Báo cáo đồ án

Project 1

Khoa Công nghệ Thông tin
Đại học Khoa học Tự nhiên TP HCM

Nguyễn Văn Hậu –20127493

MỤC LỤC

Nội dung

1 Tổng quan

1.1 Thông tin sinh viên:

MSSV	Họ tên
20127493	**Nguyễn Văn Hậu**

Đánh giá mức độ hoàn thành: 100%

1.2 Thông tin đồ án:

Trong đồ án này sinh viên sẽ tiến hành nghiên cứu và thực hiện thuật toán
Có 5 thuật toán cần thực hiện:

- Breadth-first search
- Uniform-cost search
- Iterative deepening search
- Greedy-best first search
- Graph-search A*

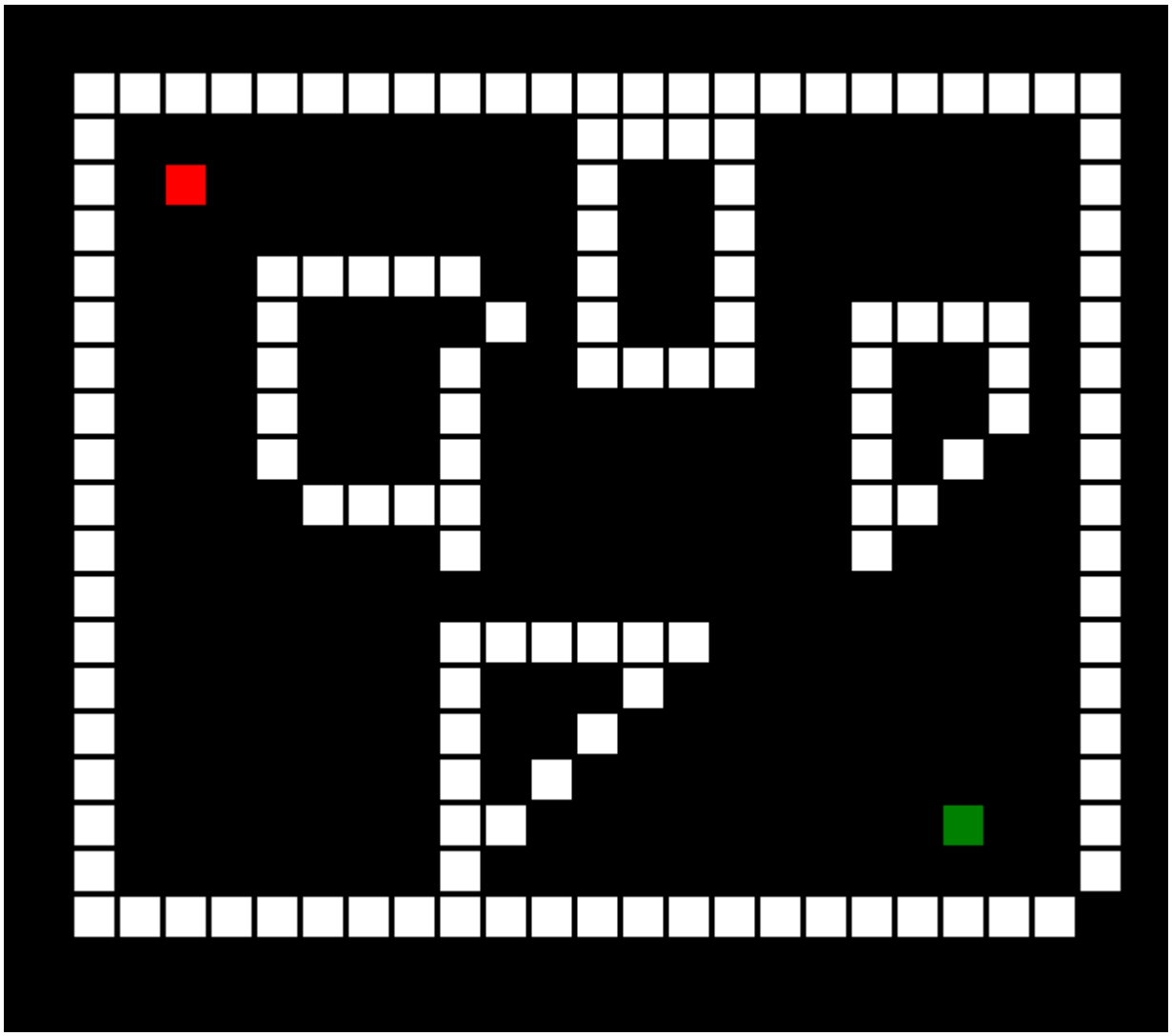
2 Mô tả về thuật toán

1. Breadth-first search:

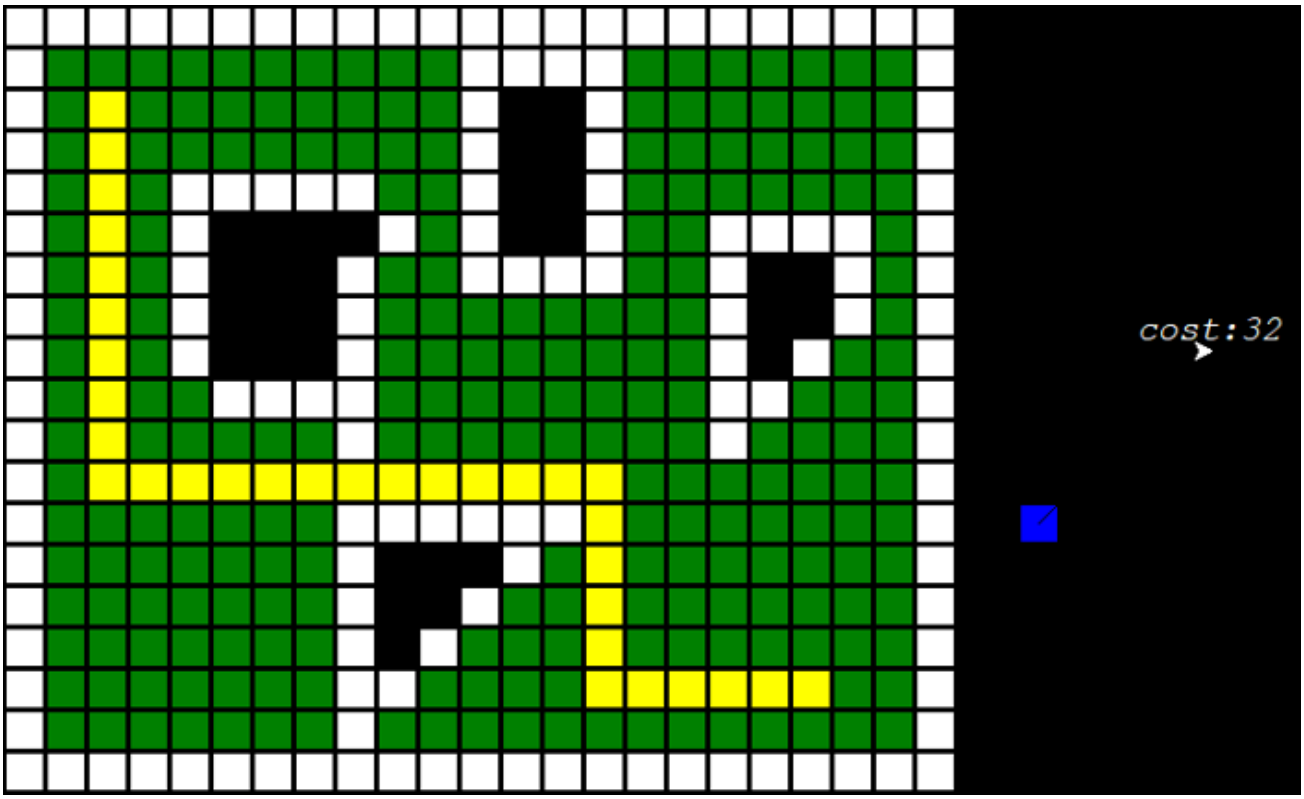
- Là kỹ thuật tìm kiếm trên tất cả các nút của một mức trong không gian bài toán trước khi chuyển sang các nút của mức tiếp theo.
- Tìm kiếm theo chiều rộng (BFS) là một thuật toán tìm kiếm trong đồ thị trong đó việc tìm kiếm chỉ bao gồm 2 thao tác:
  - Cho trước một đỉnh của đồ thị
  - Thêm các đỉnh kề với đỉnh vừa cho vào danh sách có thể hướng tới tiếp theo
- Pesudo code

```
function BREADTH-FIRST-SEARCH(problem) returns a solution, or failure
  node ← a node with STATE = problem.INITIAL-STATE, PATH-COST = 0
  if problem.GOAL-TEST(node.STATE) then return SOLUTION(node)
  frontier ← a FIFO queue with node as the only element
  explored ← an empty set
  loop do
    if EMPTY?(frontier) then return failure
    node ← POP(frontier) /* chooses the shallowest node in frontier */
    add node.STATE to explored
    for each action in problem.ACTIONS(node.STATE) do
      child ← CHILD-NODE(problem, node, action)
      if child.STATE is not in explored and not in frontier then
        if problem.GOAL-TEST(child.STATE) then return SOLUTION(child)
        frontier ← INSERT(child, frontier)
```

- **Ưu điểm**
  - Kỹ thuật tìm kiếm rộng là kỹ thuật vét cạn không gian trạng thái bài toán vì vậy sẽ tìm được lời giải nếu có.
  - Đường đi qua điểm ít nhất
- **Khuyết điểm**
  - không phù hợp vào các bài toán có giá trị lớn :
  - Tốn bộ nhớ
  - Tốn nhiều bước để xử lí
  - Nếu thiếu dữ kiện thì không hoàn thành bài toán
- **Input**
  - Với điểm bắt đầu là ô vuông có màu đỏ
  - Điểm đích là ô vuông có màu xanh
  - Các đa giác là các vật cản



- **Output:**
  Sau khi tiến hành thực hiện UCS thì ta sẽ được kết quả :
  Đường màu vàng là đường đi từ điểm bắt đầu đến điểm kết thúc
  Cost là chi phí thực hiện đường đi



- **Kết luận**

Đây là thuật toán nên được áp dụng trong những đồ thị nhỏ vì sẽ luôn tìm được đường đi với ít bước nhất nhưng không phải lúc nào cũng là con đường ngắn nhất và không tối ưu được chi phí bộ nhớ

2. Uniform-cost search

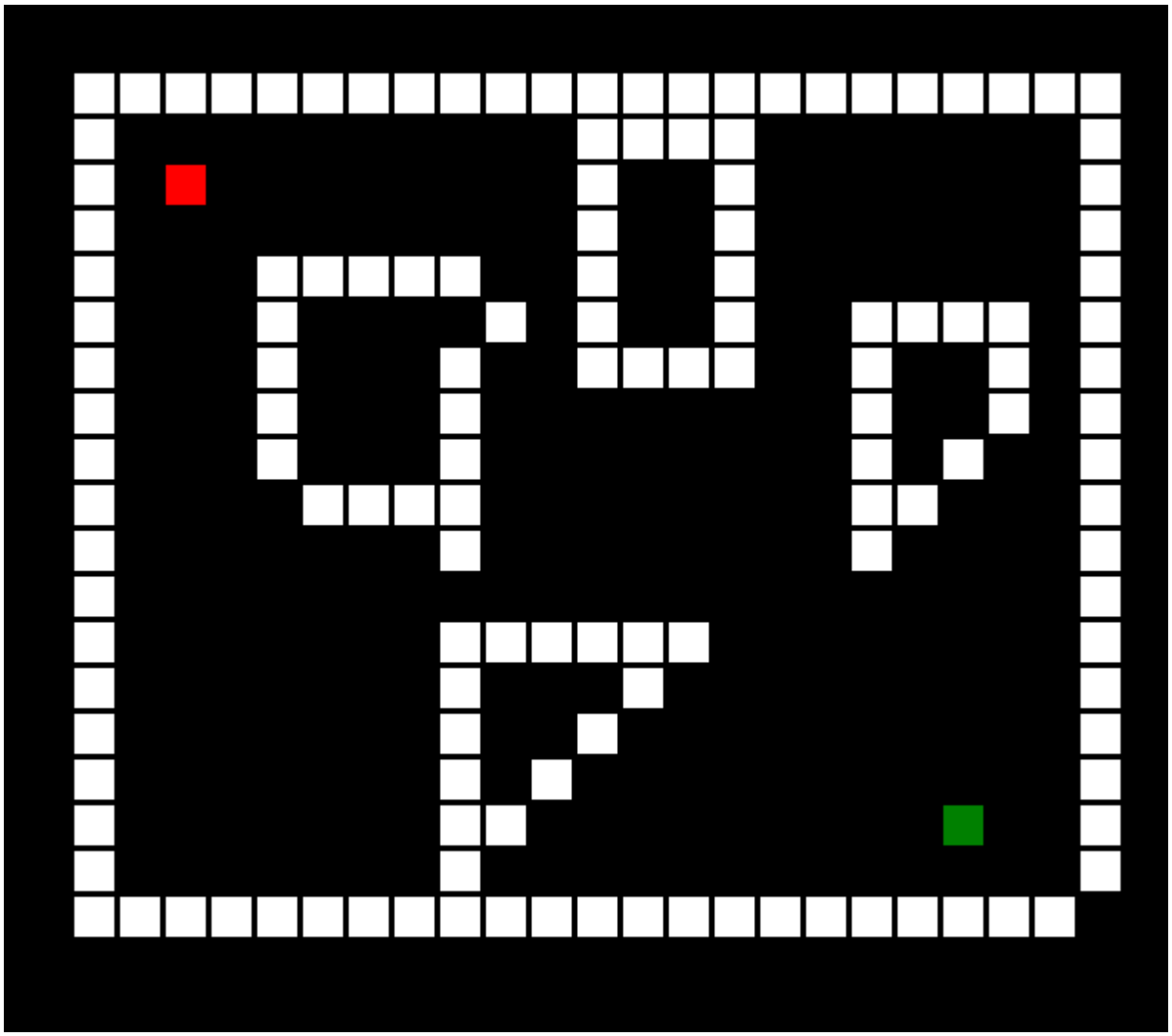
- Ucs là phương pháp tìm kiếm đường đi ngắn nhất với việc sử dụng hàng đợi ưu tiên(priorityQueue)
- Nếu như đồ thị không có trọng số (trọng số bằng nhau) thì ucs tương tự như là BFS
- Nếu như đồ thị có trọng số thì UCS sẽ tương tự như thuật toán Dijkstra

Pseudo code

```
function UNIFORM-COST-SEARCH(problem) returns a solution, or failure
  node ← a node with STATE = problem.INITIAL-STATE, PATH-COST = 0
  frontier ← a priority queue ordered by PATH-COST, with node as the element
  explored ← an empty set
  loop do
    if EMPTY?(frontier) then return failure
    node ← POP(frontier) /* chooses the lowest-cost node in frontier */
    if problem.GOAL-TEST(node.STATE) then return SOLUTION(node)
    add node.STATE to explored
    for each action in problem.ACTIONS(node.STATE) do
      child ← CHILD-NODE(problem, node, action)
      if child.STATE is not in explored and not in frontier then
        frontier ← INSERT(child, frontier)
      else if child.STATE is in frontier with higher PATH-COST then
        replace that frontier node with child
```

- **Ưu điểm**
  - Luôn tìm thấy được kết quả nếu có
  - Tối ưu nếu chi phí lớn hơn không
- **Khuyết điểm**
  - Phải sử dụng hàng đợi ưu tiên
  - Chi phí lưu trữ lớn
- **Input**

  Với điểm bắt đầu là ô vuông có màu đỏ
  Điểm đích là ô vuông có màu xanh
  Các đa giác là các vật cản



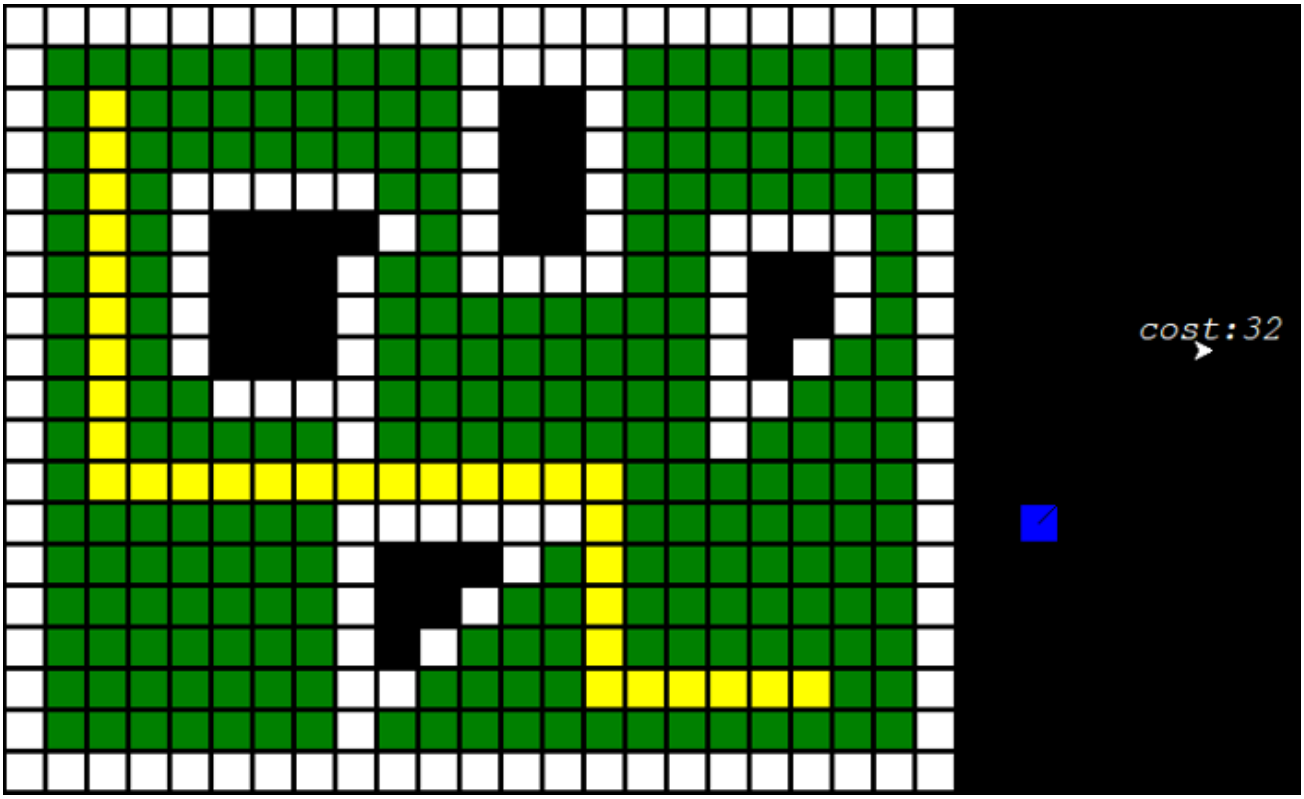
- **Output:**

Sau khi tiến hành thực hiện bfs thì ta sẽ được kết quả :

Đường màu vàng là đường đi từ điểm bắt đầu đến điểm kết thúc

Đường màu xanh là đường đã đi

Cost là chi phí thực hiện đường đi



- **Kết luận**
  - UCS là tìm kiếm mù và là giải pháp tối ưu để tìm đường dẫn từ nút gốc đến nút đích với chi phí tích luỹ ít nhất trong tìm kiếm có trọng số

3. Iterative deepening search

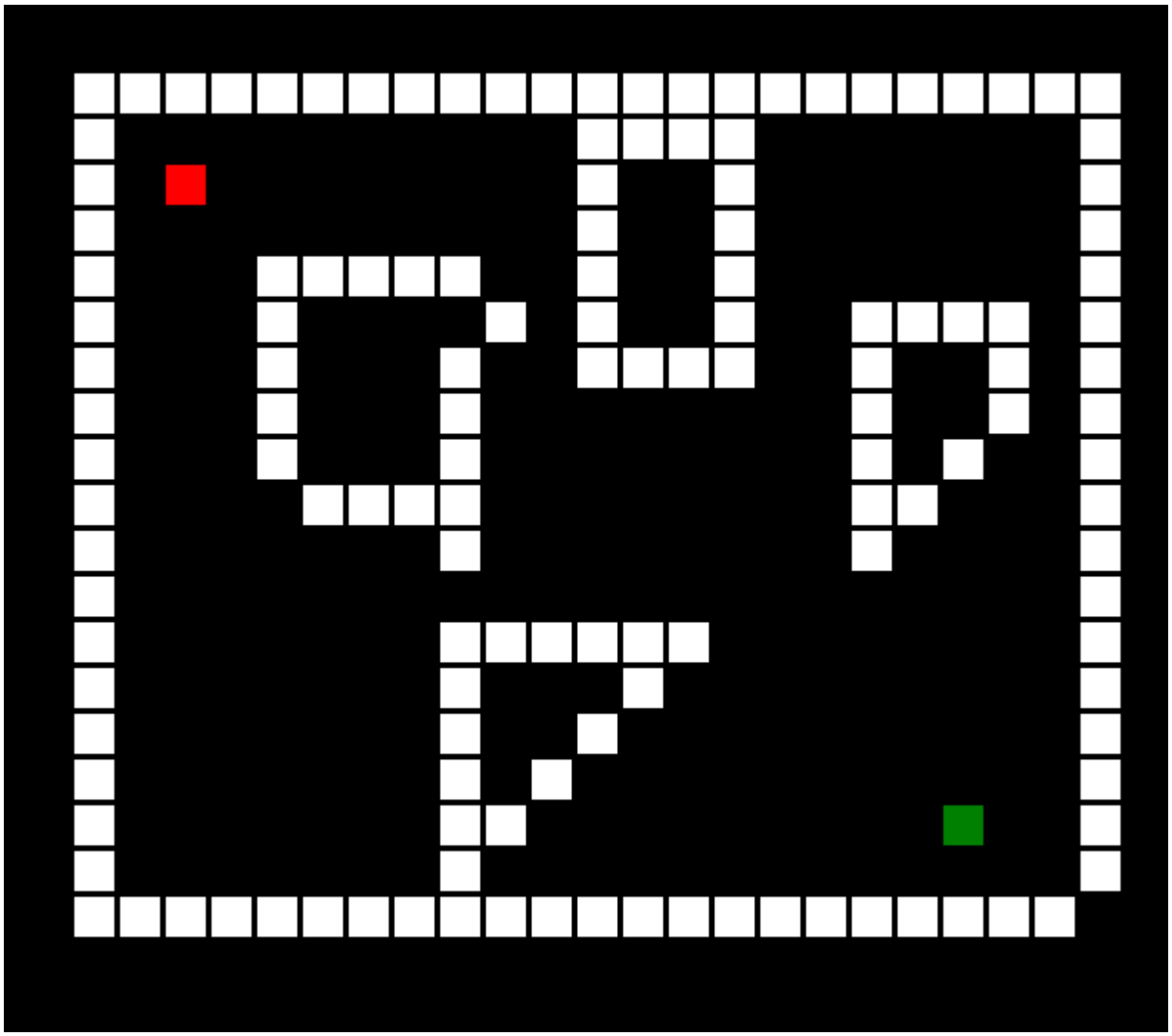
- Là thuật toán tìm kiếm mù giống như BFS và DFS. IDS là sự kết hợp của hai thuật toán BFS và DFS và khắc phục được những khuyết điểm của cả hai

Pseudo code

```
function ITERATIVE-DEEPENING-SEARCH(problem) returns a solution, or failure
  for depth = 0 to ∞ do
    result ← DEPTH-LIMITED-SEARCH(problem, depth)
  if result ≠ cutoff then return result
```

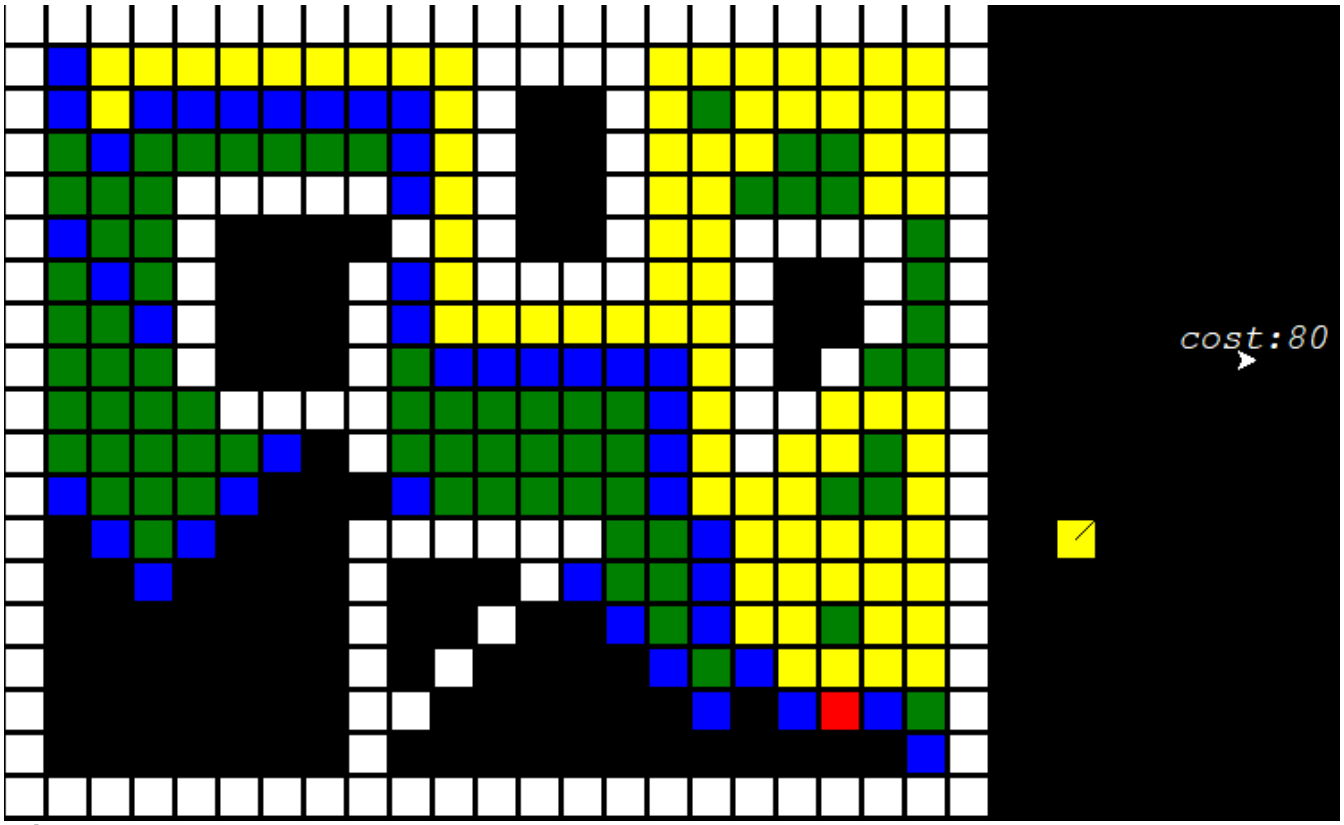
- **Ưu điểm**
  - Hiệu suất tốt hơn BFS và DFS
  - Độ phức tạp của không gian là O(d)
  - Tối ưu nếu mỗi bước cost =1
- **Khuyết điểm**
  - Thời gian thực hiện là cấp số nhân O(d^b)
- **Input**

Với điểm bắt đầu là ô vuông có màu đỏ
Điểm đích là ô vuông có màu xanh
Các đa giác là các vật cản



- **Output**

Sau khi tiến hành thực hiện IDS thì ta sẽ được kết quả :
Đường màu vàng là đường đi từ điểm bắt đầu đến điểm kết thúc
Màu xanh dương là frontier
Màu xanh lá cây là đường đã được duyệt
Cost là chi phí thực hiện đường đi



- **Kết luận**
  - Là 1 thuật toán kết hợp của BFS và DFS và khắc phục những khuyết điểm của họ

4. Greedy-best first search

- Đây là kỹ thuật tìm kiếm dựa vào tìm kiếm kinh nghiệm(sử dụng hàm đánh giá trong bài là manhattan distance)
- Và hàm đánh giá tốt có thể giảm thời gian và không gian nhớ 1 cách đáng kể

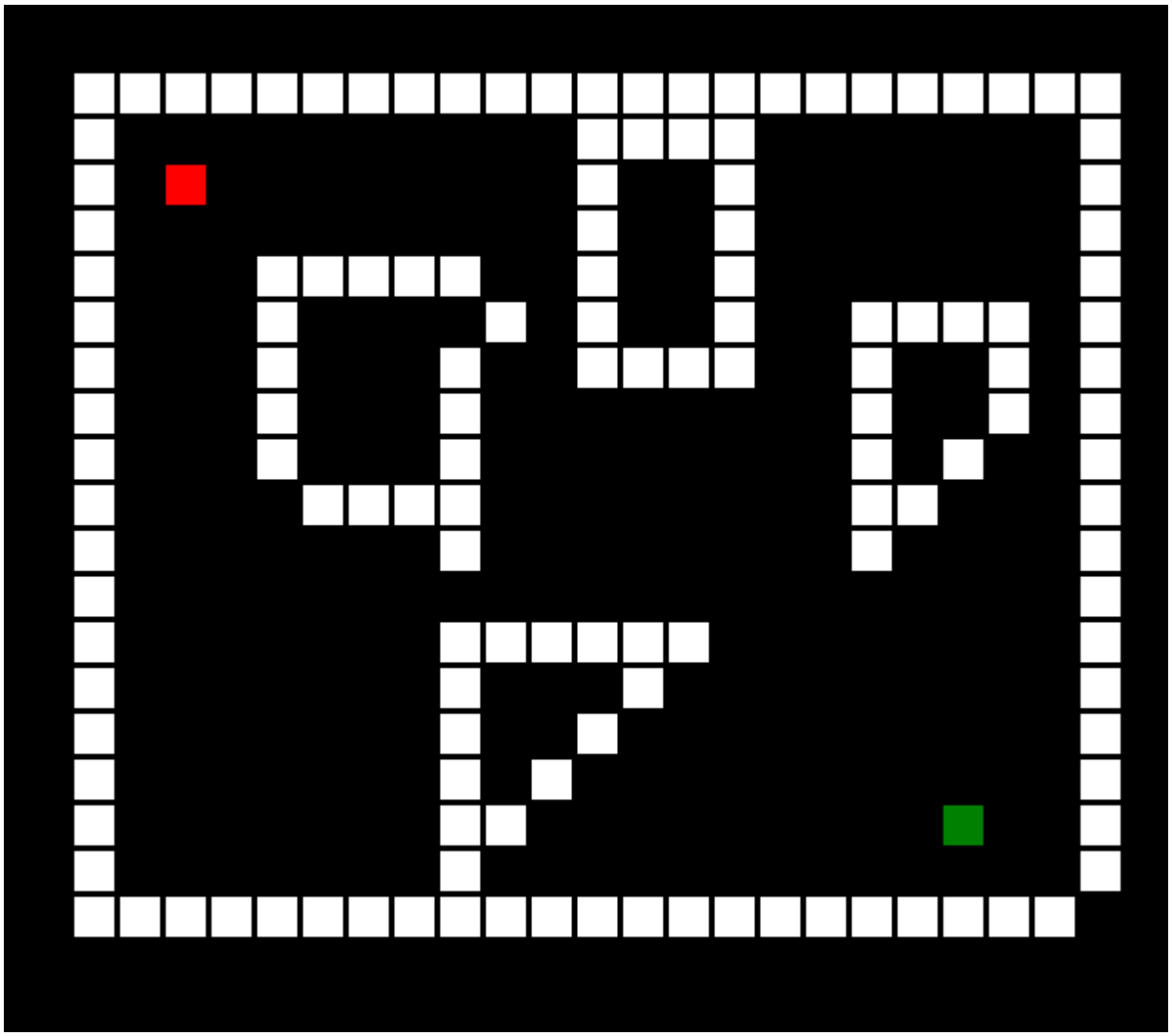
Pesudo code

```
    1) Create an empty PriorityQueue
       PriorityQueue pq;
    2) Insert "start" in pq.
       pq.insert(start)
    3) Until PriorityQueue is empty
          u = PriorityQueue.DeleteMin
          If u is the goal
             Exit
          Else
             Foreach neighbor v of u
                If v "Unvisited"
                    Mark v "Visited"
                    pq.insert(v)
             Mark u "Examined"
End procedure
```

- Ưu điểm:
  - Linh hoạt hơn BFS và DFS
  - Độ phức tạp thời gian thấp hơn breadth first search
  - Độ phức tạp thời gian và không gian có thể giảm nếu hàm đánh giá tốt

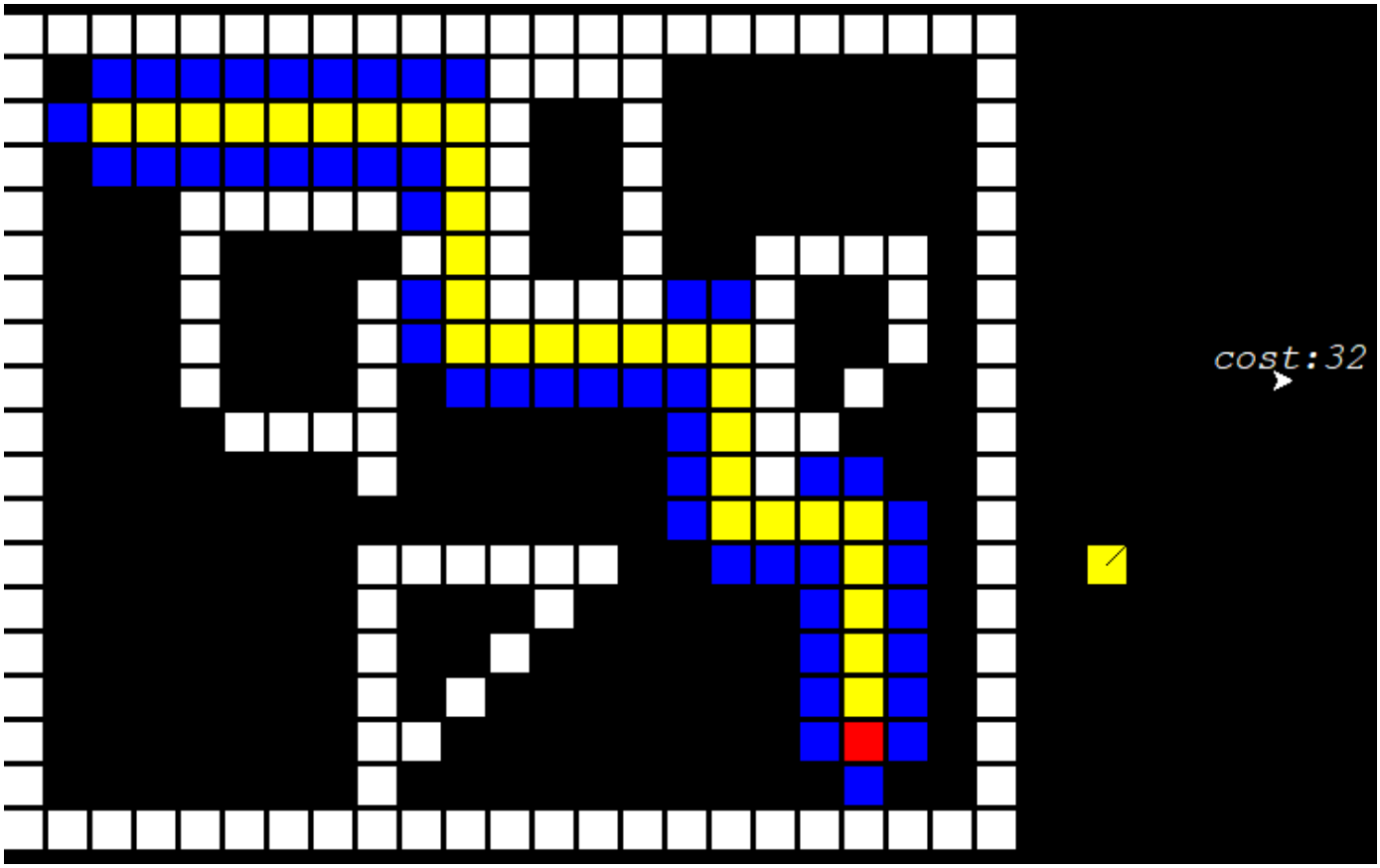
- Khuyết điểm
  - Không có tối ưu
  - Có thể bị kẹt trong vòng lặp
- Input

  Với điểm bắt đầu là ô vuông có màu đỏ
  Điểm đích là ô vuông có màu xanh
  Các đa giác là các vật cản



- **Output**

Sau khi tiến hành thực hiện IDS thì ta sẽ được kết quả :
Đường màu vàng là đường đi từ điểm bắt đầu đến điểm kết thúc
Màu xanh dương là frontier
Màu xanh lá cây là đường đã được duyệt
Cost là chi phí thực hiện đường đi

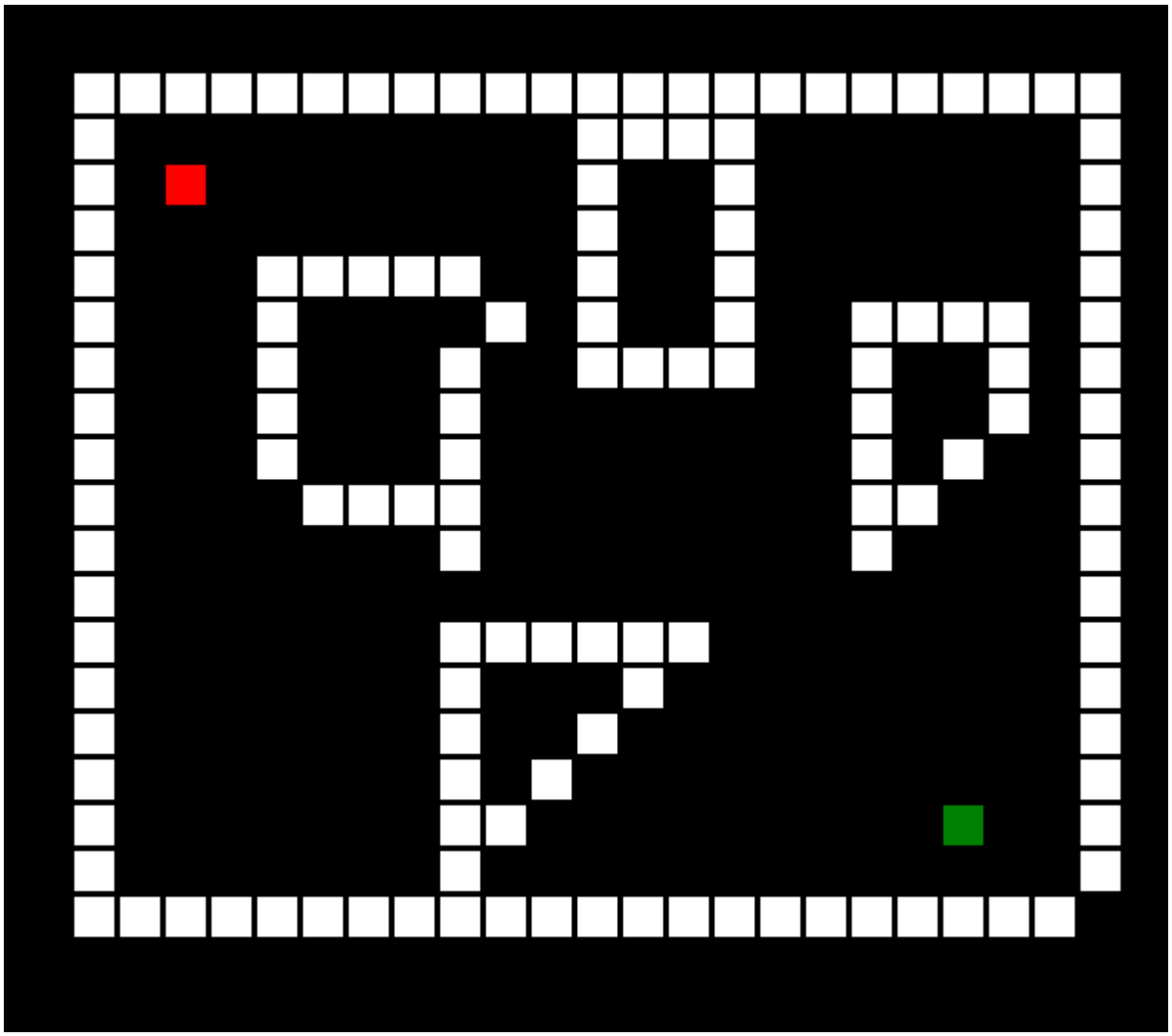


- Kết luận:
  - Là kỹ thuật tìm kiếm dựa vào kinh nghiệm (hàm đánh giá tốt thì làm cho thuật toán được cải thiện đáng kể)
  - Và best first search nồng nặc mùi greedy

5. **A* search**

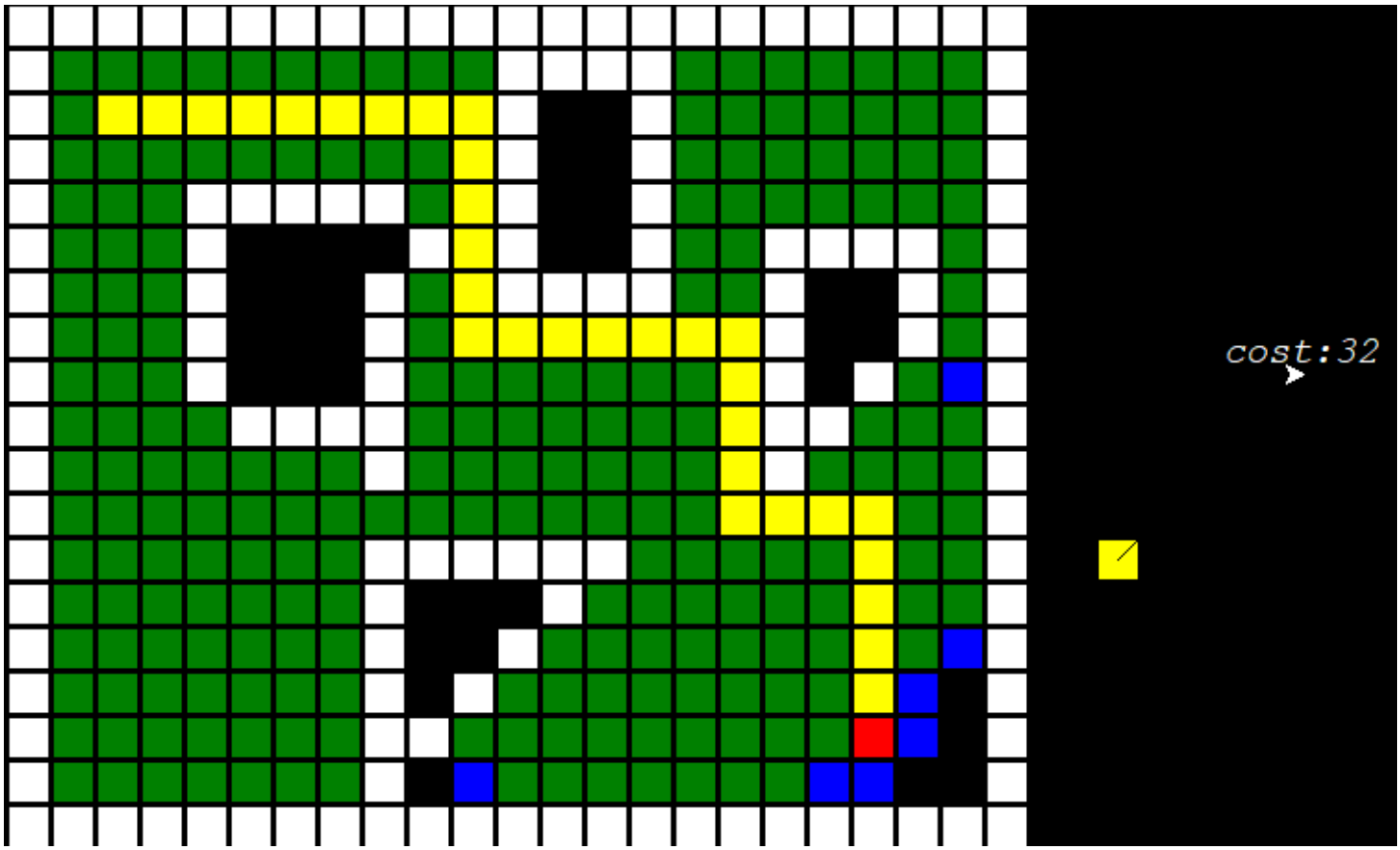
- là sự kết hợp giữa UCS và best first search
- tìm kiếm đường đi ngắn nhất thông qua kinh nghiệm(hàm đánh giá)
- **ưu điểm**
  - hoàn thành và tối ưu
  - Là 1 trong những kỹ thuật tốt nhất và được dùng để giải quyết những trường hợp phức tạp
- **Khuyết điểm**
  - Tốc độ thực thi phụ thuộc vào độ chính xác của thuật toán heuristic được sử dụng để tính h(n)
- **Input**

  Với điểm bắt đầu là ô vuông có màu đỏ
  Điểm đích là ô vuông có màu xanh
  Các đa giác là các vật cản



- **Output**

Sau khi tiến hành thực hiện IDS thì ta sẽ được kết quả :
Đường màu vàng là đường đi từ điểm bắt đầu đến điểm kết thúc
Màu xanh dương là frontier
Màu xanh lá cây là đường đã được duyệt
Cost là chi phí thực hiện đường đi



3 Hướng dẫn chạy thuật toán

Bước 1: nhập vào filename input
Bước 2: nhập vào các số từ 1-5 tương ứng với loại search mà bạn muốn thực thi

```
Enter filename: input.txt
1.Breadth-first search

2.Uniform-cost search

3.Iterative deepening search

4.Greedy-best first search

5.Graph-search A*

------------------&&----------------------
 please input the option do you want(1-5):
```

Lúc này sẽ chương trình sẽ hiện lên cửa sổ turtle và hiển thị kết quả.

4 Tài liệu tham khảo

- **Tham khảo việc vẽ turtle: https://github.com/tonypdavis/BFS-Maze-Solver**
- **Tham khảo về BFS: https://github.com/tonypdavis/BFS-Maze-Solver**
- **Tham khảo về UCS: slide môn học và https://stackoverflow.com/questions/48884990/using-uniform-cost-search-on-a-matrix-in-python**
- **Tham khảo về IDS: https://stackoverflow.com/questions/70412198/iterative-deepening-search-python-implementation**
- **Tham khảo về gbfs : https://www.youtube.com/watch?v=9p3LXBLqdmQ**
- **Tham khảo về manhattan: https://www.youtube.com/watch?v=p3HbBlcXDTE&t=195s**