**BÁO CÁO ĐỒ ÁN 1**

**TÌM KIẾM**

**Yêu cầu 1**

1. **THÔNG TIN THÀNH VIÊN NHÓM**

|  |  |  |
| --- | --- | --- |
| MSSV | Tên | Email |
| 1512031 | Huỳnh Cao Biên | 1512031@student.hcmus.edu.vn |
| 1512041 | Võ Hoàng Minh Chí | 1512041@student.hcmus.edu.vn |

1. **CÁC GIAI ĐOẠN VÀ MỐC THỜI GIAN THỰC HIỆN ĐỒ ÁN**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Giai đoạn** | **Bắt đầu** | **Kết thúc** | **Địa điểm** | | **Nội dung** | **Tiến độ đồ án** |
| Bắt đầu ra đồ án | 12/10/2017 | 12/10/2017 | * Ở trường * Trực tiếp | | * Họp báo, định hình đồ án, triển khai giải pháp thực hiện và phân chia công việc | 5% |
| 14/10/2017 | 14/10/2017 | * Ở nhà * Facebook * Github | | * Thiết kế teamplate document cho hai báo cáo Yêu cầu 1 và Yêu cầu 2. * Tạo hai tài khoản free trên Github để 2 thành viên làm việc trên cùng một thư mục. * Tạo kiến trúc thư mục nộp đồ án như yêu cầu. | 5% |
| Giữa đồ án | 16/10/2017 | 16/10/2017 | * Ở nhà * Facebook * Github | | * Thiết kế lưu trữ đồ thị * Viết code lưu tru đồ thị bằng C/C++ * Đọc file input.txt và suy nghĩ thuật toán nập đồ thị vào mẫu thiết kế | 10% |
| 17/10/2017 | 17/10/2017 | * Ở nhà * Facebook * Github | | * Cài đặt thuật toán nạp đồ thị từ file input.txt vào mẫu thiết kế đồ thị thành công * Cài đặt các thuật toán DFS,BFS,UCS,GBFS,Astart | 20% |
|  |
| Kết thúc đồ án |  |  |  | |  |  |
| Tổng cộng |  | | |  | | 100% |

1. **PHÂN CÔNG NHIỆM VỤ LÀM ĐỒ ÁN**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Ngày bắt đầu | Ngày kết thúc | Nhiệm vụ | Người thực hiện | Tỉ lệ công việc | Mức độ hoàn thành | Ghi chú |
| 12/10/2017 | 12/10/2017 | * Họp báo, định hình đồ án, triển khai giải pháp thực hiện và phân chia công việc | Huỳnh Cao Biên  Võ, Hoàng Minh Chí | 5% | 100% |  |
| 14/10/2017 | 14/10/2017 | * Thiết kế teamplate document cho hai báo cáo Yêu cầu 1 và Yêu cầu 2. * Tạo hai tài khoản free trên Github để 2 thành viên làm việc trên cùng một thư mục. * Tạo kiến trúc thư mục nộp đồ án như yêu cầu. | Huỳnh Cao Biên,  Võ Hoàng Minh Chí | 10% | 100% |  |
| 16/10/2017 | 16/10/2017 | * Thiết kế lưu trữ đồ thị * Viết code lưu tru đồ thị bằng C/C++ * Đọc file input.txt và suy nghĩ thuật toán nập đồ thị vào mẫu thiết kế | Huỳnh Cao Biên,  Võ Hoàng Minh Chí | 50% | 90% |  |
| 17/10/2017 | 17/10/2017 | * Cài đặt thuật toán nạp đồ thị từ file input.txt vào mẫu thiết kế đồ thị thành công * Cài đặt các thuật toán DFS,BFS,UCS,GBFS,Astart | Huỳnh Cao Biên,  Võ Hoàng Minh Chí | 30% | 90% | Thuật toán ÚC chưa được cài đặt |
| 02/11/2017 | 02/11/2017 | Thiết kế menu Application Console | Huỳnh Cao Biên,  Võ Hoàng Minh Chí | 5% | 100% |  |

1. **MÔ TẢ THIẾT KẾ LƯU TRỮ ĐỒ THỊ**
2. **MÔ TẢ THUẬT TOÁN**

**1. Thuật toán DFS – Tìm kiếm theo chiều sâu:**

– Đây là thuật toán tìm các đỉnh bằng cách duyệt theo chiều sâu.

– Xuất phát từ 1 đỉnh và đi mãi cho đến khi không thể đi tiếp, sau đó đi về lại đỉnh đầu.  
Trong quá trình quay lại:

+ nếu gặp đường đi khác thì đi cho đến khi không đi tiếp được nữa

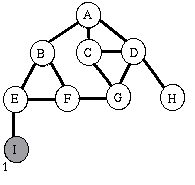
+ nếu không tìm ra đường đi nào khác thì ngừng việc tìm kiếm.

– Trong quá trình đi đến đỉnh khác, thuật toán sẽ lưu lại đỉnh cha vừa đi qua để khi đi ngược lại từ đỉnh Kết thúc đến đỉnh Xuất phát, ta có thể xem được đường đi từ đỉnh Kết thúc đến đỉnh Bắt Đầu (có thể số lần đi không ít nhất, các bạn có thể tham khảo thuật toán [**BFS**](https://lhchuong.wordpress.com/2013/11/08/thuat-toan-bfs-tim-kiem-theo-chieu-ro%CC%A3ng/)).

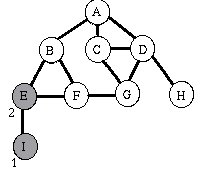
– Sở dĩ thuật toán này tìm được đường đi là nhờ vào cơ chế tô màu và lưu đỉnh cha. Quá trình tô màu khiến 1 đỉnh không thể xét 2 lần trở lên và có thể xem được đường đi từ đỉnh Kết Thúc đến đỉnh Xuất phát dựa vào việc lưu đỉnh cha.

– Sau đây là minh họa về thuật toán:

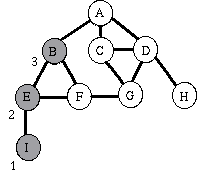
+ ***Hình 1*** đi từ đỉnh bắt đầu, đi cho đến khi không đi được nữa.

**[](https://lhchuong.files.wordpress.com/2013/11/1.png)**

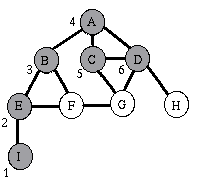
**Hình 1**

[](https://lhchuong.files.wordpress.com/2013/11/21.png)

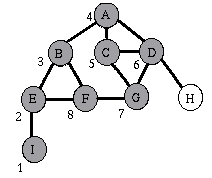
**Hình 2**

[](https://lhchuong.files.wordpress.com/2013/11/3.png)

**Hình 3**

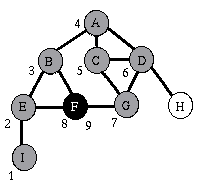
[](https://lhchuong.files.wordpress.com/2013/11/4.png)

**Hình 4**

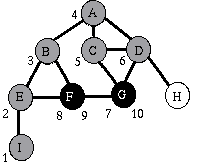
[](https://lhchuong.files.wordpress.com/2013/11/5.png)

**Hình 5**

+ ***Hình 6*** do không đi được nữa nên quay ngược về lại đỉnh bắt đầu.

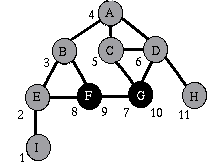
[](https://lhchuong.files.wordpress.com/2013/11/61.png)

**Hình 6**

[](https://lhchuong.files.wordpress.com/2013/11/71.png)

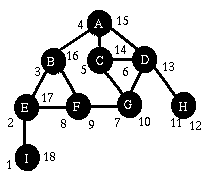
**Hình 7**

+ ***Hình 8*** khi quay lại đến đỉnh D, gặp đỉnh H vẫn chưa được tô màu, tìm được đường đi mới.

[](https://lhchuong.files.wordpress.com/2013/11/8.png)

**Hình 8**

+ ***Hình 9*** sau khi đi qua đỉnh H, không thể đi tiếp được nữa nên tiến hành quay lại đến đỉnh xuất phát.

[](https://lhchuong.files.wordpress.com/2013/11/9.png)

**Hình 9**

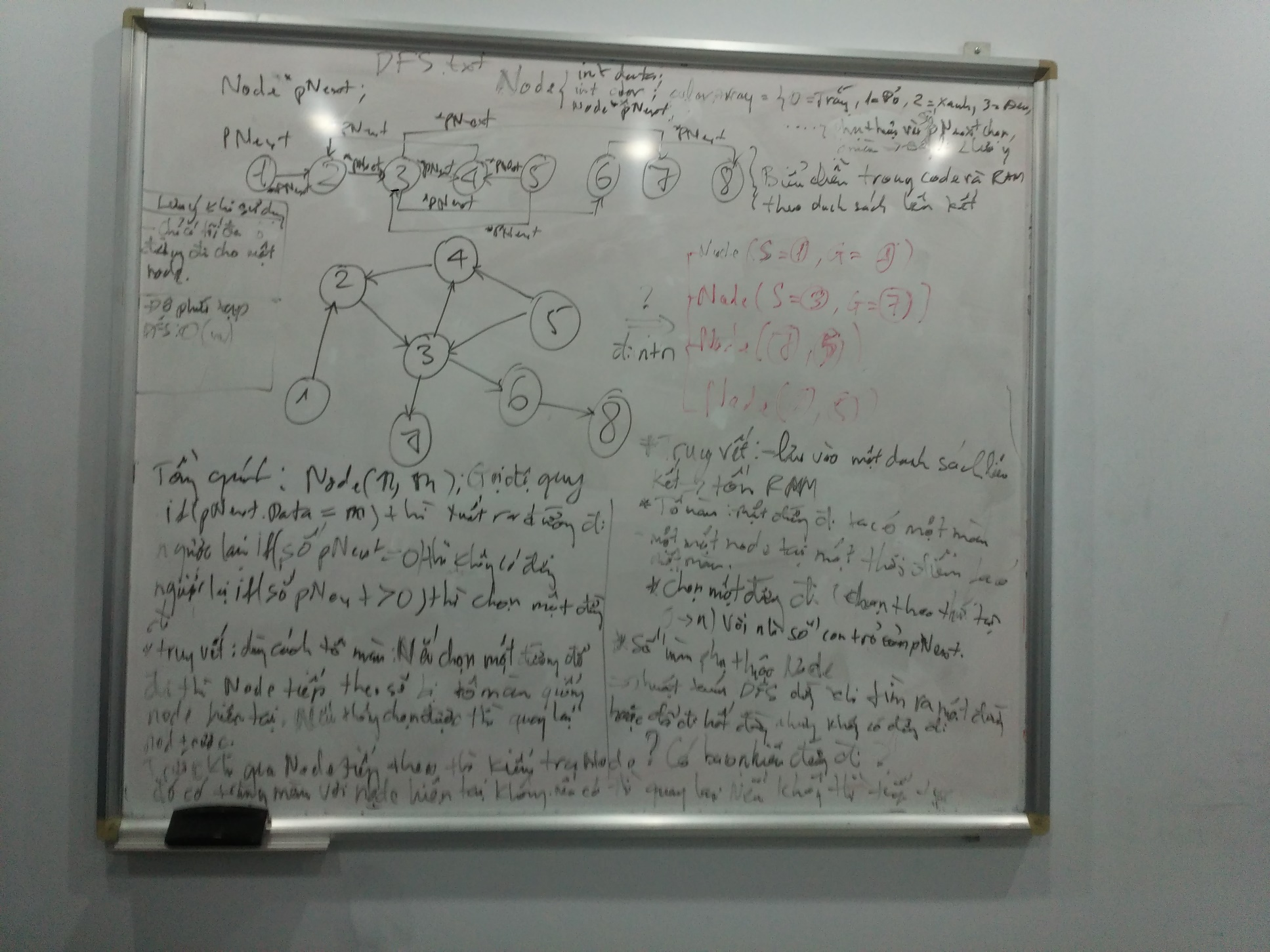


Figure 1 – Tìm lời giải thuật toán tìm kiếm theo chiều sâu

**2. Thuật toán BFS – Tìm kiếm theo chiều rộng**

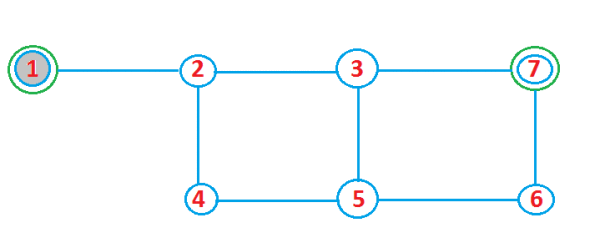
– Đây là thuật toán tìm các đỉnh bằng cách duyệt theo chiều rộng.

– Xuất phát từ 1 đỉnh và đi tới các đỉnh kề nó, tiếp tục cho đến khi không còn đỉnh nào có thể đi.

– Trong quá trình đi đến đỉnh kề, tiến hành lưu lại đỉnh cha của đỉnh kề để khi đi ngược lại từ đỉnh Kết thúc đến đỉnh Xuất phát, ta có được đường đi ngắn nhất.

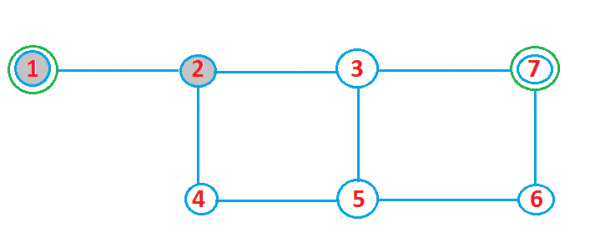
– Sở dĩ thuật toán này tìm được đường đi ngắn nhất là nhờ vào cơ chế tô màu và lưu đỉnh cha. Quá trình tô màu khiến 1 đỉnh không thể xét 2 lần trở lên và có thể xem được đường đi từ đỉnh Kết Thúc đến đỉnh Xuất phát dựa vào việc lưu đỉnh cha.

– Sau đây là minh họa về thuật toán:

+ ***Hình 1*** : Xuất phát từ đỉnh 1[](https://lhchuong.files.wordpress.com/2013/11/untitled.png)

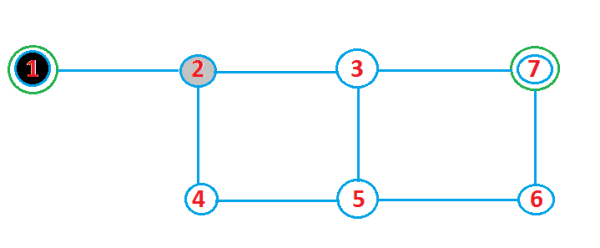
**Hình 1**

 + ***Hình 2*** : Đi đến đỉnh 2, như vậy nút 1 là nút cha của nút 2

[](https://lhchuong.files.wordpress.com/2013/11/untitled1.png)

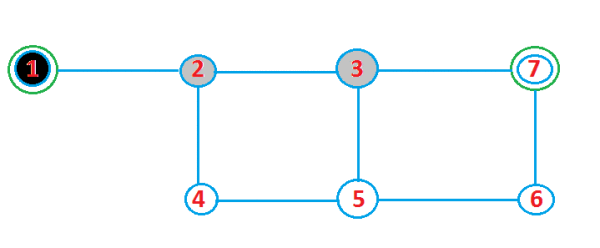
**Hình 2**

 + ***Hình 3*** : Đã đi hết tất cả các đỉnh kề của đỉnh 1, tiến hành bôi đen đỉnh 1

[](https://lhchuong.files.wordpress.com/2013/11/untitled2.png)

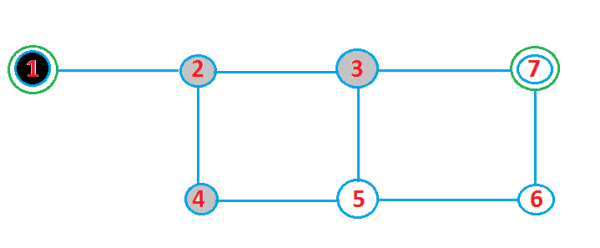
**Hình 3**

 + ***Hình 4***: Xuất phát từ đỉnh 2, chọn đỉnh 3, nút cha của đỉnh 3 là đỉnh 2

[](https://lhchuong.files.wordpress.com/2013/11/untitled3.png)

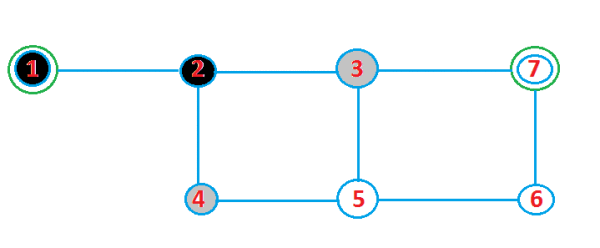
**Hình 4**

 + ***Hình 5*** : Xuất phát từ đỉnh 2, bôi đen đỉnh 4, nút cha của đỉnh 4 là đỉnh 2

[](https://lhchuong.files.wordpress.com/2013/11/untitled4.png)

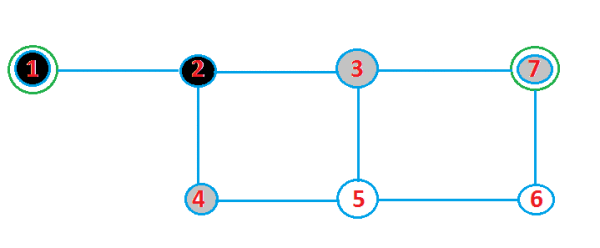
**Hình 5**

 + ***Hình 6*** : Đã đi hết tất cả các đỉnh kề của đỉnh 2, tiến hành bôi đen đỉnh 2

[](https://lhchuong.files.wordpress.com/2013/11/untitled5.png)

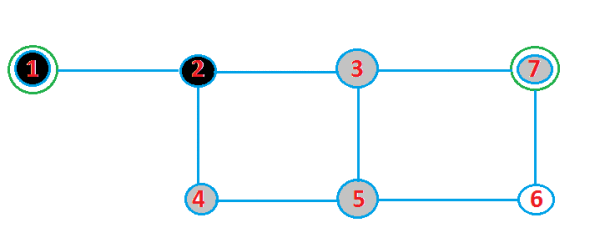
**Hình 6**

 + ***Hình 7*** : Xuất phát tử đỉnh 3, đi đến đỉnh 7, như vậy đỉnh 3 là đỉnh cha của đỉnh 7

[](https://lhchuong.files.wordpress.com/2013/11/untitled7.png)

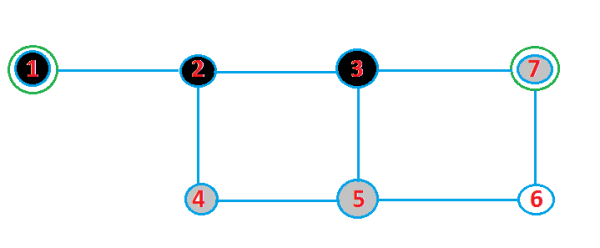
**Hình 7**

 + ***Hình 8*** : Xuất phát từ đỉnh 3, đi đến đỉnh 5, như vậy đỉnh 3 là đỉnh cha của đỉnh 5

[](https://lhchuong.files.wordpress.com/2013/11/untitled8.png)

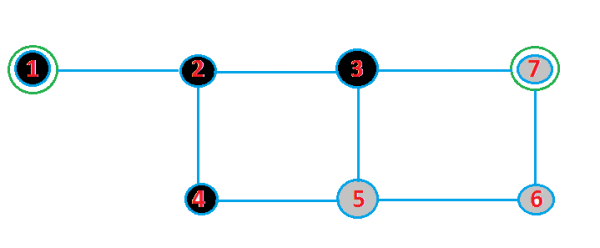
**Hình 8**

 + ***Hình 3*** : Đã đi hết tất cả các đỉnh kề của đỉnh 3, tiến hành bôi đen đỉnh 3

[](https://lhchuong.files.wordpress.com/2013/11/untitled9.png)

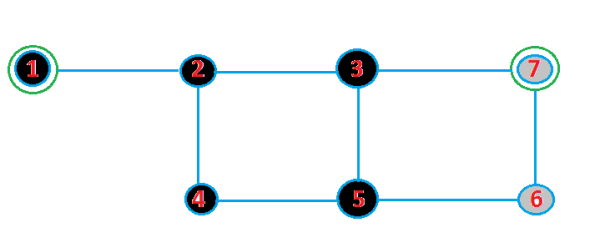
**Hình 9**

 + ***Hình 10*** : Xuất phát từ đỉnh 5, đi đến đỉnh 6, như vậy đỉnh 5 là đỉnh cha của đỉnh 6

[](https://lhchuong.files.wordpress.com/2013/11/untitled11.png)

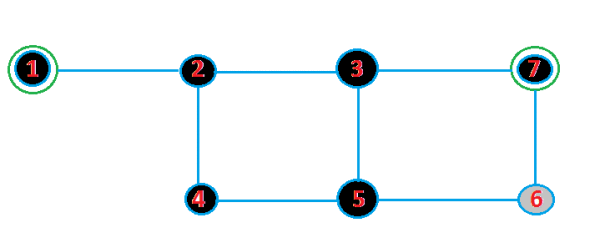
**Hình 10**

 + ***Hình 11***: Đã đi hết tất cả các đỉnh kề của đỉnh 5, tiến hành bôi đen đỉnh 5

[](https://lhchuong.files.wordpress.com/2013/11/untitled12.png)

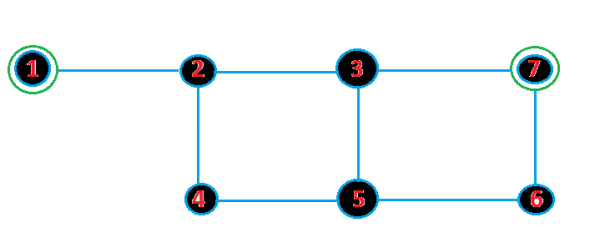
**Hình 11**

 + ***Hình 12***: Đã đi hết tất cả các đỉnh kề của đỉnh 7, tiến hành bôi đen đỉnh 7

[](https://lhchuong.files.wordpress.com/2013/11/untitled13.png)

**Hình 12**

 + ***Hình 13*** : Đã đi hết tất cả các đỉnh kề của đỉnh6, tiến hành bôi đen đỉnh 6

[](https://lhchuong.files.wordpress.com/2013/11/untitled14.png)

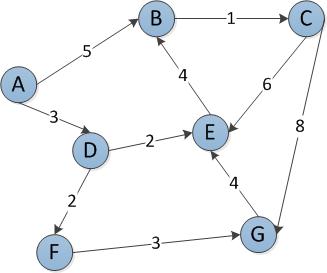
**Hình 13**

 – Như vậy ta vừa đi hết tất cả các đỉnh trong đồ thị, và mỗi lần đi đến đỉnh mới, ta đều lưu lại nút cha của đỉnh mới, dựa vào những đỉnh cha này, ta có liệt kê đường đi ngắn nhất bằng cách đi ngược từ đỉnh Kết thúc, đến đỉnh cha của đỉnh Kết thúc … rồi đến đỉnh cha của đỉnh tiếp theo … đến đỉnh Bắt đầu.

**3. Thuật toán UCS – Tìm kiếm chi phí đồng nhất**

**Mở rộng các tập đã xét và hàng đợi ưu tiên:**  
Nút bắt đầu: A  
Nút kết thúc: G

|  |  |  |  |
| --- | --- | --- | --- |
| **Các bước** | **Biên giới: tập các đối tượng (nút và chi phí)** | **Mở rộng**[[\*]](https://vi.wikipedia.org/wiki/T%C3%ACm_ki%E1%BA%BFm_chi_ph%C3%AD_%C4%91%E1%BB%81u#endnote_expand.5E_.2A) | **Đã xét: tập các nút đã xét** |
| 1 | {(A,0)} | A | ∅ |
| 2 | {(D,3),(B,5)} | D | {A} |
| 3 | {(B,5),(E,5),(F,5)} | B | {A,D} |
| 4 | {(E,5),(F,5),(C,6)} | E | {A,D,B} |
| 5 | {(F,5),(C,6)}[[\*\*]](https://vi.wikipedia.org/wiki/T%C3%ACm_ki%E1%BA%BFm_chi_ph%C3%AD_%C4%91%E1%BB%81u#endnote_B_not_in_frontier.5E_.2A.2A) | F | {A,D,B,E} |
| 6 | {(C,6),(G,8)} | C | {A,D,B,E,F} |
| 7 | {(G,8)} | G | {A,D,B,E,F,C} |
| 8 | ∅ |  |  |



Nút được chọn để duyệt cho bước tiếp theo.

B không được thêm vào hàng đợi vì nó đã nằm trong tập đã xét.  
Đường đi được tìm thấy: A -> D -> F -> G. Ở bước số 3, các nút B, E, F đều có chi phí là 5, trường hợp này có thể chọn 1 nút bất kỳ hoặc ưu tiên theo thứ tự các nút (chẳng hạn như theo thứ tự chữ cái B, E, F thì chọn B).

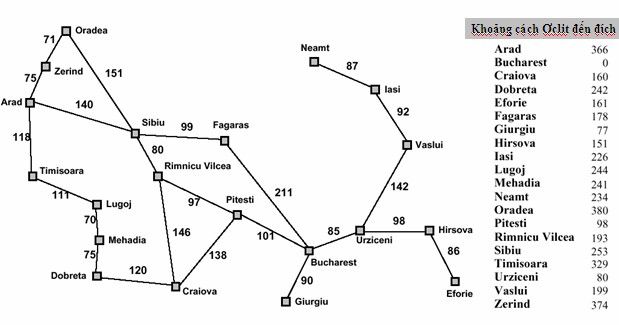
## Các biến thể của giải thuật best first search

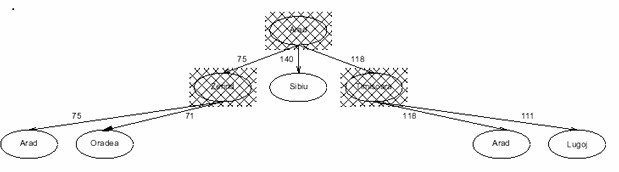
Ý tưởng của giải thuật tìm kiếm tốt nhất đầu tiên (best first search) là mở rộng cây tìm kiếm theo hướng ưu tiên các nút lá có triển vọng chứa trạng thái đích (dựa trên hàm đánh giá h). Giải thuật best-first-search có các biến thể sau:

* Khi hàm h(n) là chi phí của dãy phép chuyển từ trạng thái đầu đến trạng thái n thì giải thuật best-first-search có tên gọi khác là giải thuật tìm kiếm đều (uniform search). Trong

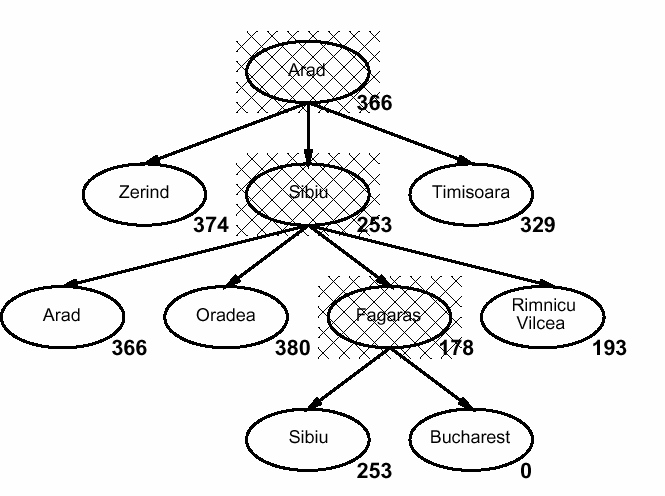
trường hợp này, cây tìm kiếm sẽ mở rộng đều về tất cả các hướng theo vết dầu loang từ trạng thái đầu. Khi hàm chi phí của dãy phép chuyển là số các đỉnh trung gian thì giải thuật uniform search trở thành giải thuật tìm kiếm theo chiều rộng. Giải thuật uniform search sẽ cho lời giải với chi phí nhỏ nhất, tuy nhiên cây tìm kiếm sinh ra trong giải thuật này thường có kích thước rất lớn.

* Khi h(n) là ước lượng chi phí/khoảng cách từ n đến đích (ví dụ như khoảng cách Manhatan trong bài toán 8 số ở trên) thì giải thuật best-first-search được gọi là giải thuật tham ăn (greedy search). Giải thuật tham ăn sẽ chọn nút lá n “gần” đến đích nhất trong số các nút lá của cây tìm kiếm để mở rộng cây, và nó không quan tâm đến chi phí từ trạng thái đầu đến n. Do vậy giải thuật có xu hướng cho ra kết quả trong thời gian nhanh nhất, nhưng không phải lúc nào cũng là lời giải ngắn nhất.
* Khi h(n) = f(n) + g(n), trong đó f(n) là hàm chi phí/khoảng cách từ trạng thái đầu đến n và g(n) là hàm ước lượng chi phí/khoảng cách từ n đến trạng thái đích, và nếu g(n) là ước lượng dưới của hàm chi phí/khoảng cách thực sự từ n đến trạng thái đích thì giải thuật best-first-search được gọi là giải thuật A\*. Giải thuật A\* là giải thuật trung hòa giữa hai giải thuật uniform và giải thuật greedy ở trên. A\* cho lời giải có chi phí nhỏ nhất (bạn đọc có thể tìm hiểu chứng minh điều này ở các tài liệu khác) và cây tìm kiếm có kích thước vừa phải.

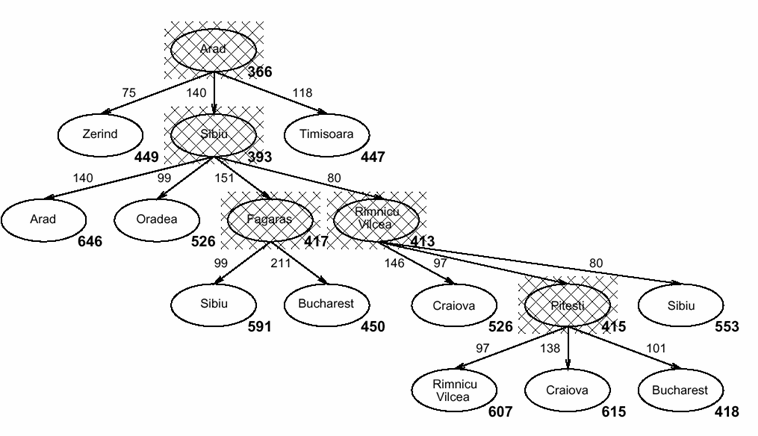


Ví dụ, đối với bài toán tìm đường đi từ thành phố Arad đến thành phố Bucharest đã mô tả trong 1.b, nếu chúng ta sử dụng khoảng cách Ơclit (khoảng cách theo đường chim bay) từ mỗi thành phố đến đích (xem hình vẽ trên) thì các giải thuật uniform, greedy và A\* sẽ cho các cây tìm kiếm như sau:

Một phần cây tìm kiếm của giải thuật Uniform search



Cây tìm kiếm của giải thuật Greedy search



Cây tìm kiếm của giải thuật A\*

1. **HƯỚNG DẪN CHẠY CHƯƠNG TRÌNH THÀNH CÔNG**
2. **NHỮNG TÀI LIỆU VÀ LINK THAM KHẢO**
3. <https://lhchuong.wordpress.com/2013/11/24/thua%CC%A3t-toan-dfs-tim-kiem-theo-chieu-sau/>
4. <https://lhchuong.wordpress.com/2013/11/08/thuat-toan-bfs-tim-kiem-theo-chieu-ro%CC%A3ng/>
5. <https://vi.wikipedia.org/wiki/T%C3%ACm_ki%E1%BA%BFm_chi_ph%C3%AD_%C4%91%E1%BB%81u>
6. <http://ccs1.hnue.edu.vn/hoanpt/AI/Tailieu-TriTueNhanTao_new1.pdf>