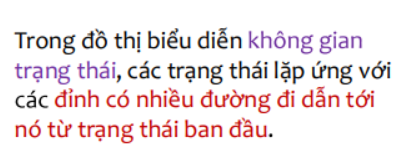
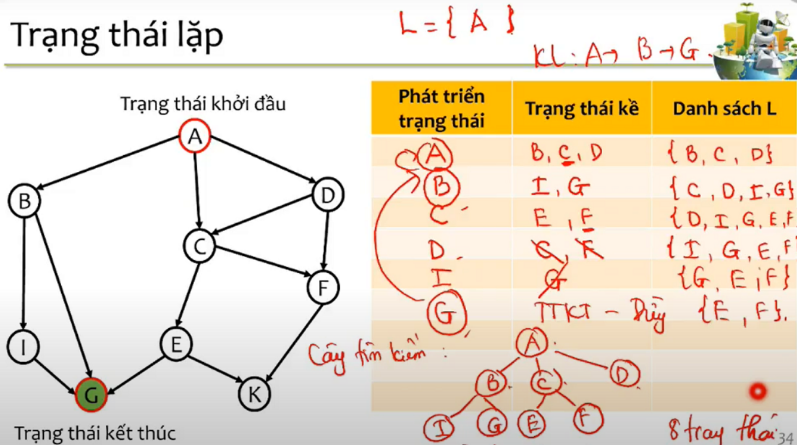
**Các thuật toán tìm kiếm**

**Tìm kiếm mù:**

* Tìm kiếm theo chiều rộng(BFS): Xét các đỉnh kề với đỉnh đang xét sau đó cho vào cuối danh sách L, loại đỉnh đang xét khỏi ds L. Duyệt đến hết cây
* Tìm kiếm theo chiều sâu(DFS): Xét các đỉnh kề với đỉnh đang xét sau đó cho vào đầu danh sách L, loại đỉnh đang xét khỏi ds L. Duyệt đến hết cây

Nhận xét: Trạng thái sinh ra nhiều, độ phức tạp cao.

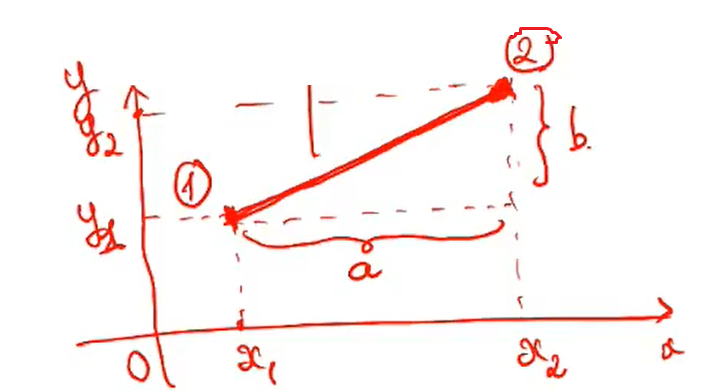
**Trạng thái lặp:**

* Gây lãng phí thời gian, nếu đồ thị có chu trình thì quá trình tìm kiếm sẽ không dừng.
* **Cách loại bỏ trạng thái lặp:** không xét các trạng thái đã phát triển

**Hàm đánh giá(*heuristic function*):** ước lượng khoảng cách từ trạng thái đang xét với trạng thái kết thúc(khác với trộng số) càng gần trạng thái kết thúc(= 0) thì giá trị hàm càng nhỏ. Vd:

* khoảng đường chim bay giữa 2 node
* tổng của những ô sai vị trí

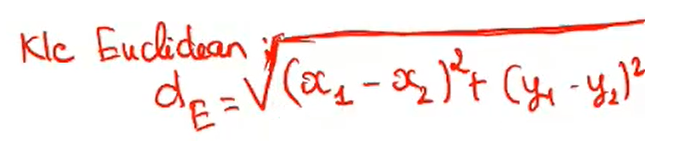
Các hàm đánh giá:

Xét trên hệ trục tọa độ:

* Manhattan distance: (robot đi 4 hướng)

**dM = a + b = |x1 – x2| + |y1 – y2|**

* Euclidean distance: (robot đi 8 hướng)



Hàm đánh giá của tìm kiếm tối ưu:

f(u) = g(u) + h(u)

đường đi ngắn nhất đến u và khoảng cách ước lượng từ u đến đích

**Tìm kiếm kinh nghiệm**: giảm các bước phát triển trạng thái -> giảm đồ phức tạp thuật toán

* TT Best first search:
* TT Hill Climbing search
* Đánh giá tìm kiếm tốt nhất **Best first search**:

tìm đường đi mà sinh ra số lượng trạng thái sinh ra là ít nhất

khác với tìm kiếm bth ds L sắp xếp theo tăng dần theo giá trị hàm đánh giá( h(x) )

* phát triển từ BFS
* Đánh giá leo đồi(hill climbing search)

tương tự như best first cũng sắp xếp theo hàm đánh giá nhưng sắp xếp theo từng đoạn(các đỉnh kề vs đỉnh đang xét) rồi đưa L1 lên đầu ds L

* phát triển từ DFS

**Tìm kiếm tối ưu:**

Sử dụng hàm đánh giá f(u):

f(u) = g(u) + h(u)

đường đi ngắn nhất đến u và khoảng cách ước lượng từ u đến đích

Đối với bài toán tìm đường đi của robot: nếu đi chéo thì g = g + 1.4(căn 2)

tìm kiếm A\*: dựa vào hàm đánh giá f(u) tuy nhiên tập L sẽ được sắp xếp tăng dần

tìm kiếm nhánh và cận: cx dựa vào hàm đánh giá f(u) tuy nhiên tập L1 sẽ được sắp xếp rồi chồng lên L cũ

Note: nếu gặp trạng thái kết thúc cập nhật cost và tiếp tục làm tiếp.

Đến khi ds rỗng

Note: nếu xét tiếp mà có trạng thái lặp(đã có trog ds L) nếu trạng thái lặp có cost lớn hơn trạng thái đã có thì t có thể bỏ không thêm vào ds L