- Hàm Heuristic được sử dụng để tính giá trị của h(n).

- Giá trị của hàm Heuristic tại trạng thái mục tiêu là 0. Giá trị hàm Heuristic càng nhỏ thì càng đến gần trạng thái mục tiêu.

- Hàm Heuristic được tính bằng công thức:

h(n) = different\_Color\_point + 5\*bonus\_Stack\_used

+ Trong đó biến "different\_Color\_point" được tính bằng cách xác định các cặp màu nằm cạnh nhau trong mỗi ống. Nếu 2 màu nằm cạnh nhau là khác nhau thì "different\_Color\_point = different\_Color\_point + 500" có nghĩa là tăng biến "different\_Color\_point" lên thêm 500 đơn vị. Mục đích của biến này dùng để đánh giá độ tốt của một trạng thái.

+ Biến "bonus\_Stack\_used" được sử dụng để tính số lượng Stack mà ta đang sử dụng nằm trong số "bonusStack". Mục đích của "bonus\_Stack\_used" là dùng để rút gọn số lượng ống được sử dụng trong 1 trạng thái là nhỏ nhất có thể để nhanh đi đến trạng thái đích

+ Giá trị của "different\_Color\_point" tăng nhanh hơn "bonus\_Stack\_used" 100 lần bởi vì ta đang ưu tiên giải quyết vấn đề khác nhau về màu sắc hơn là vấn đề sử dụng số ống ít nhất. Lí do chọn mức tăng 500 cho biến "different\_Color\_point" là bởi vì trong hàm f(n) = g(n) + h(n) với g(n) là số bước đã đi được thì trong các trường hợp phức tạp số bước cần đi để giải bài toán có thể > 100 bước. Trong khi đó ta cần biến "different\_Color\_point" phải có độ ưu tiên cao nhất trong hàm f(n) bởi vì nó là yếu tố quyết định cần để xét hàng đợi "open\_set" để nhanh chóng đi đến lời giải, làm giảm thời gian và số lượng trạng thái cần xét. Vì vậy "different\_Color\_point" cần phải có trọng số cao nhất.

Để áp dụng giải thuật A\* vào bài toán. Ta thực hiện các bước chính sau:

+ Chuyển đổi bài toán thành dạng đồ thị tìm đường đi bằng cách coi các trạng thái là các node của đồ thị.

+ Định nghĩa thông tin mỗi node bao gồm : trạng thái hiện tại, số bước đã đi qua, source stack, destination stack.

+ Viết hàm "get\_neighbors" để xác định một tập các nút tiếp theo liền kề cần xét. Do đây là đồ thị có chu trình nên có thể xảy ra tình trạng lặp lại các trạng thái đã xét, do đó ta thêm vào "visited" list để lưu tất cả trạng thái đã đi qua.

+ Cuối cùng là áp dụng code theo đoạn mã giả ở hình 1 với một số thay đổi nhỏ để đi đến lời giải của bài toán