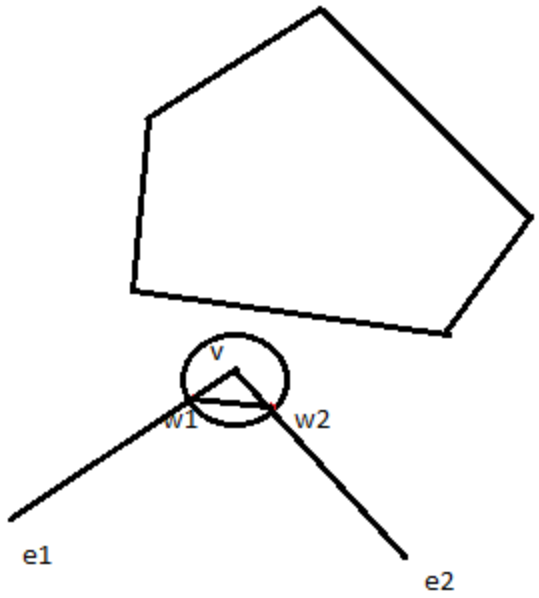
I. Trả lời câu hỏi :

a) Có vô hạn trạng thái (x,y) với x,y ϵ Q. Do đó có vô số đường đi từ đỉnh xuất phát đến đỉnh đích.

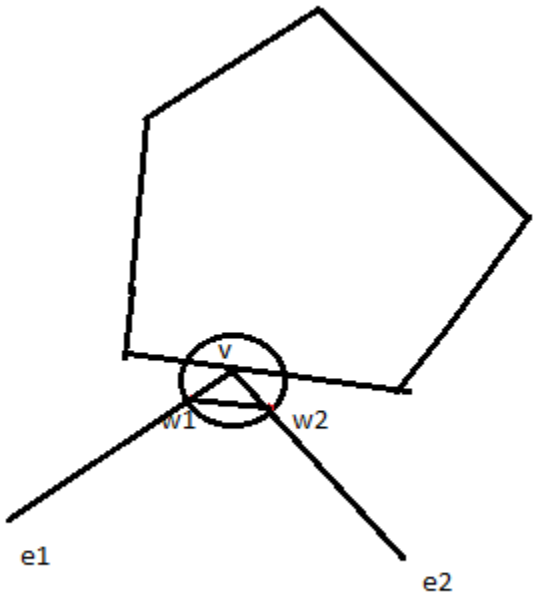
b) Đặt tập đỉnh của đồ thị là V(P) và tập cạnh của đồ thị là E(P). Giả sử có đường đi ngắn nhất t từ S đến G với số đỉnh không thuộc V(P) là nhỏ nhất. Nếu không tồn tại đỉnh nào thì bài toán đã được chứng minh. Ngược lại, tồn tại v ϵ V(t)\{V(P)∪{S,G}}.v có thể nằm trên cạnh của các đa giác hoặc không. Gọi $e_1$ ,$e_2$ ϵ E(t) là 2 cạnh kề với v. Nếu $e_1$ ,$e_2$ thẳng hàng thì ta có thể xóa v ra khỏi t, trái với giả thuyết là số đỉnh không thuộc V(P) là nhỏ nhất. Từ v ,vẽ đường tròn tâm v bán kính ϵ cắt $e_1$ ,$e_2$ tại $w_1$ ,$w_2$ .Vì e1,e2 không thẳng hàng nên $w_1w_2$ ngắn hơn $w_1vw_2$ , do đó tồn tại đường đi ngắn hơn từ S sang G ( trái với giả thuyết t là đường đi ngắn nhất).Trường hợp v nằm trên cạnh của P nhưng không là đỉnh của P chứng minh tương tự. Vậy v ϵ V(t) ∩V(P) hay đường đi ngắn nhất từ S đến G phải chứa các đỉnh của các đa giác.

Không gian trạng thái lúc này là (x,y) với (x,y) ϵ V(P). Không gian trạng thái này tổng số đỉnh của các đa giác

Trường hợp v không nằm trên cạnh các đa giác:




Trường hợp v thuộc cạnh của đa giác:




c) Các hàm cần thiết để thực thi bài toán tìm kiếm :

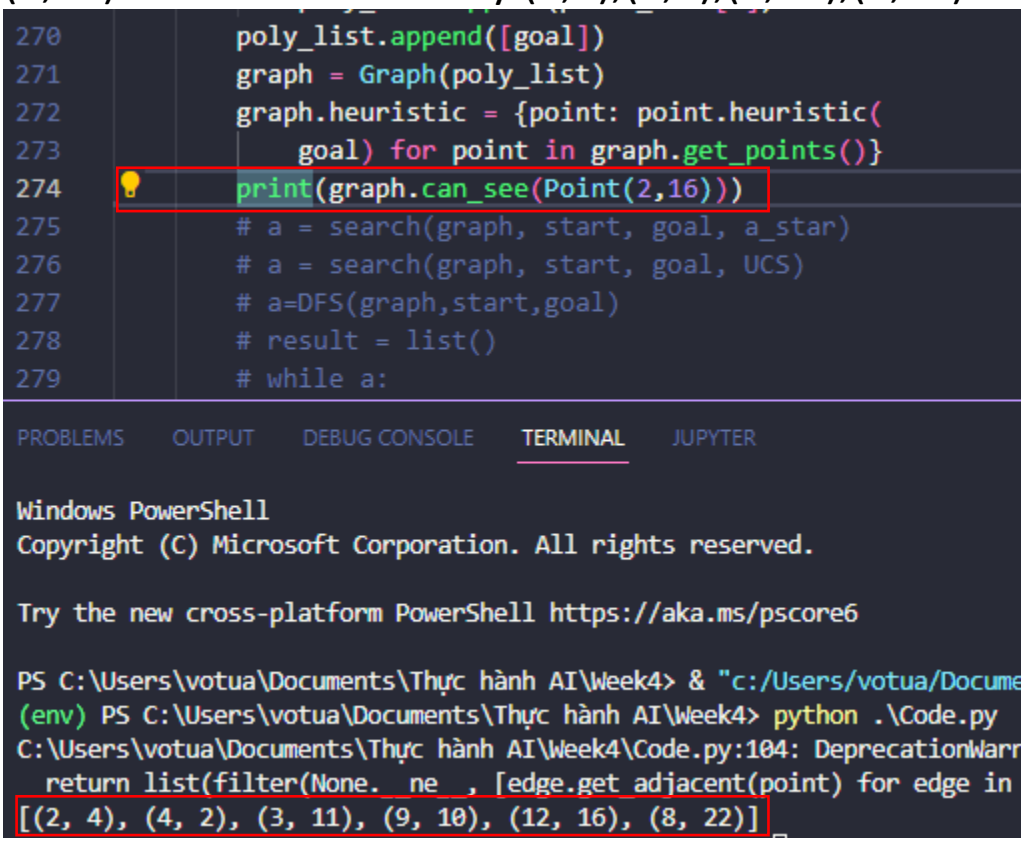
- Hàm successor chính là hàm can_see

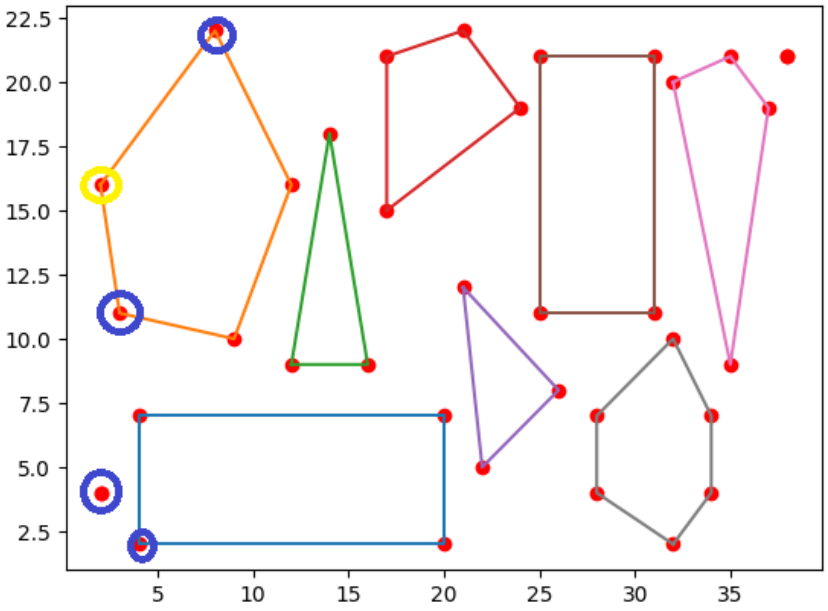
can_see(node) trả về các đỉnh có thể được nhìn thấy từ node, do đó từ node có thể đi các đỉnh đó bằng một bước.

- Hàm eclid_distance(point1,point2) trả về khoảng cách eclid giữa 2 điểm point1 và point2.
- Hàm h(point) chính là hàm heuristic, trả về khoảng cách eclid từ point đến goal.

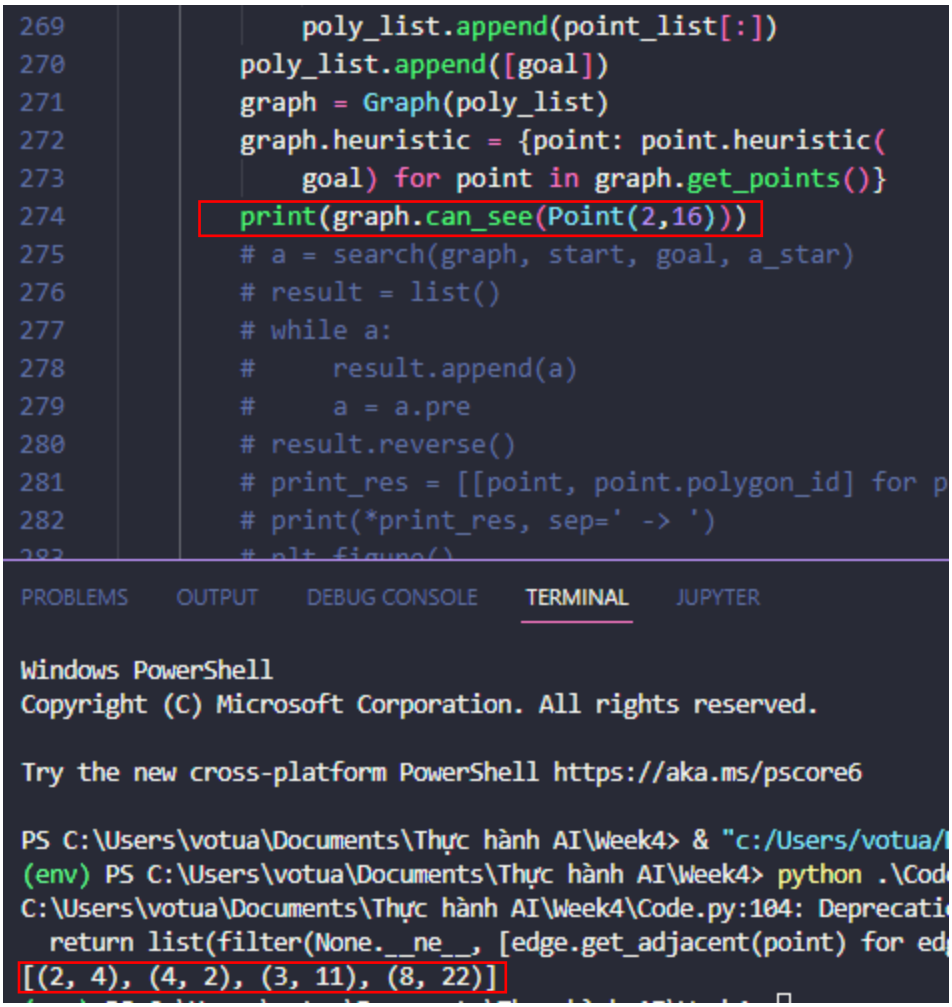
II. Nhận xét code

Hàm can_see bị sai. Ở đây in thử can_see của điểm (2,16), kết quả cho thấy từ (2,16) có thể nhìn thấy các điểm (2,4),(4,2),(3,11),(9,10),(12,16),(8,22) trong khi thực tế từ (2,16) chỉ có thể nhìn thấy (2,4),(4,2),(3,11),(8,22).



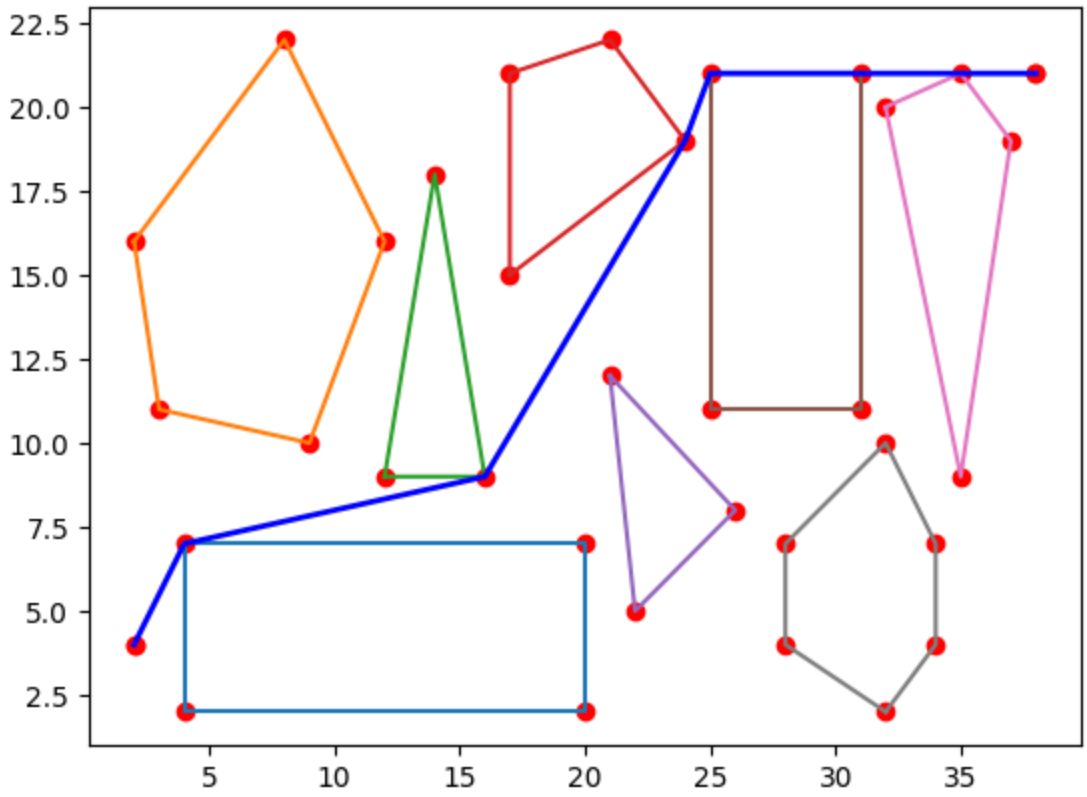



Điểm khoanh vàng là (2,16), các điểm khoanh xanh là các điểm mà từ vàng có thể nhìn thấy được.
Hàm can_see sửa lại được trình bày trong code. Kết quả sau khi sửa hàm can_see:

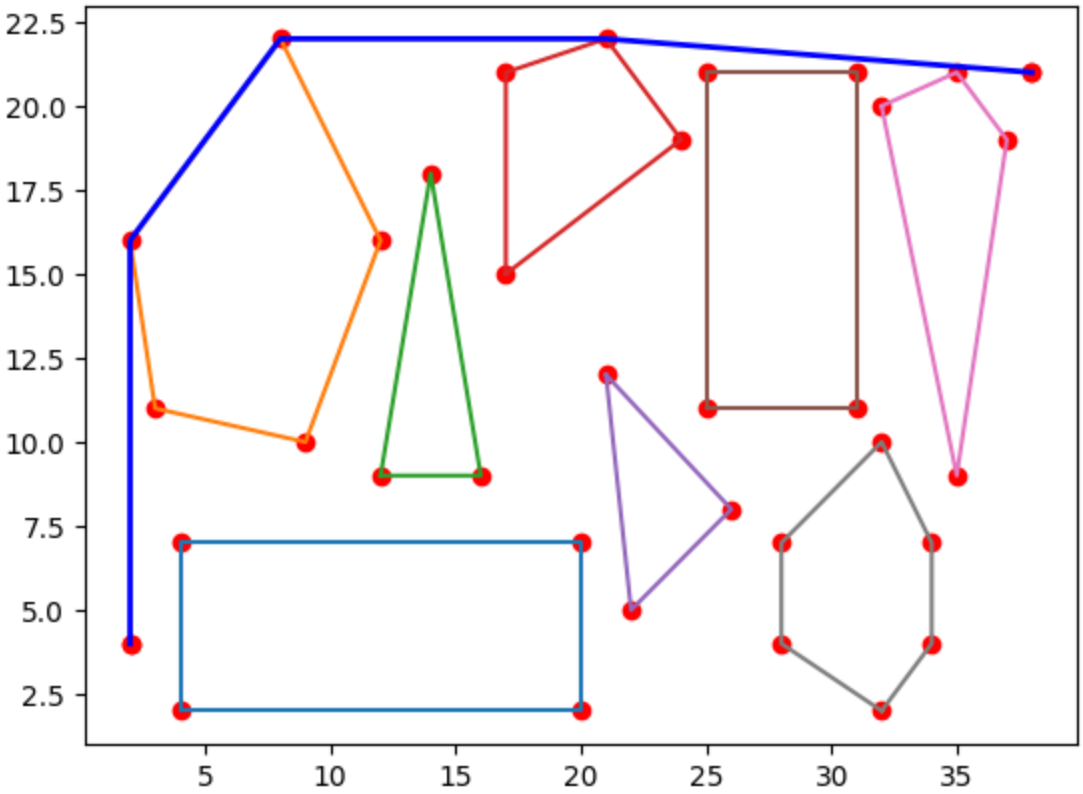



Các hàm BFS, DFS, UCS được trình bày trong file code.
Kết quả tìm đường đi theo các thuật toán :

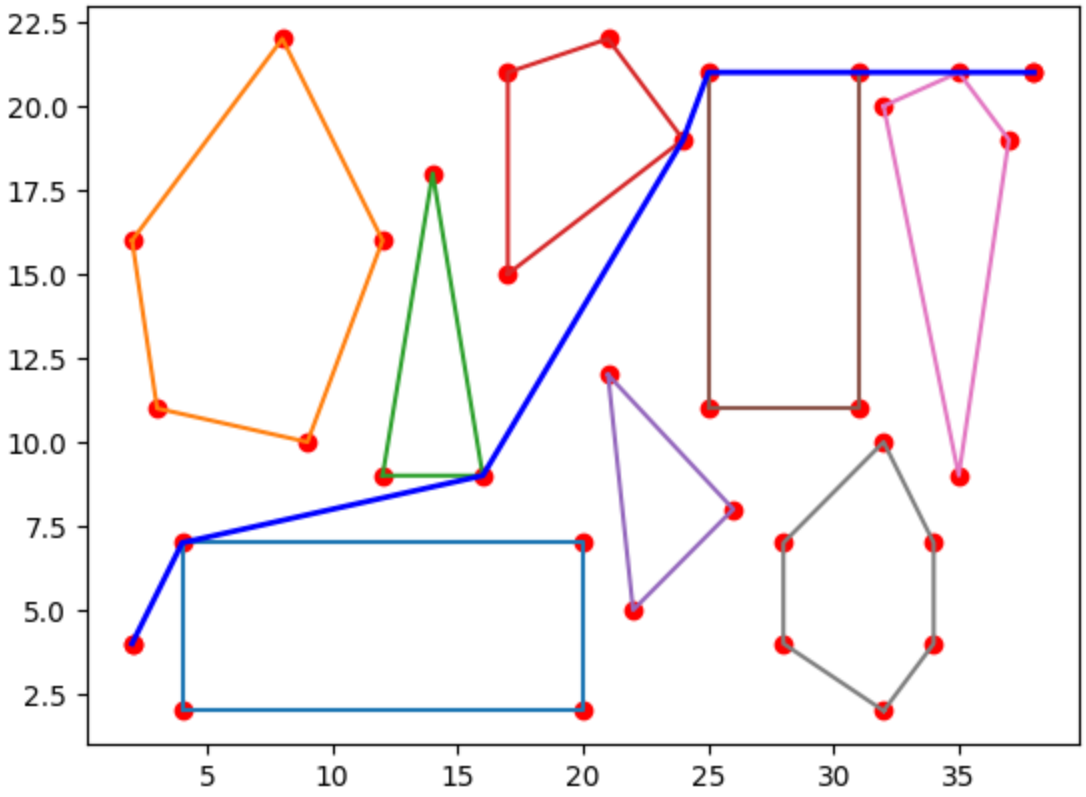
Astart :



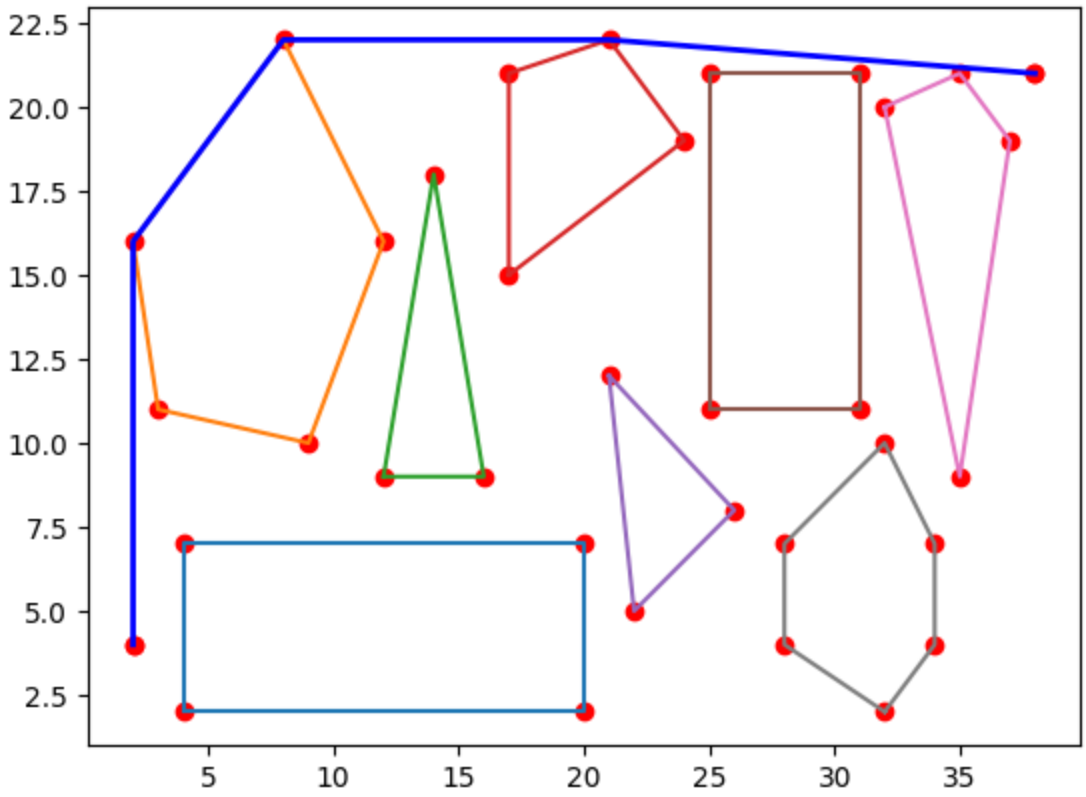

Greedy-Best-First Search:



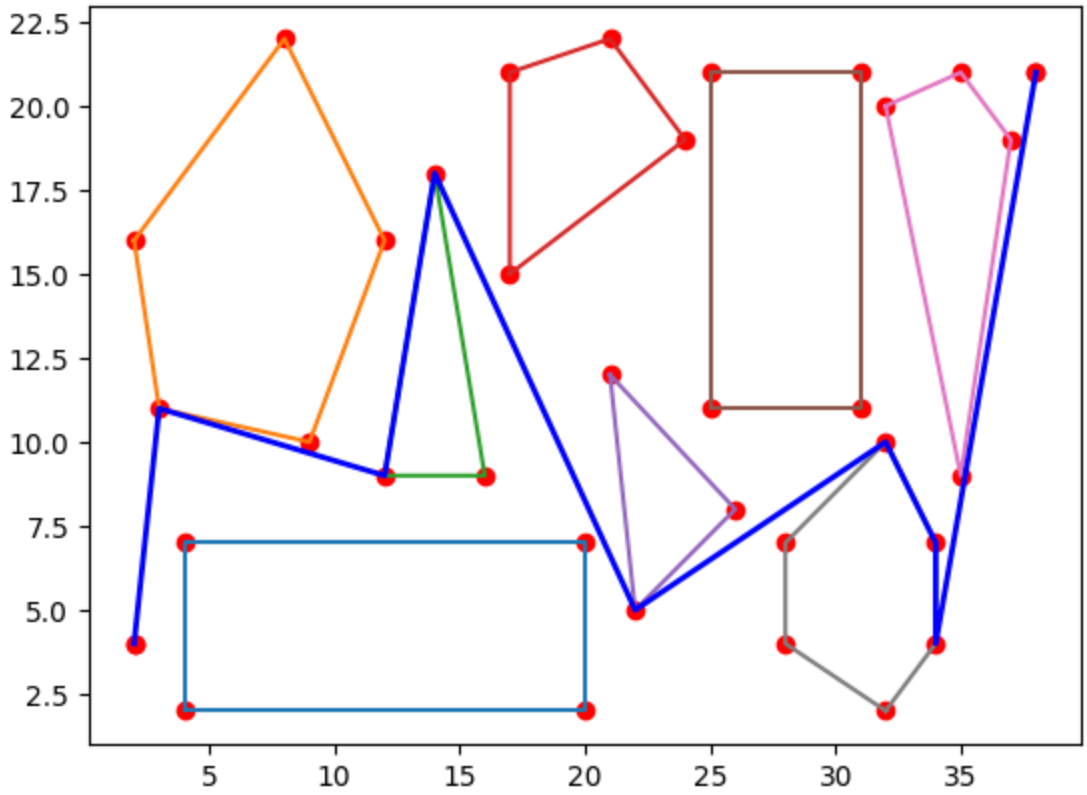

Uniform Cost Search:




Breadth-First Search:




Depth-First Search:




III. Nhận xét các thuật toán tìm kiếm

- Các thuật toán tìm kiếm có thông tin về goal : Astar,GBFS
- Các thuật toán tìm kiếm không có thông tin về goal : UCS,BFS,DFS
- Có thể xem Astar là thuật toán tìm kiếm tổng quát với f(n) =g(n)+h(n), khi g(n)=0 thì Astar trở thành GBFS, khi h(n)=0 thì Astar trở thành UCS, và khi mọi step costs đều bằng nhau thì UCS trở thành BFS.
- Hàm can_see là hàm successor đóng vai trò rất quan trọng, các thuật toán chỉ có thể chạy đúng nếu hàm can_see được cài đặt đúng