Đại học Khoa học Tự nhiên Tp. Hồ Chí Minh
Khoa Công nghệ Thông tin
Lớp Cử nhân Tài năng khóa 2016
Môn học: Cơ sở Trí tuệ Nhân tạo

BÁO CÁO ĐỒ ÁN
TÌM HIỂU HEURISTIC VỚI A*

Tháng 10/2018

MỤC LỤC

I. Thông tin từng thành viên

Họ và Tên	Mã số Sinh viên
Trần Quốc Cường	1612843
Trần Mai Khiêm	1612869

II. Công việc được giao cho từng thành viên và mức độ hoàn thành

Công việc	Người thực hiện và mức độ hoàn thành	
	Trần Quốc Cường	**Trần Mai Khiêm**
Xây dựng và cài đặt thuật toán A*	**100%**	
Xây dựng GUI cho thuật toán A*		**100%**
Kiểm thử thuật toán A*		**100%**
Xây dựng và cài đặt thuật toán ARA*	**100%**	
Kiểm thử thuật toán ARA*		**100%**
Báo cáo	**100%**	

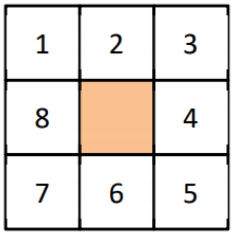
III. Những vấn đề chưa thực hiện được

Những vấn đề chưa thực hiện được tập trung nhiều vào phần GUI

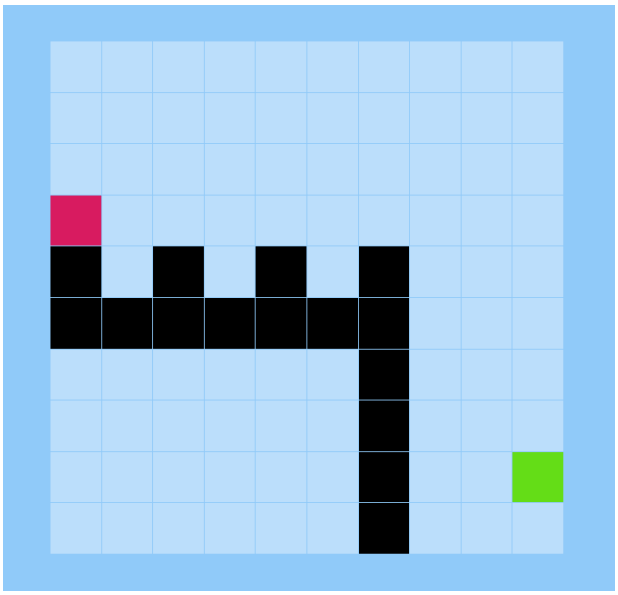
- Phần GUI của A* không thể vừa kết hợp import từ file và vừa vẽ trên GUI
- Khi mà nhập liệu trên GUI chưa hợp lệ (số nút nguồn hoặc số nút đích khác 1) thì không thể dựa vào cấu hình hiện tại để sửa đổi mà phải nhấn vào nút Reset
- Tốc độ vẽ của GUI bắt đầu chậm thấy rõ từ $n \geq 200$

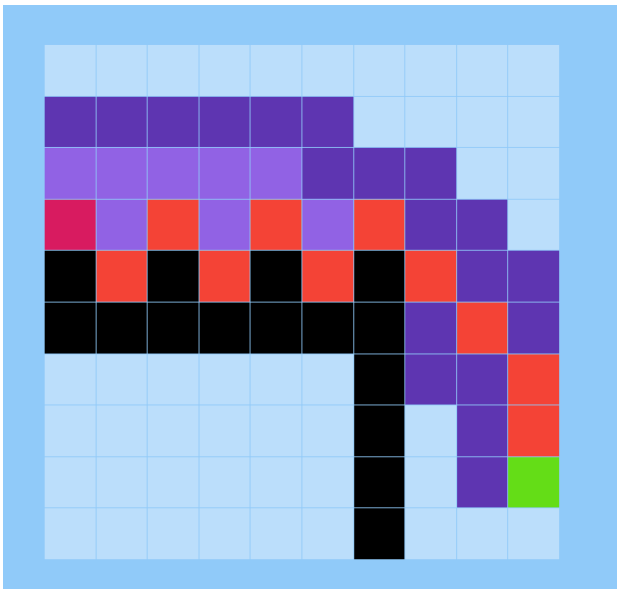
IV. Các bộ test đặc trưng

1. Thứ tự mở các ô gần kề



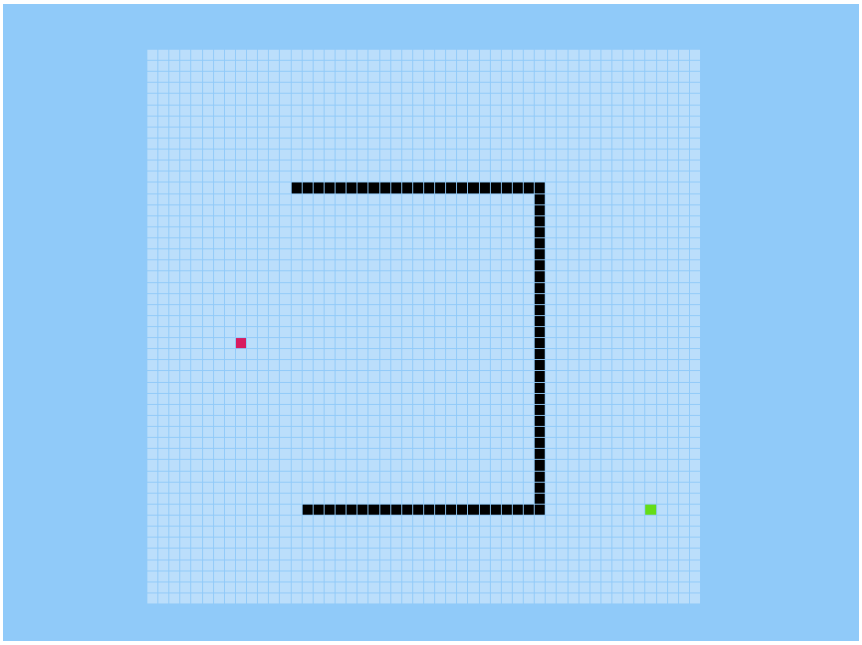
Ta đang sử dụng hàm heuristic là khoảng cách Euclidean nên mặc dù thứ tự cho các ô vào hàng đợi là theo chiều kim đồng hồ nhưng khi mở các ô trong tập OPEN thì thuật toán không hẳn sẽ chạy theo giống như độ ưu tiên như vậy. Lý do là một số ô sẽ có hàm heuristic nhỏ hơn các ô còn lại nên sẽ được mở trước.





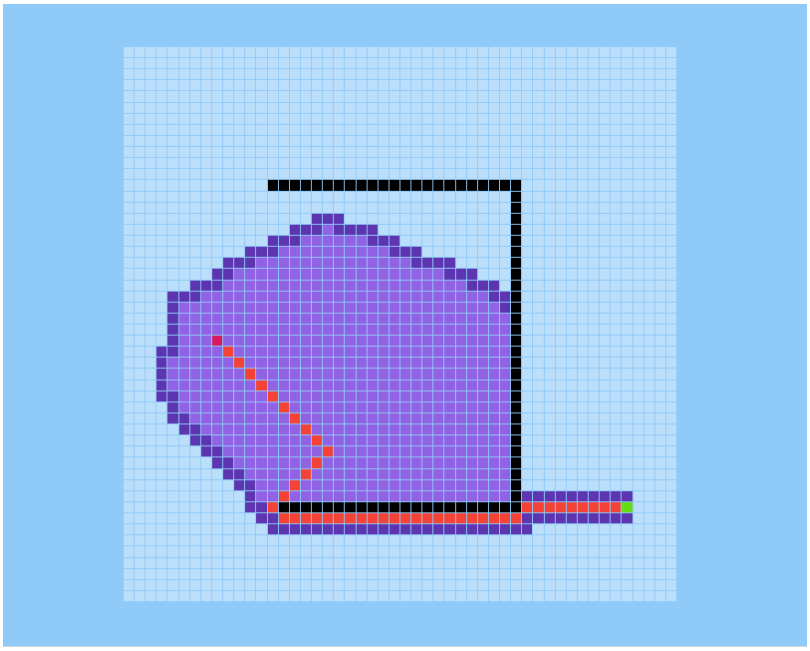
Như trong bộ test trên vì đi từ ô nguồn ở tọa độ (3, 0) đi ngang theo một đường thẳng tới ô có tọa độ (3, 6) thì thuật toán A* lại mở theo các ô theo đường gấp khúc.

2. Xét các ô khả năng cao là nằm trên đường đi ngắn nhất trước

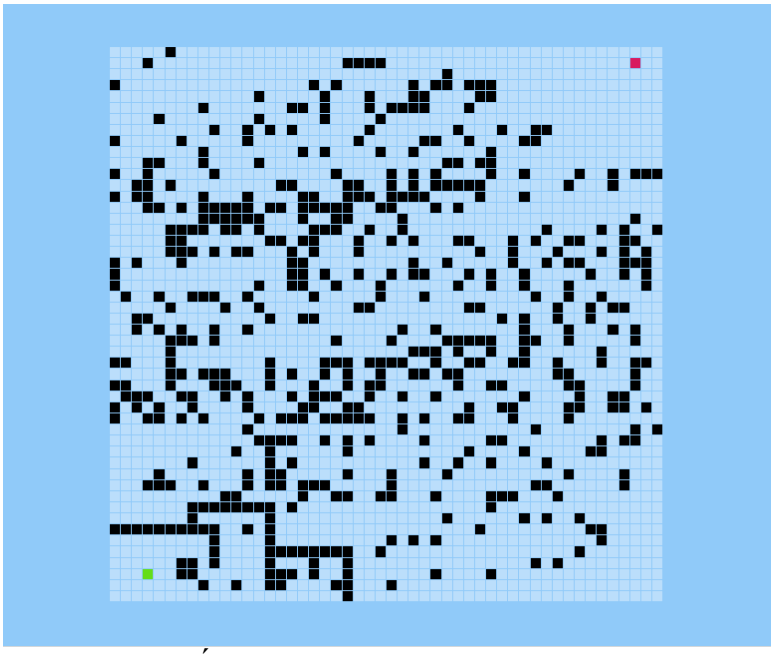


Ta xét trường hợp điểm nguồn đang đứng trước một chướng ngại vật như trong thực tế có hình chữ U. Thì nếu sử dụng các thuật toán như Uniform Cost Search thì ta sẽ phải duyệt tất cả các ô trong chướng ngại vật đó. Nhờ việc sử dụng thêm hàm heuristic trong A* thì

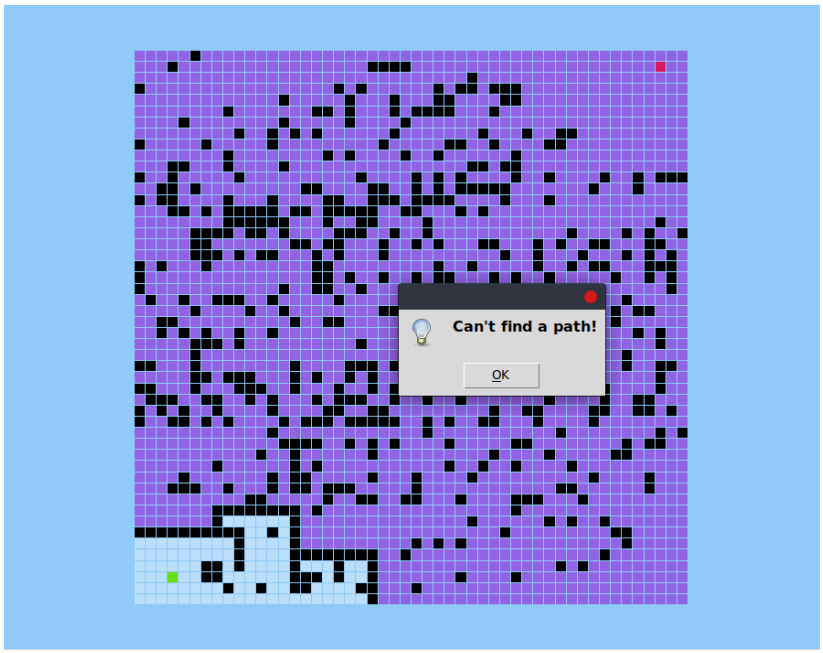
ta chỉ cần duyệt một số phần trong chướng ngại vật mà những ô đó có xu hướng “gần” với điểm nguồn nhất



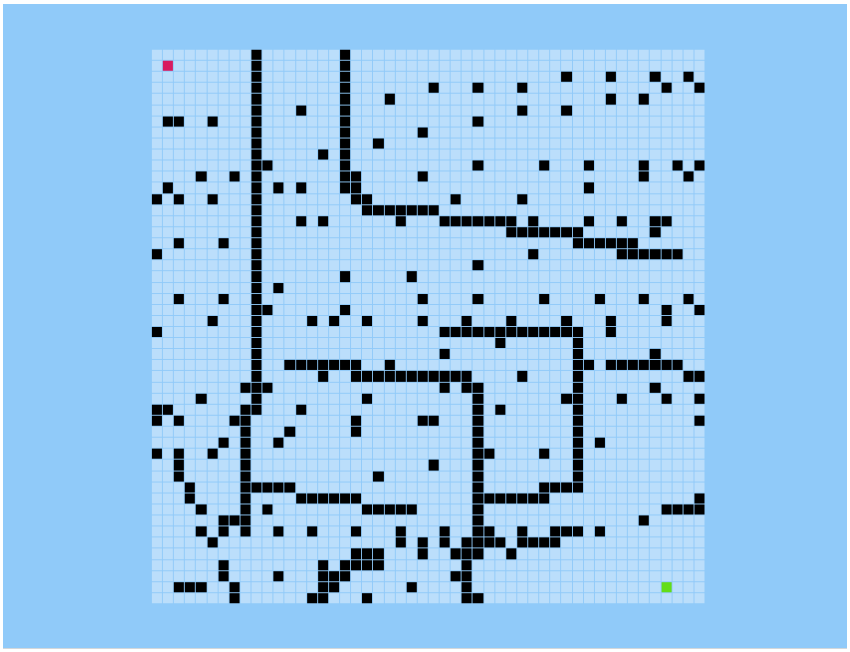
3. Trường hợp không tồn tại đường đi



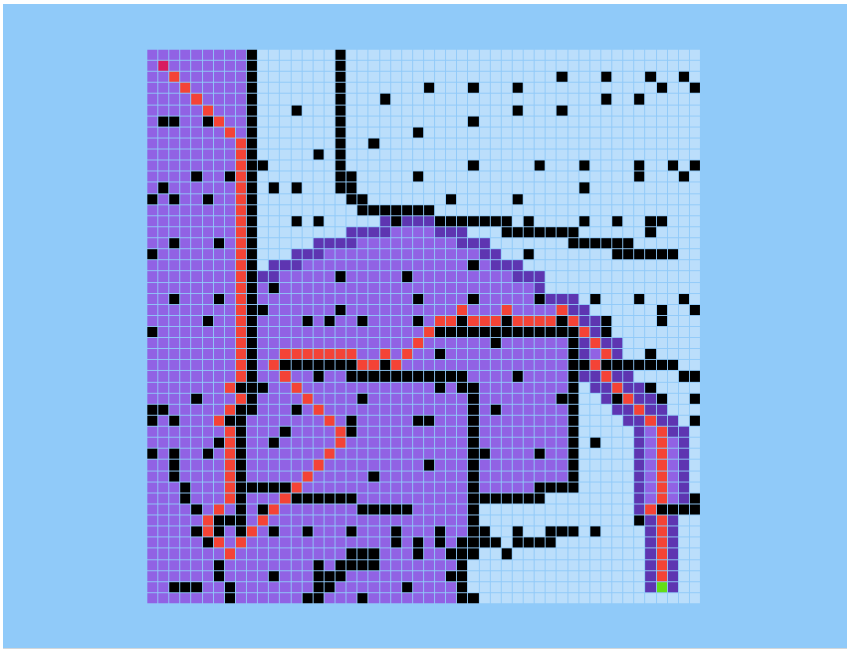
Trong trường hợp không tìm thấy đường đi thì thuật toán A* sẽ mở những ô mà nó có thể đến được. Thuật toán sẽ dừng lại khi tất cả các ô đó đã được mở.



4. Trường hợp với số đỉnh lớn



Với n = 50



V. Hàm heuristic mới chấp nhận được

1. Nhận xét heuristic dùng khoảng cách Euclidean

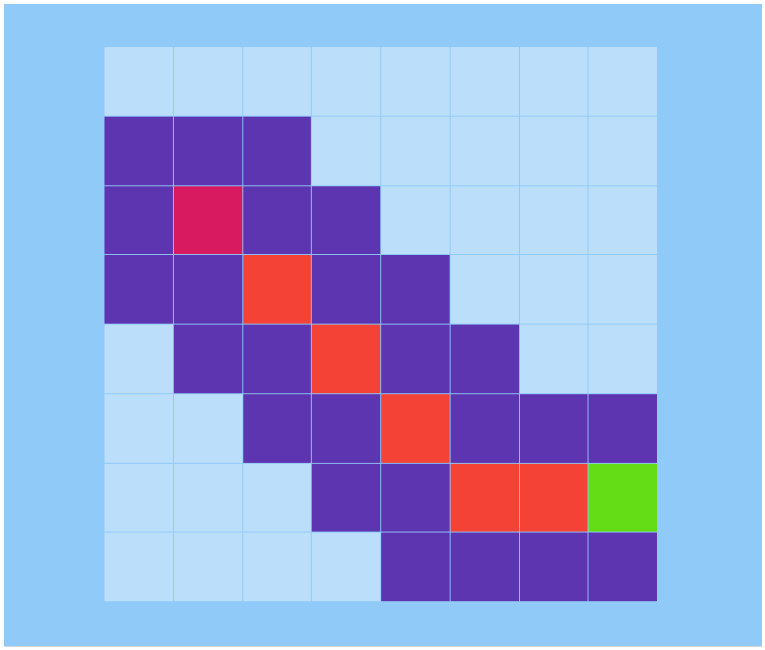
Gọi ô đích có tọa độ (Gx, Gy)

Ta có hàm heuristic của ô có tọa độ (x_0, y_0) là

$$h(x_0, y_0) = \sqrt{(x_0 - Gx)^2 + (y_0 - Gy)^2}$$

Xét $(Gx, Gy) = (6, 7)$. Khi đó

$$h(2, 1) = \sqrt{(2-6)^2 + (1-7)^2} = \sqrt{52} \approx 7.2$$



Nhưng ta có thể thấy như trong hình thì $h^*(2,1) = 6$

$$\rightarrow \exists h(x_0, y_0) > h^*(x_0, y_0)$$

Nên heuristic dùng khoảng cách Euclidean là một heuristic không chấp nhận được

2. Hàm heuristic mới

Với mỗi bước đi gọi ô hiện tại là (x_0, y_0) thì những ô đến được từ ô đó gọi là (x_1, y_1)

Gọi $\Delta x = x_1 - x_0,\ \Delta y = y_1 - y_0$ thì ta có $|\Delta x| \leq 1,\ |\Delta y| \leq 1$

Dựa vào đó ta đặt ra hàm heuristic của ô có tọa độ (x_0, y_0) chính là giá trị lớn nhất của khoảng cách theo x và khoảng cách theo y của (x_0, y_0) và (Gx, Gy)

$$h(x_0, y_0) = \max(|x_0 - Gx|, |y_0 - Gy|)$$

Dễ thấy $h(x, y) \geq 0$, vì là giá trị lớn hơn trong hai giá trị không âm

Ta có $h(x, y) \leq h^*(x, y)$, vì

- Chiếu theo trục x (tức là chiếu theo từng hàng) di chuyển từ (x_0, y_0) đến (Gx, Gy) tương ứng với di chuyển từ hàng x_0 đến hàng Gx ta lại có $|\Delta x| \leq 1$ nên số bước ít nhất sẽ là $|x_0 - Gx|$

- Chiếu theo trục y (tức là chiếu theo từng côt) di chuyển từ (x_0, y_0) đến (Gx, Gy) tương ứng với di chuyển từ cột y_0 đến cột Gy ta lại có $|\Delta y| \leq 1$ nên số bước ít nhất sẽ là $|x_1 - Gy|$

$$\rightarrow h^*(x_0, y_0) \geq \max(|x_0 - Gx|, |y_0 - Gy|)$$

Vậy hàm heuristic $h(x_0, y_0) = \max(|x_0 - Gx|, |y_0 - Gy|)$ là chấp nhận được

Ta sẽ dùng hàm heuristic trên trong thuật toán ARA* sẽ trình bày ở phần sau

VI. Tổng quan thuật toán ARA*

Trong thực tế, thời gian để giải những bài toán tìm đường đi hay lập lịch thường bị giới hạn

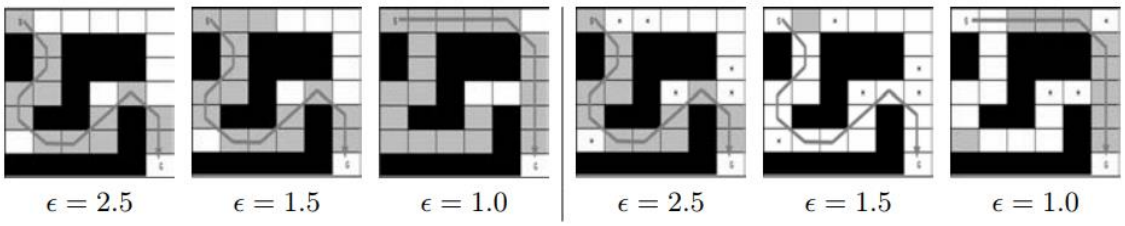
ARA* ra đời nhằm mục đích giải quyết bài toán tìm đường đi dựa trên phương pháp của A* nhưng với mỗi khoảng thời gian nhất định phải tìm ra được đường đi tốt nhất có thể.

Để tăng tốc A* người ta thường thêm vào hệ số $\in \geq 1$ vào hàm

$$f(v) = g(v) + \in * h(v)$$

Có một cách tiếp cận là ta bắt đầu với một giá trị $\in$ lớn ban đầu, khi đó thuật toán sẽ chạy rất nhanh mặc dù rất dễ rơi vào trường hợp chưa tối ưu kết quả. Ta tiến hành tìm đường đi với $\in$ hiện tại. Nếu thời gian còn lại vẫn còn cho phép thì tiến hành giảm $\in$ xuống rồi tìm đường đi với $\in$ mới này. $\in$ càng nhỏ thì đường đi càng tối ưu hơn, đi cùng với đó là thời gian thực hiện cũng lớn hơn. Cứ tiến hành cho đến khi còn thời gian hoặc $\in = 1$ (thuật toán A* thông thường).

Với thuật toán Anytime A* thì khi ta giảm $\in$ xuống thì ta vẫn phải duyệt lại tất cả các ô dù đã từng được mở trước đó trong trường hợp $\in$ cũ. Còn với ARA* thì tận dụng lại nhưng ô đã lan ở $\in$ cũ và chỉ cần cập nhật ở ô cần thay đổi



Bên trái là thuật toán Anytime A* và bên phải là thuật toán ARA*

ARA* làm việc bằng cách thực A* nhiều lần, bắt đầu bằng ϵ lớn và tiến hành giảm ϵ cho đến khi $\epsilon = 1$. Sau mỗi lần thay đổi ϵ, đường đi tìm được phải thỏa mã là tối ưu với ϵ tương ứng. Chạy A* lại từ đầu với mỗi lượt đòi hỏi chi phí rất lớn và lãng phí. Ta chỉ cần thực hiện mở lại những ô chưa từng được mở hoặc là những ô mà chưa thích hợp

Mã giả

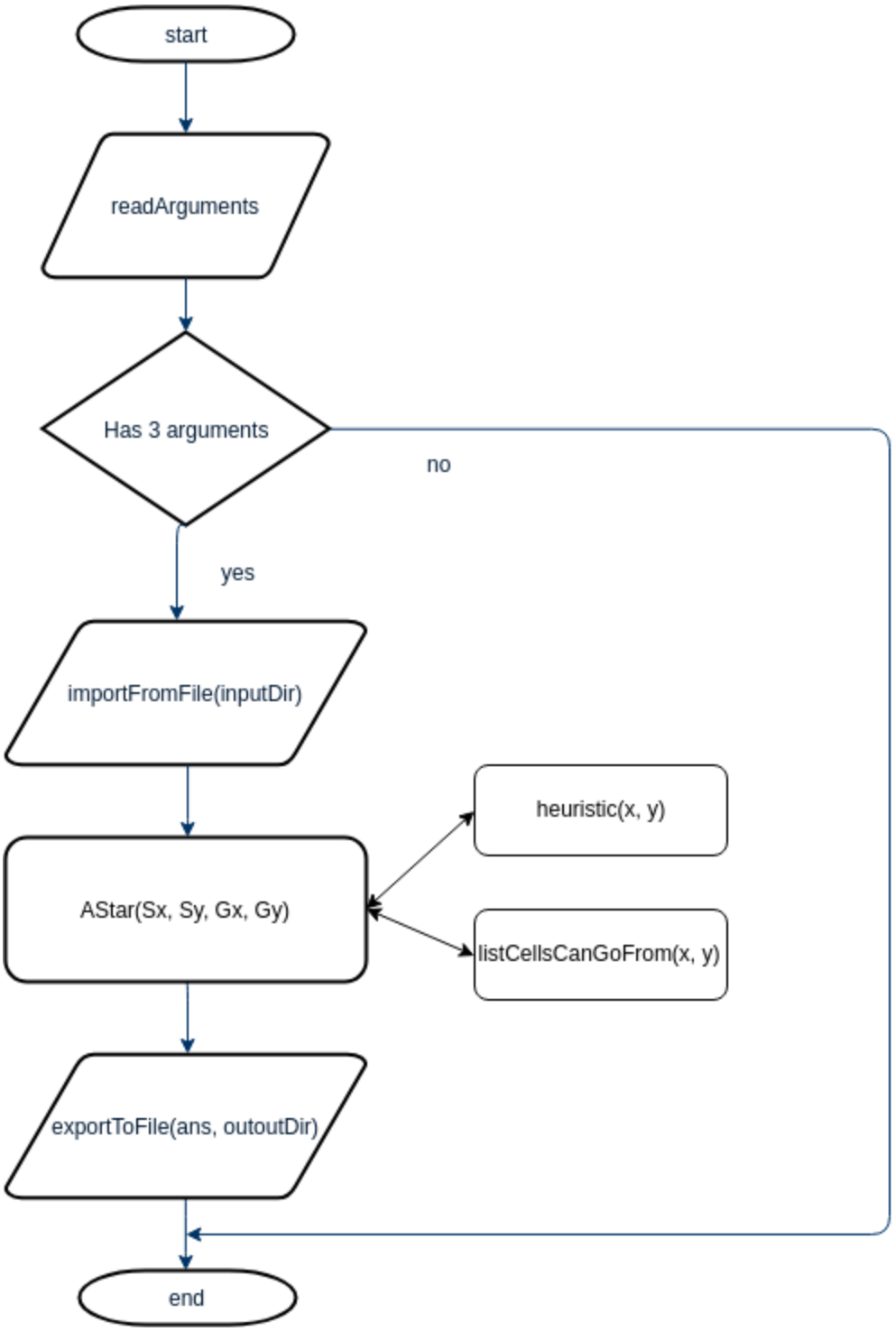
```
procedure fvalue(s)
01 return g(s) + ε * h(s);

procedure ImprovePath()
02 while(fvalue(s_goal) > min_{s∈OPEN}(fvalue(s)))
03   remove s with the smallest fvalue(s) from OPEN;
04   CLOSED = CLOSED ∪ {s};
05   for each successor s' of s
06     if s' was not visited before then
07       g(s') = ∞;
08     if g(s') > g(s) + c(s, s')
09       g(s') = g(s) + c(s, s');
10       if s' ∉ CLOSED
11         insert s' into OPEN with fvalue(s');
12       else
13         insert s' into INCONS;
```

```
procedure Main()
01' g(s_goal) = ∞; g(s_start) = 0;
02' OPEN = CLOSED = INCONS = ∅;
03' insert s_start into OPEN with fvalue(s_start);
04' ImprovePath();
05' ε' = min(ε, g(s_goal)/ min_{s∈OPEN∪INCONS}(g(s)+h(s)));
06' publish current ε'-suboptimal solution;
07' while ε' > 1
08'   decrease ε;
09'   Move states from INCONS into OPEN;
10'   Update the priorities for all s ∈ OPEN according to fvalue(s);
11'   CLOSED = ∅;
12'   ImprovePath();
13'   ε' = min(ε, g(s_goal)/ min_{s∈OPEN∪INCONS}(g(s)+h(s)));
14'   publish current ε'-suboptimal solution;
```

Một trạng thái gọi là không nhất quán nội bộ nếu nếu với mỗi lần ta giảm giá trị g của nó xuống thì lần sau trạng thái đó sẽ được mở rộng. Nói cách khác nếu ta giảm giá trị g của trạng thái đó xuống thì những trạng thái con của nó cũng có thể có giá trị g bị giảm theo. Nên nó cũng khiến các trạng thái con của nó trở thành không nhất quán nội bộ. A* với một hàm heuristic thích hợp sẽ đảm bảo không mở một ô nào đó quá 1 lần. Với việc $\epsilon > 1$ thì có thể vi phạm tính thích hợp và tính chấp nhận được nên vì thế tìm kiếm A* có thể mở lại một trạng thái nhiều lần.

VII. Sơ đồ biểu diễn hệ thống phần mềm

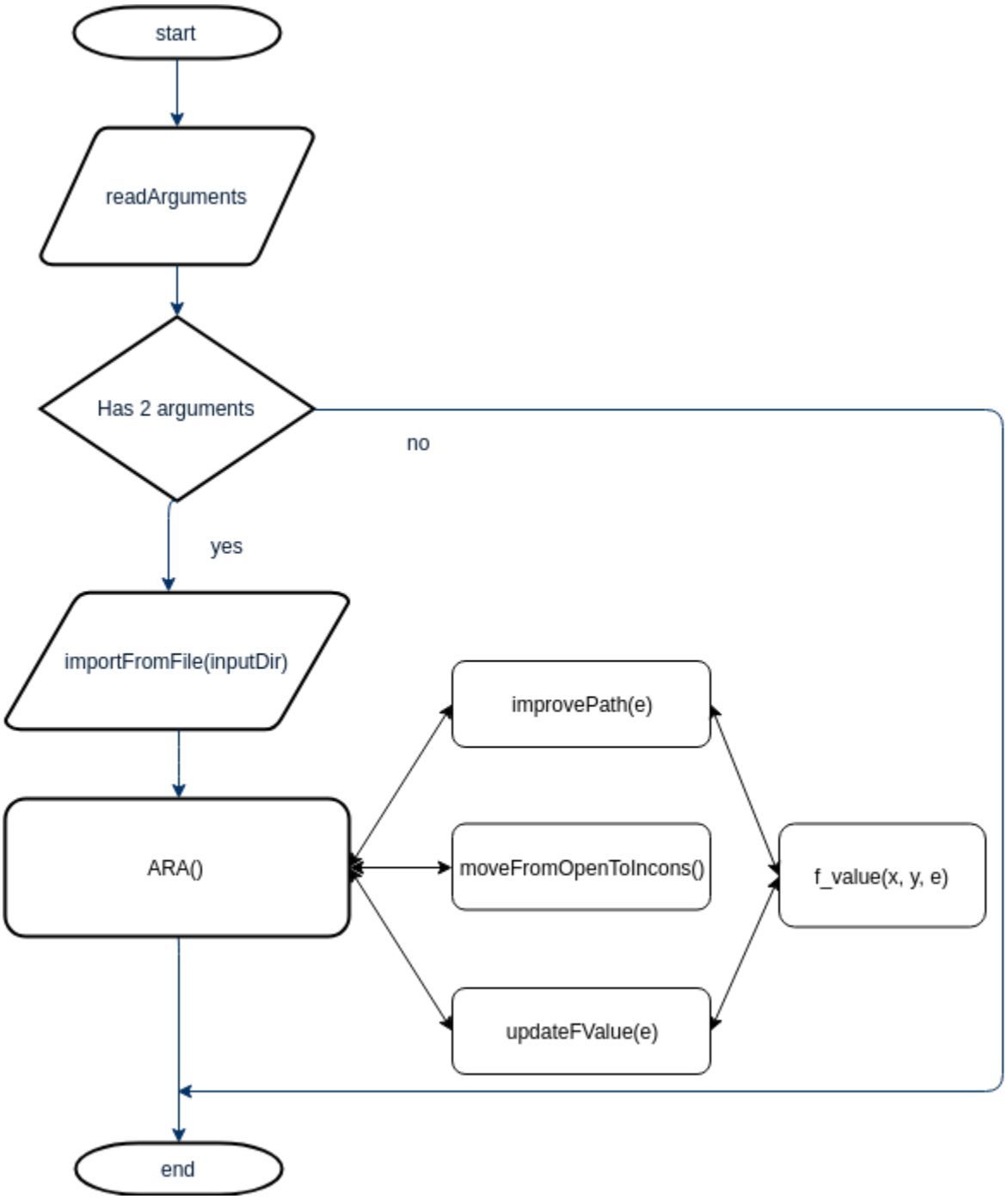
1. Thuật toán A*



- readArguments(): đọc và trả về đường dẫn input và output
- inputFromFile(inputDir): đọc dữ liệu từ file có đường dẫn inputDir (n, Sx, Sy, Gx, Gy, bảng n x n)
- AStar(Sx, Sy, Gx, Gy): tìm đường đi ngắn nhất dựa trên thuật toán A* giữa (Sx, Sy) và (Gx, Gy)
- heuristic(x, y): giá trị h(x, y)

- listCellsCanGoFrom(x, y): những ô mà có thể cho vào hàng đợi OPEN trong những ô lận cận (x, y)
- exportToFile(ans, outputDir): xuất kết quả lưu trong ans ra file có đường dẫn outputDir

2. Thuật toán ARA*



- readArguments(): đọc và trả về đường dẫn input
- inputFromFile(inputDir): đọc dữ liệu từ file có đường dẫn inputDir (n, Sx, Sy, Gx, Gy, bảng n x n, thời gian chạy tối đa)
- ARA(): tìm đường đi ngắn nhất dựa trên thuật toán A* giữa (Sx, Sy) và (Gx, Gy)
- improvePath(e): mở rộng đường đi với giá trị e
- moveFromOpenToIncons(): di chuyển từ OPEN sang tập INCONS
- updateFValue(e): cập nhật giá trị f ứng với e mới trong các phần tử của tập OPEN
- f_value(x, y, e): trả về f(x, y) = g(x, y) + e * h(x, y)

Ta nên dùng một thread riêng để quản lý thời gian chạy

VII. Cấu trúc dữ liệu đã cài đặt

1. Thuật toán A*

- Lưu các thông tin bài toán
  - Số nguyên n
  - Tọa độ ô bắt đầu (Sx, Sy)
  - Tọa độ ô đích (Gx, Gy)
  - Bảng arr có độ lớn n x n lưu lại trạng thái (1: nếu là chướng ngại vật, 0: ngược lại)
- Lưu các thông tin tại từng ô (x, y)
  - g(x, y): giá trị g(x, y) hiện tại
  - h(x, y): giá trị heuristic (có thể tính trước)
  - f(x, y) = g(x, y) + h(x, y)
- Tập hợp heap
  - Lưu những ô đã mở rộng và chưa lấy ra khỏi hàng đợi
  - Lưu dưới dạng priority queue (tìm những ô nằm trong OPEN mà có giá trị f(x, y) nhỏ nhất)
- Tập hợp visited
  - Lưu những ô đã mở rộng và lấy ra khỏi hàng đợi
  - Chỉ cần lưu dưới dạng set (nhanh hanh priority queue)
- Kết quả
  - Là một tuple gồm 2 giá trị: số ô đi qua và tọa độ của những ô đó

2. Thuật toán ARA*

- Lưu các thông tin bài toán
  - Số nguyên n
  - Tọa độ ô bắt đầu (Sx, Sy)

    - Tọa độ ô đích (Gx, Gy)
    - Bảng arr có độ lớn n x n lưu lại trạng thái (1: nếu là chướng ngại vật, 0: ngược lại)
    - Thời gian cho phép
- Lưu các thông tin tại từng ô (x, y)
    - g(x, y): giá trị g(x, y) hiện tại
    - Lưu dưới dạng set
- Tập hợp OPEN
    - Lưu những ô đang xét với e hiện tại và chưa lấy ra khỏi hàng đợi
    - Lưu dưới dạng priority queue (tìm những ô nằm trong OPEN mà có giá trị f(x, y, e) nhỏ nhất)
- Tập hợp INCONS
    - Lưu những ô mà không thích nhất quán, dùng cho bước giảm e sau
    - Lưu dưới dạng set
- Tập hợp CLOSED
    - Lưu những ô đã mở rộng và lấy ra khỏi OPEN
    - Chỉ cần lưu dưới dạng set

VIII. Mô tả thuật toán chính, những thay đổi và cải tiến

1. Thuật toán chính

Bài toán:

Cho bảng n x n, trong đó có một ô bắt đầu và một ô đích, có thể có nhiều ô không thể đi qua được. Bằng cách di chuyển từ một ô sang 8 ô kề cận tìm đường đi ngắn nhất đi từ ô nguồn đến ô đích

Thuật toán:

- B1: Với mỗi ô ngoài những ô có chướng ngại vật ta không thể đi qua được ta đánh giá hàm heuristic của ô đó (khoảng cách euclidean, giá trị lớn nhất của khoảng cách theo hàng và theo cột, …)
- B2: Gọi g(x, y) là đường đi ngắn nhất hiện tại đi từ ô (Sx, Sy) đến ô (x, y)
- B3: Xây dựng một hàng đợi ưu tiên theo giá trị f(x, y) = g(x, y) + h(x, y). Đầu tiên khởi tạo chỉ có một phần tử là (Sx, Sy) gọi là OPEN
- B4: Xây dựng một tập hợp lưu lại những ô đã mở rộng và lấy ra khỏi hàng đợi ưu tiên gọi là CLOSED
- B5: Lấy phần tử có giá trị f(x, y) nhỏ nhất từ OPEN và đẩy vào CLOSED. Nếu OPEN là tập rỗng thì tới B7

- B6: Xét những ô liền kề với nó mà ta có thể đi qua (không chứa chướng ngại vật) và chưa nằm trong tập CLOSED, cập nhật giá trị g(x, y) và h(x, y) của những ô đó rồi đẩy vào hàng đợi OPEN. Quay lại B5
- B7: Giá trị g(Gx, Gy) của ta bây giờ chính là giá trị đường đi ngắn nhất cần tìm

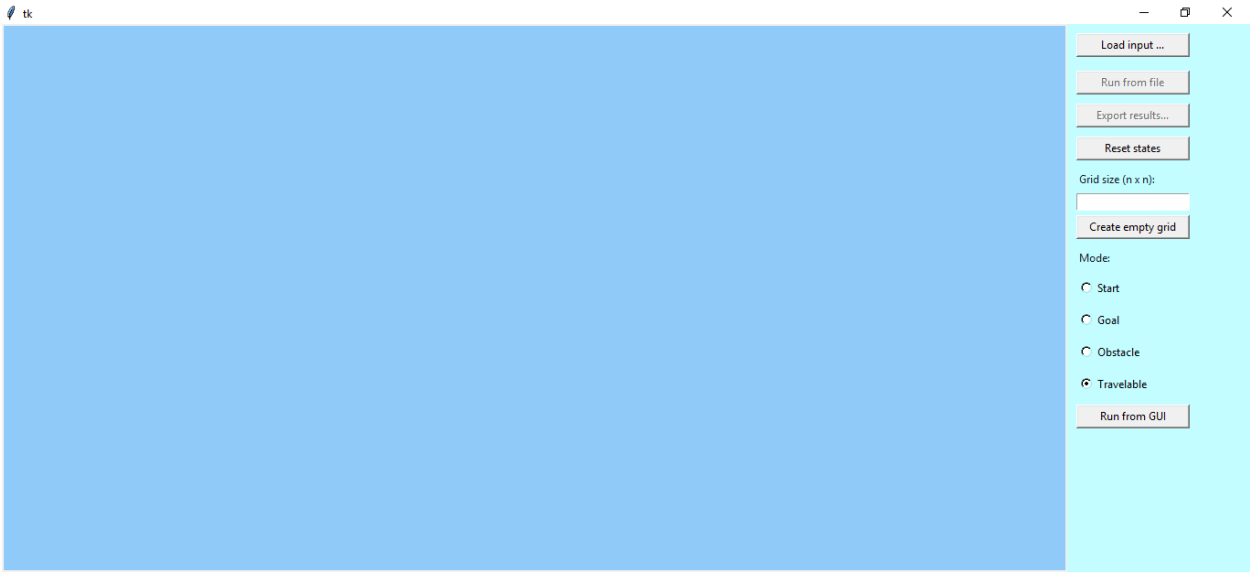
2. Thay đổi và cải tiến

- Với thuật toán A*: Trong trường hợp hàm heuristic chỉ là chấp nhận được chứ chưa phải là thích hợp, nhất quán (consistent) thì khi lấy ra một phần tử từ tập OPEN thì khi xét các ô kề cận nó ta có thể không cần kiểm tra ô kề cận không nằm trong tập CLOSED → có thể dẫn tới thời gian thực hiện lâu hơn nhưng đảm bảo kết quả là tối ưu
- Với thuật toán ARA*: Trong trường hợp hàm heuristic đã thích hợp, nhất quán (consistent) thì không có phần tử nào nằm trong tập INCONS → ta có thể bỏ đi tập hợp INCONS

IX. Hướng dẫn sử dụng GUI

Khởi chạy bằng cách

- Dùng cmd thay đổi đường dẫn tới thư mục chứa file thực thi 1612843_1612869_Lab01_GUI.exe
- Chạy lệnh ./ 1612843_1612869_Lab01_GUI.exe



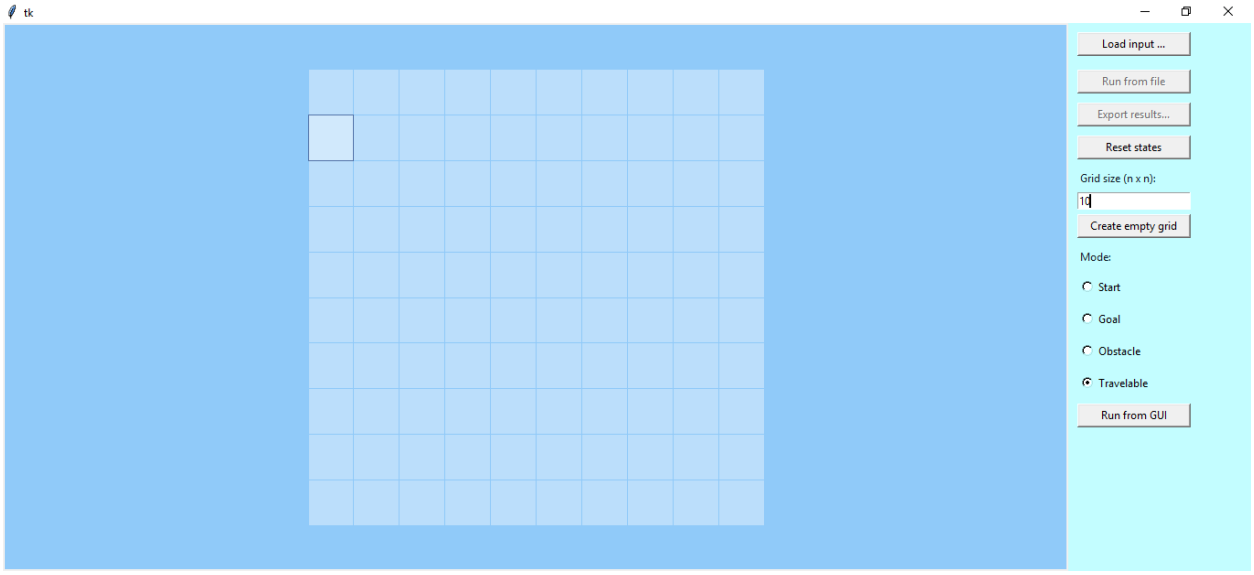
Giao diện ban đầu

1. Xử lý trên file

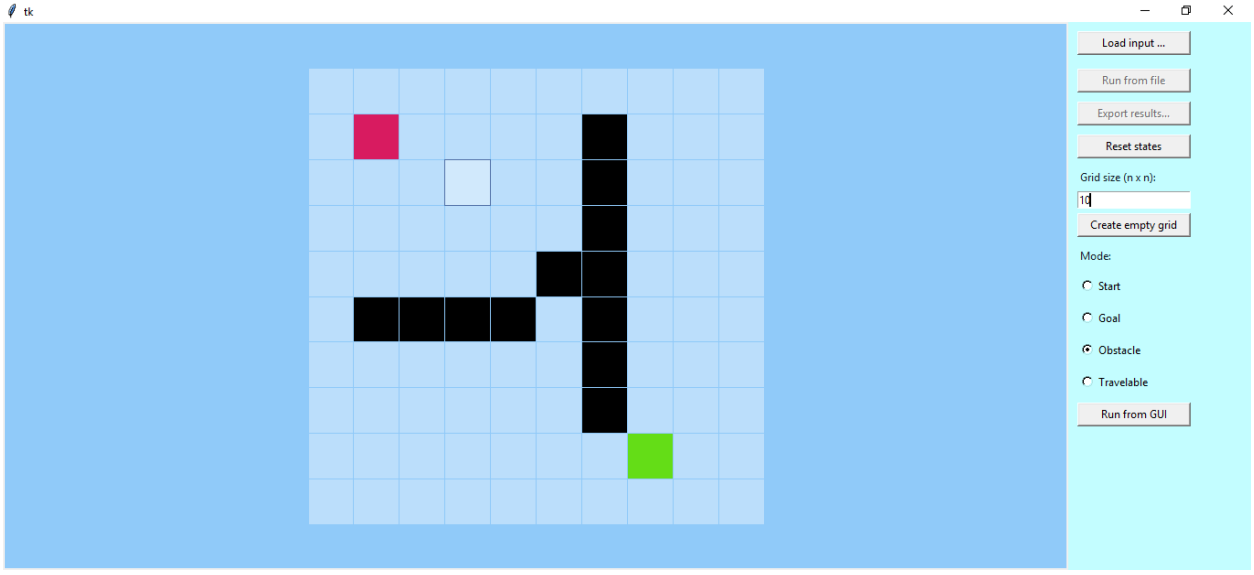
- Ta có thể nhập dữ liệu từ file bằng cách nhấn button **Load file…** Khi đó giao diện sẽ vẽ ra cấu hình hiện tại.
- Nhấn vào nút **Run from file** để chạy A*
- Xuất kết quả ra file bằng cách nhấn **Export results…**

2. Nhập từ GUI

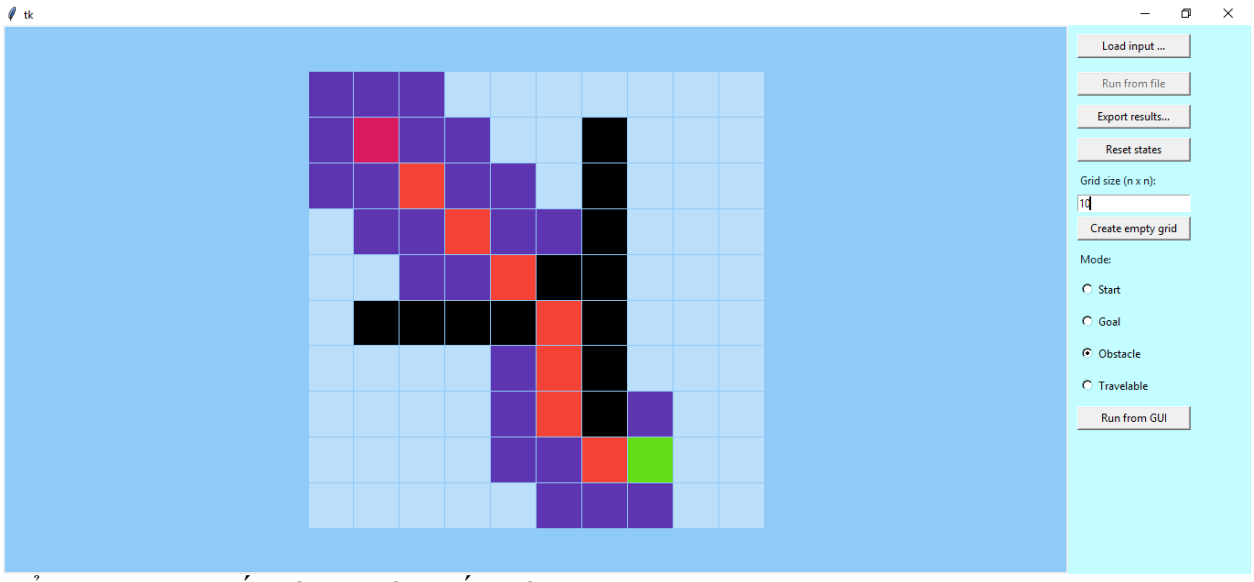
- Nhập n và nhấn nút **Create empty grid**



- Vẽ đỉnh nguồn (Start), đích (Goal), các ô chướng ngại vật (Obstacles). Nếu ô nào ta muốn xóa thông tin đã ghi thì chọn option Travelable rồi nhấn hoặc rê qua ô đó



- Nhấn nút **Run from GUI** để chạy A*



- Để chạy với bộ cấu hình khác nhấn nút **Reset states**

X. Tài liệu tham khảo

- http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html
- https://papers.nips.cc/paper/2382-ara-anytime-a-with-provable-bounds-on-sub-optimality.pdf