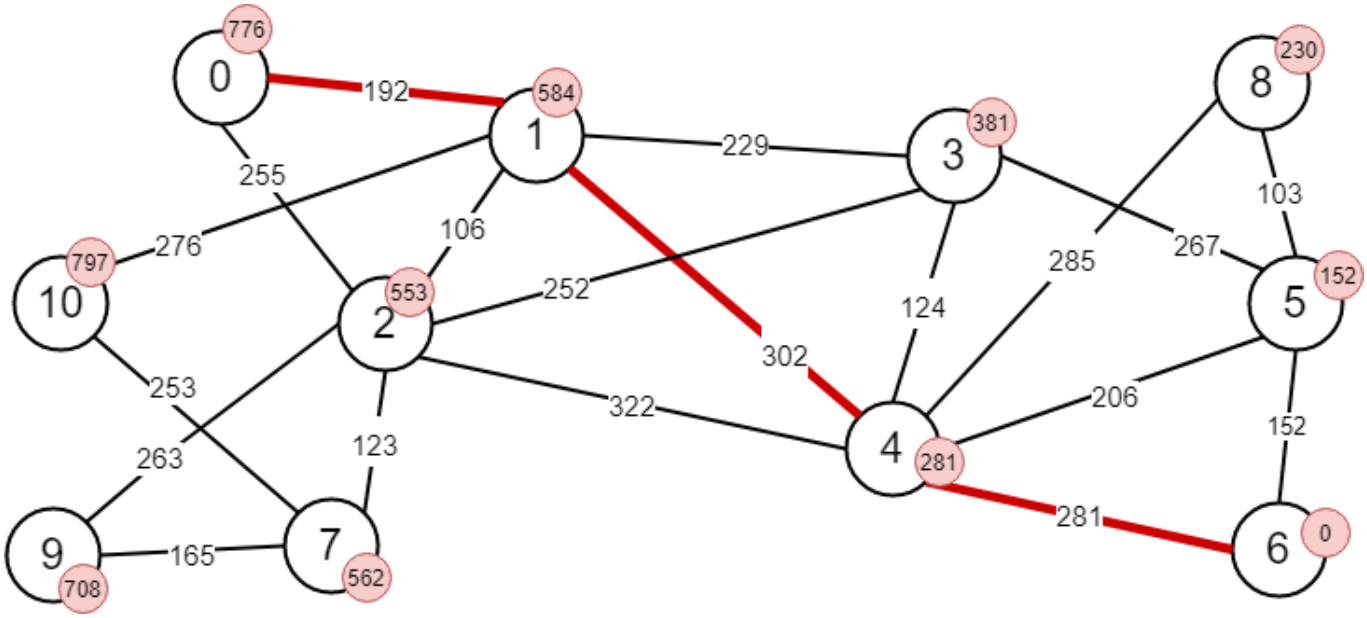
Thuật toán A*:

f(n) = g(n) + h(n)

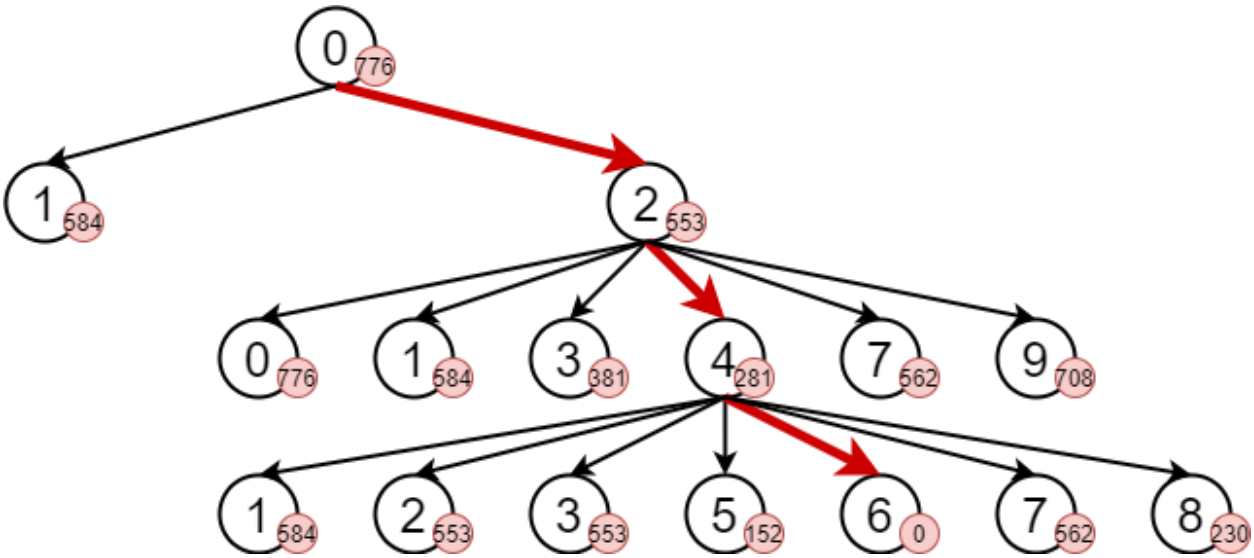
với g(n): khoảng cách giữa các cạnh, h(n) khoảng cách Euclidean từ nút tới đích.

Các bước	Hàng đợi	Mở rộng	Đã xét
1	{(0; 776)}	0	{}
2	{(1; 776), (2; 809)}	1	{(0; 776)}
3	{(4; 776), (3, 802), (2; 809), (10;1265)}	4	{(0; 776), (1; 776)}
4	{(6; 776), (3, 802), (2; 809), (5; 853), (8; 1010) (10;1265), (7; 1379)}	**6**	{(0; 776), (1; 776), (4; 776)}

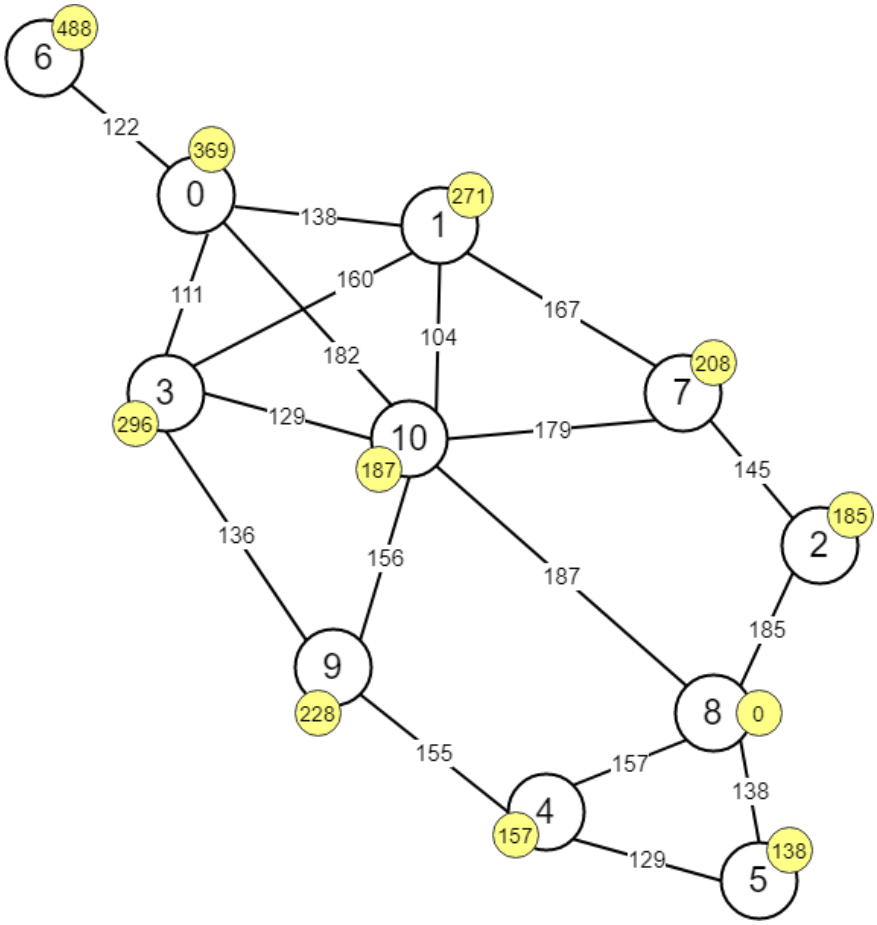
Đường đi được tìm thấy: **0 – 1 – 4 – 6**



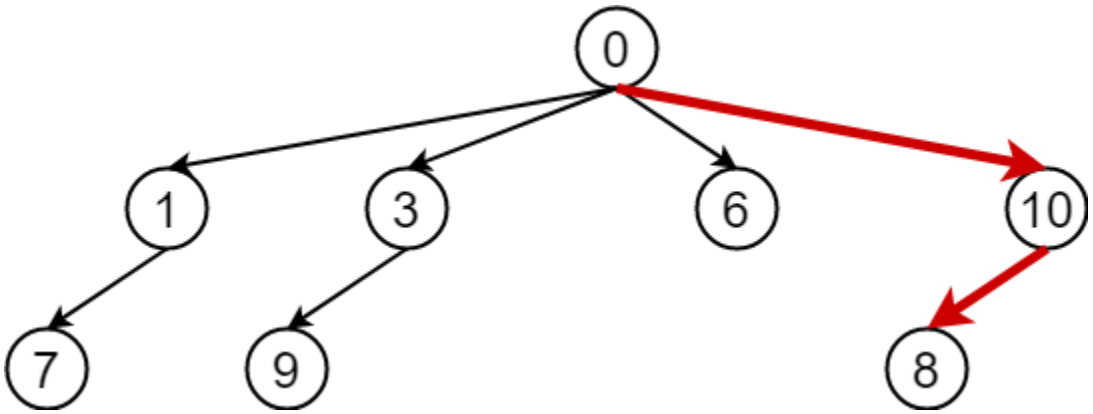
Thuật toán GBFS: 0 – 2 – 4 – 6



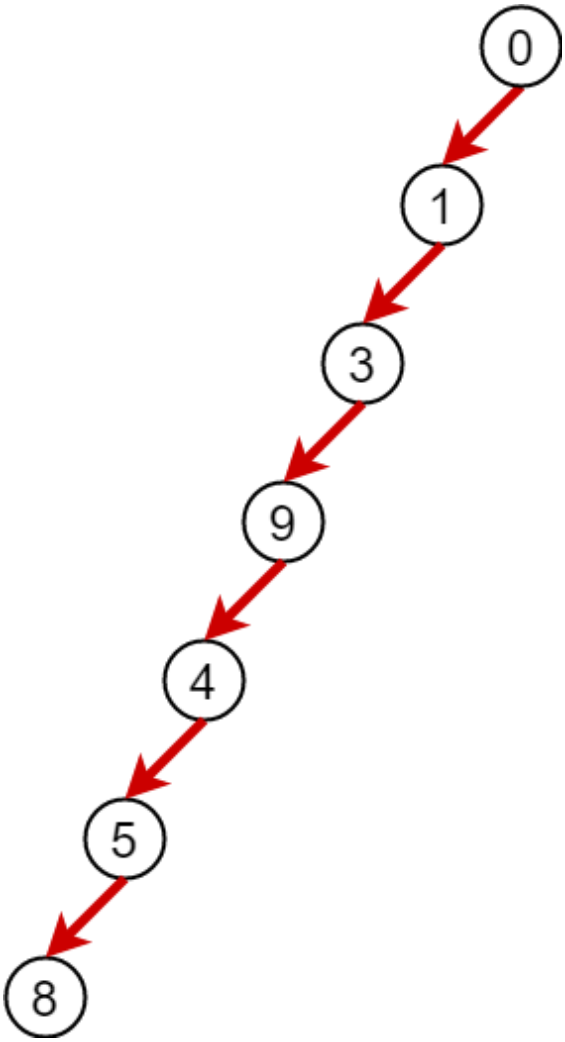
Test case 2:



Thuật toán BFS: 0 – 10 – 8



Thuật toán DFS: 0 – 1 – 3 – 9 – 4 – 5 – 8



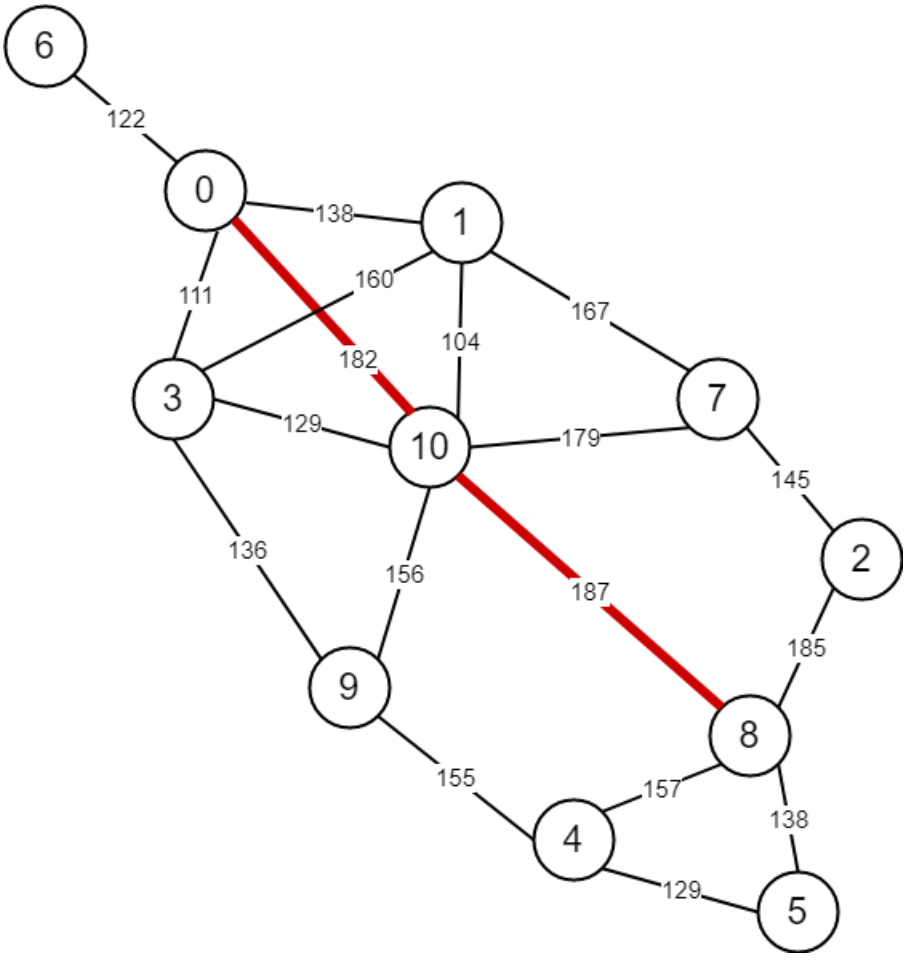
Thuật toán UCS:

Chí phí: tính theo khoảng cách Euclidean

$g(n) = sqrt((x_1 - x_2)^2 + (y_1 - y_2)^2)$.

Các bước	Hàng đợi	Mở rộng	Đã xét
1	{(0,0)}	0	{}
2	{(3; 111), (6; 121), (1; 138), (10; 182)}	3	{0}
3	{(6; 121), (1; 138), (10; 182), (9; 248)}	6	{0, 3}
4	{(1; 138), (10; 182), (9; 248)}	1	{0, 3, 6}
5	{(10; 182), (9; 248), (7, 305)}	10	{0, 3, 6, 1}
6	{(9; 248), (7, 305), (8; 369)}	9	{0, 3, 6, 1, 10}
7	{(7, 305), (8; 369), (4; 403)}	7	{0, 3, 6, 1, 10, 9}
8	{(8; 369), (4; 403), (2; 451)}	**8**	{0, 3, 6, 1, 10, 9, 7}

Đã tìm thấy đường đi: **0 – 10 – 8**



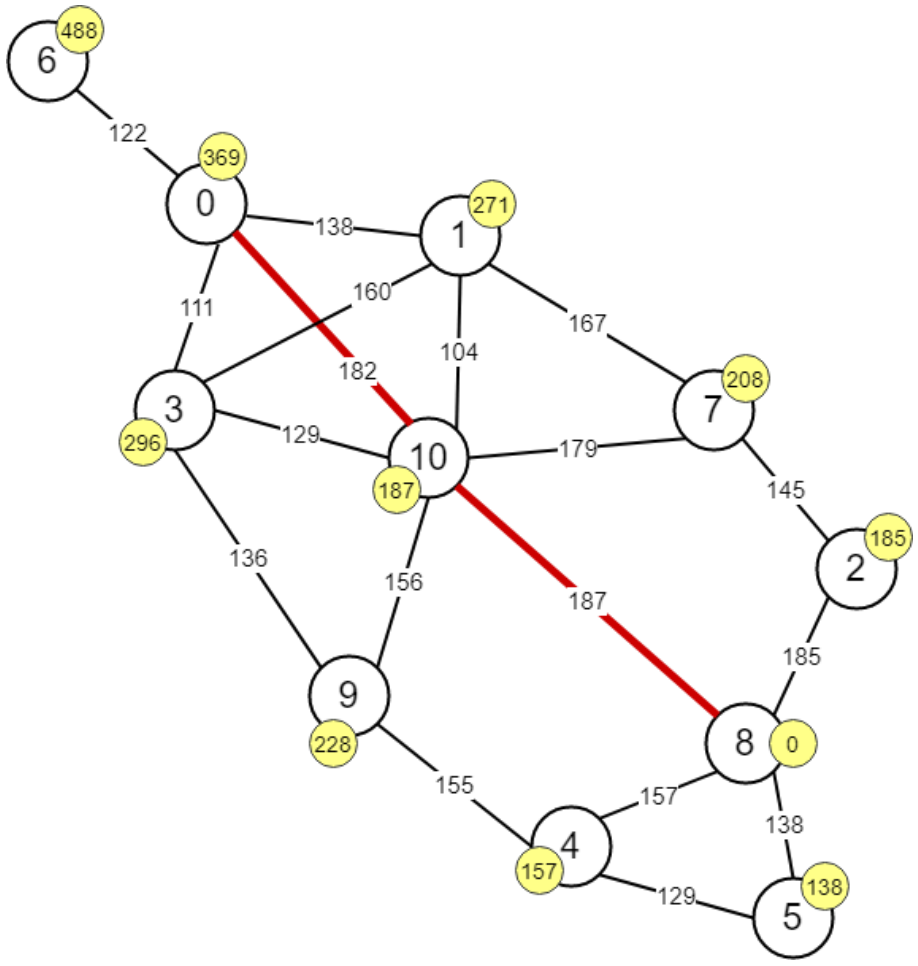
Thuật toán A*:

f(n) = g(n) + h(n)

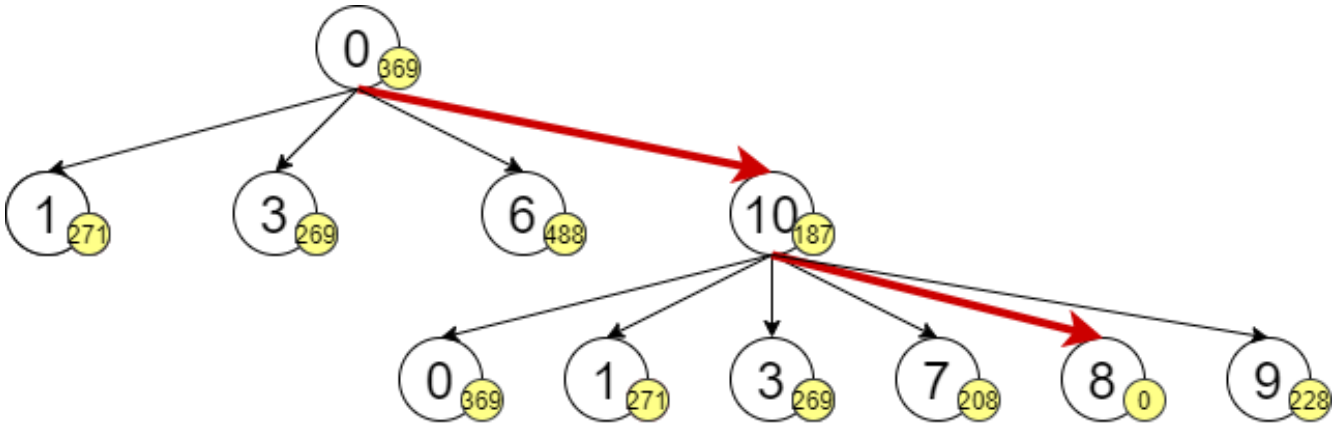
với g(n): khoảng cách giữa các cạnh, h(n) khoảng cách Euclidean từ nút tới đích.

Các bước	Hàng đợi	Mở rộng	Đã xét
1	{(0; 369)}	0	{}
2	{(10; 369), (3; 408), (1; 409), (6; 610)}	10	{(0; 369)}
3	{(8; 369), (3; 408), (1; 409), (9; 567), (7; 570), (6; 610)}	**8**	{(0; 776), (10; 369)}

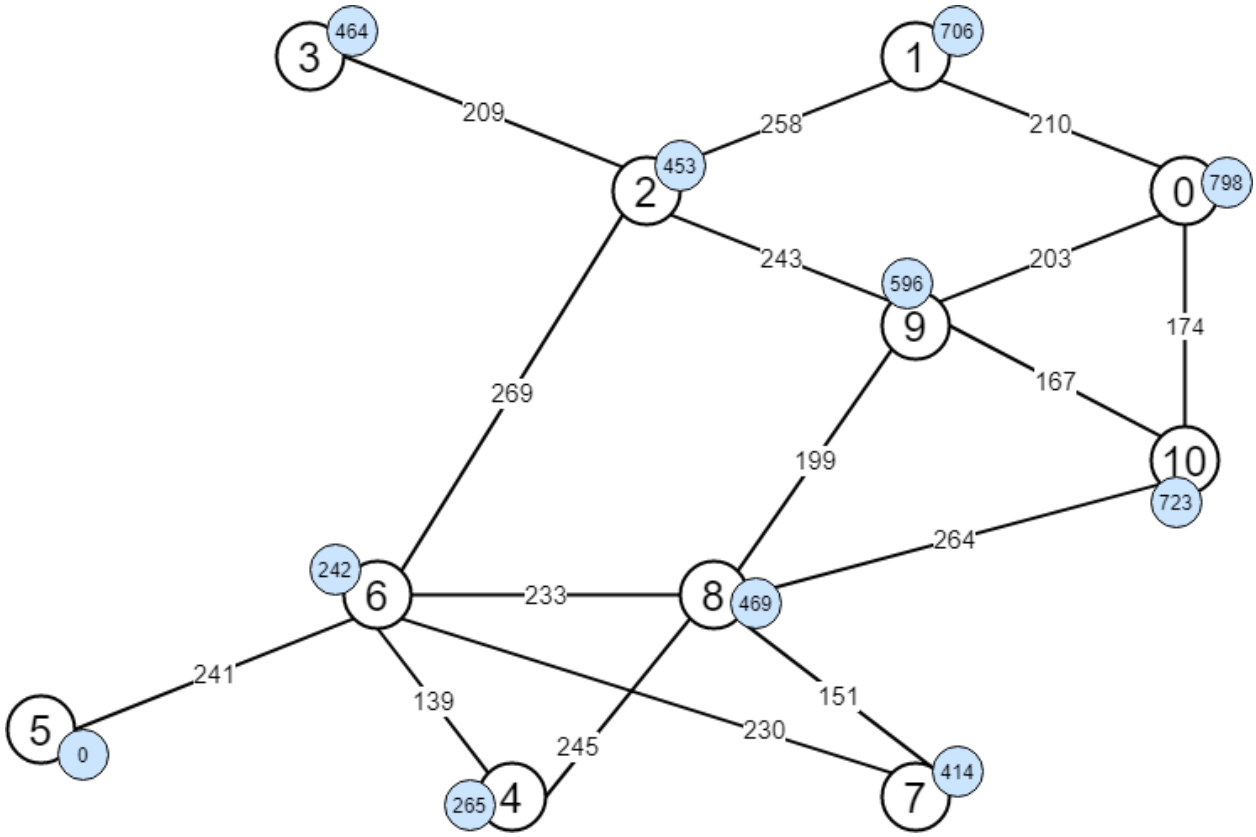
Đã tìm thấy đường đi: **0 – 10 – 8**



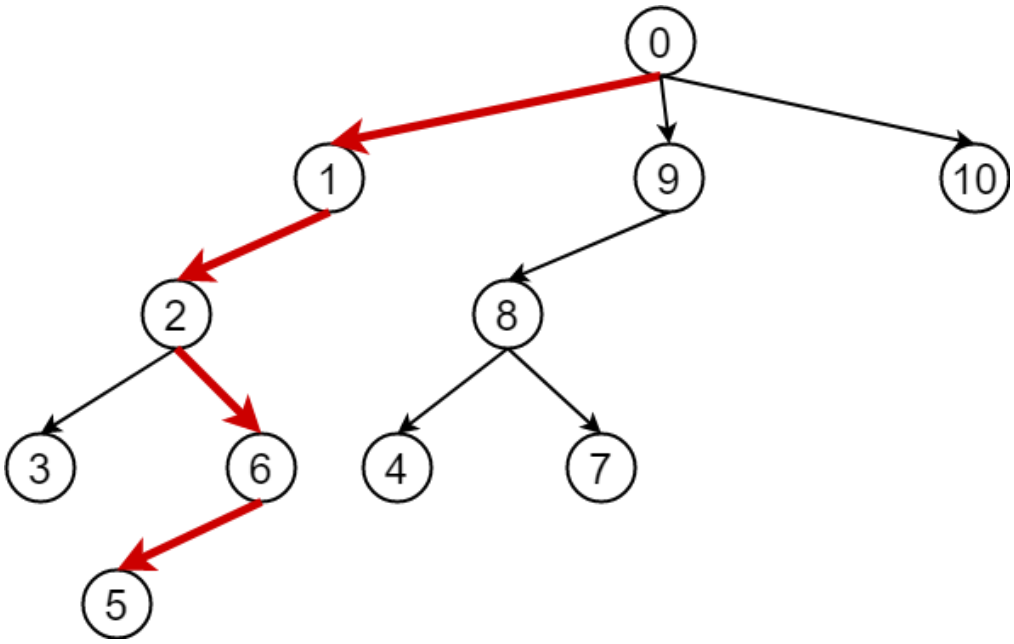
Thuật toán GBFS: 0 – 10 – 8



Test case 3:

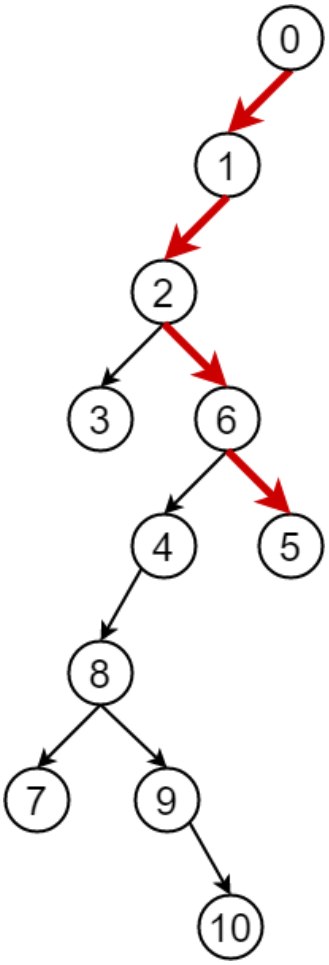


Thuật toán BFS: **0 – 1 – 2 – 6 – 5**



Thuật toán DFS:

0 – 1 – 2 – 6 – 5



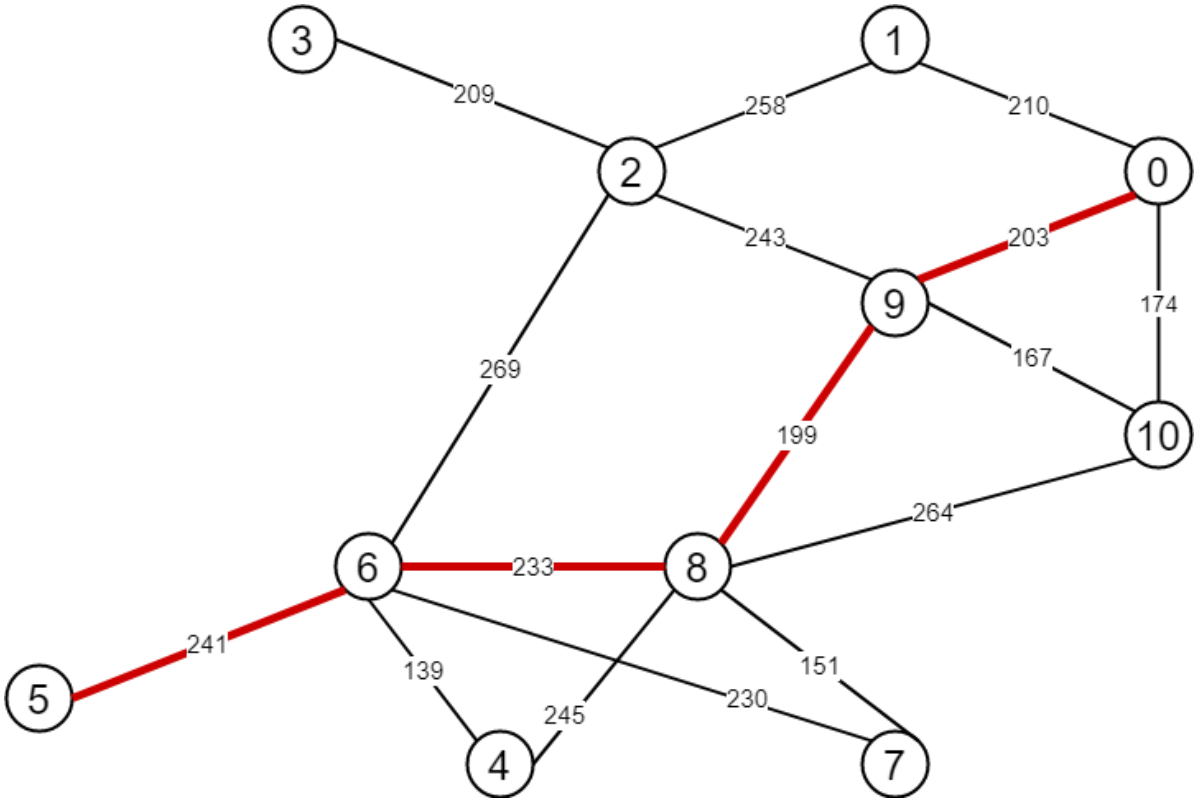
Thuật toán UCS:

Chí phí: tính theo khoảng cách Euclidean

$g(n) = sqrt((x_1 - x_2)^2 + (y_1 - y_2)^2)$.

Các bước	Hàng đợi	Mở rộng	Đã xét
1	{(0,0)}	0	{}
2	{(10; 174), (9; 203), (1; 210)}	10	{0}
3	{(9; 203), (1; 210), (8; 439)}	9	{0, 3, 10}
4	{(1; 210), (8; 402), (2; 446)}	1	{0, 3, 10, 9}
5	{(8; 402), (2; 446)}	8	{0, 3, 10, 9, 1}
6	{(2; 446), (7; 553), (6; 636); (4; 648)}	2	{0, 3, 10, 9, 1, 8}
7	{(7; 553), (6; 636); (4; 648), (3; 656)}	7	{0, 3, 10, 9, 1, 8, 2}
8	{(6; 636), (4; 648), (3; 656)}	6	{0, 3, 10, 9, 1, 8, 2, 7}
9	{(4; 648), (3; 656), (5; 878)}	4	{0, 3, 10, 9, 1, 8, 2, 7, 6}
10	{(3; 656), (5; 878)}	3	{0, 3, 10, 9, 1, 8, 2, 7, 6, 4}
11	{(5; 878)}	**5**	{0, 3, 10, 9, 1, 8, 2, 7, 6, 4, 3}

Đã tìm thấy đường đi: **0 – 9 – 8 – 6 – 5**



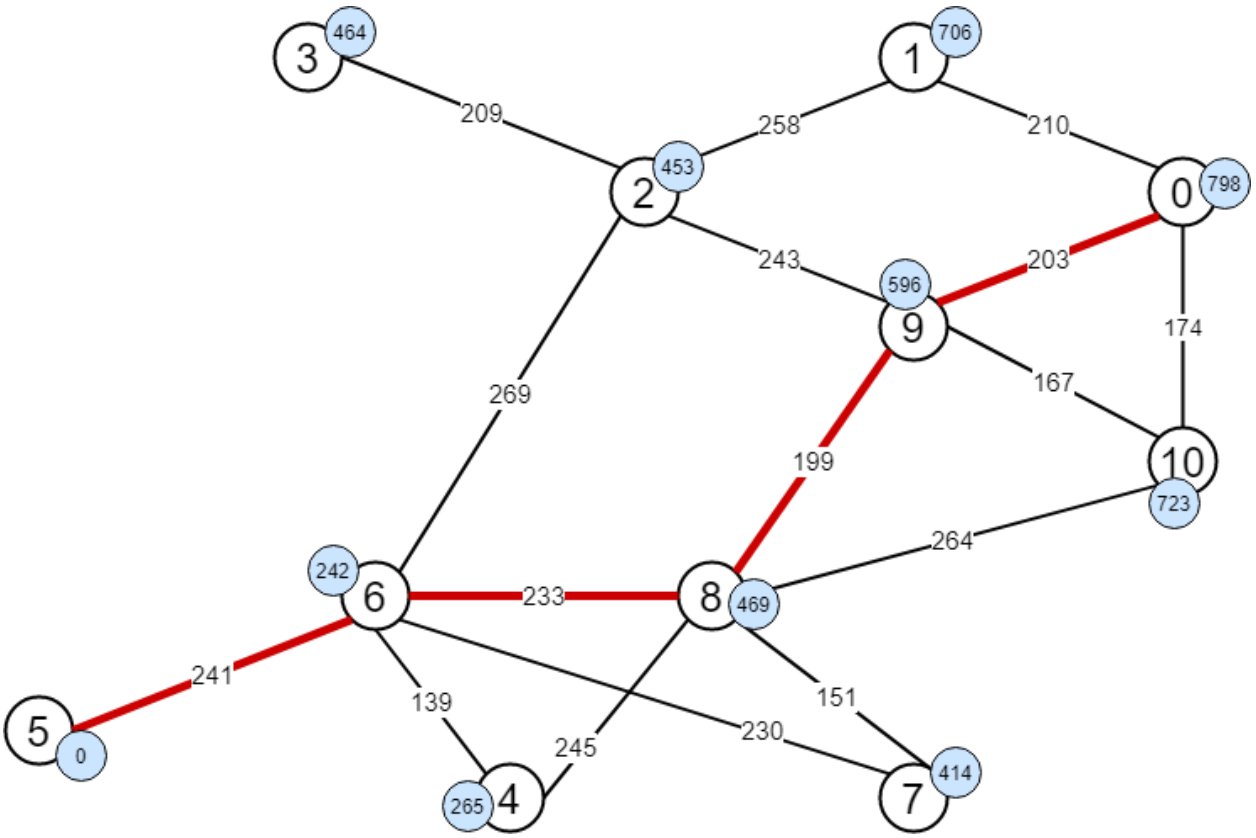
Thuật toán A*:

f(n) = g(n) + h(n)

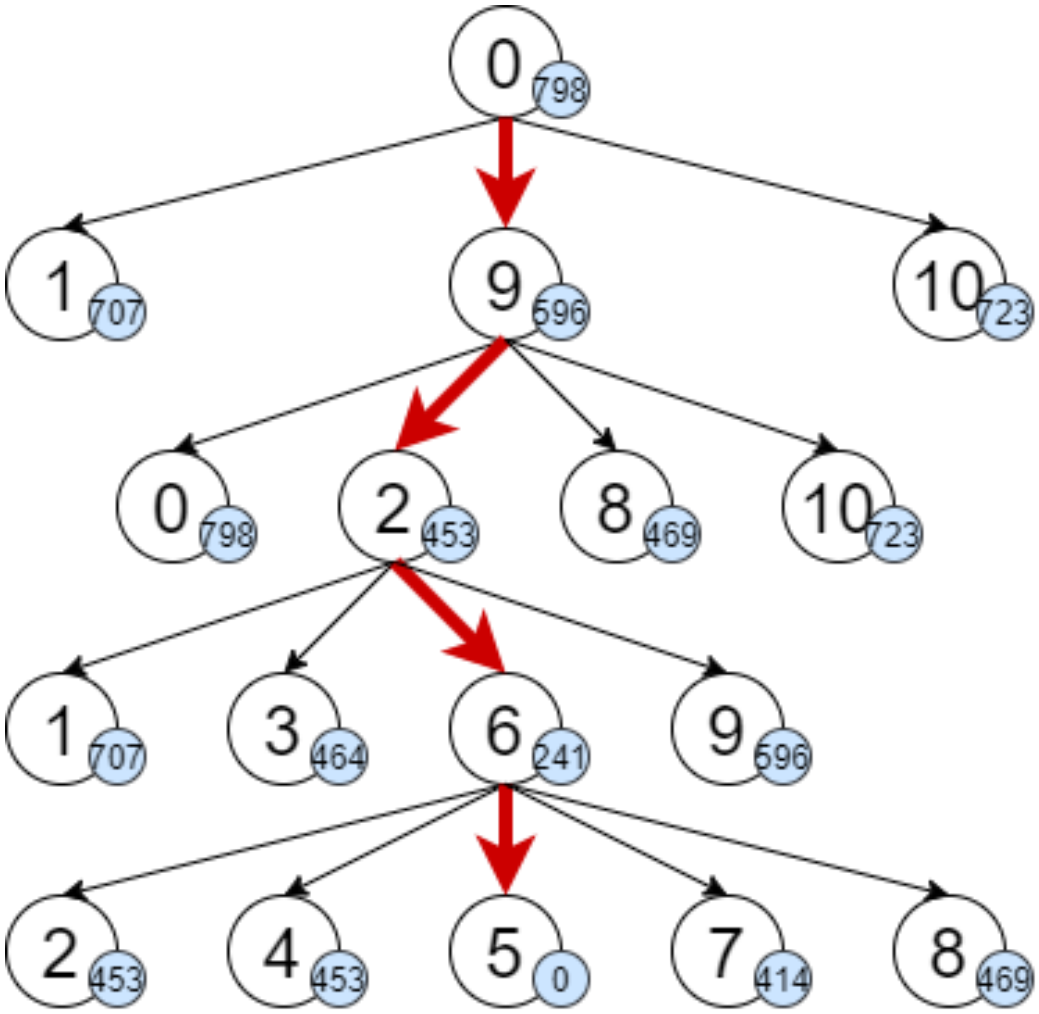
với g(n): khoảng cách giữa các cạnh, h(n) khoảng cách Euclidean từ nút tới đích.

Các bước	Hàng đợi	Mở rộng	Đã xét
1	{(0; 798)}	0	{}
2	{(9; 800), (10; 897), (1; 917)}	9	{(0; 798)}
3	{(8; 871), (10; 897), (2; 900), (1; 917)}	8	{(0; 776), (9; 800)}
4	{(6; 877), (10; 897), (2; 900), (4; 913), (1; 917), (7; 968)}	6	{(0; 776), (9; 800), (8; 871)}
5	{(5; 878), (10; 897), (2; 900), (4; 913), (1; 917), (7; 968)}	**5**	{(0; 776), (9; 800), (8; 871), (6; 877)}

Đã tìm thấy đường đi: **0 – 9 – 8 – 6 – 5**



<u>Thuật toán GBFS</u>: 0 – 9 – 2 – 6 – 5



VI. TÀI LIỆU THAM KHẢO.

Wikipedia:

- https://vi.wikipedia.org/wiki/T%C3%ACm_ki%E1%BA%BFm_theo_chi%E1%BB%81u_r%E1%BB%99ng
- https://vi.wikipedia.org/wiki/T%C3%ACm_ki%E1%BA%BFm_theo_chi%E1%BB%81u_s%C3%A2u
- https://vi.wikipedia.org/wiki/Gi%E1%BA%A3i_thu%E1%BA%ADt_t%C3%ACm_ki%E1%BA%BFm_A*
- https://vi.wikipedia.org/wiki/T%C3%ACm_ki%E1%BA%BFm_chi_ph%C3%AD_%C4%91%E1%BB%81u
- https://vi.wikipedia.org/wiki/T%C3%ACm_ki%E1%BA%BFm_theo_l%E1%BB%B1a_ch%E1%BB%8Dn_t%E1%BB%91t_nh%E1%BA%A5t

Tài liệu học tập:

- https://drive.google.com/file/d/17BRFlW9DNmHAWruK4uJrd1WLVh1qhkPj/view
- https://drive.google.com/file/d/1M0yKa_hDO40RScfmQpnnXD2ZAfyYeDNf/view

Người thực hiện

(ký tên)

Thục

Phạm Đình Thục