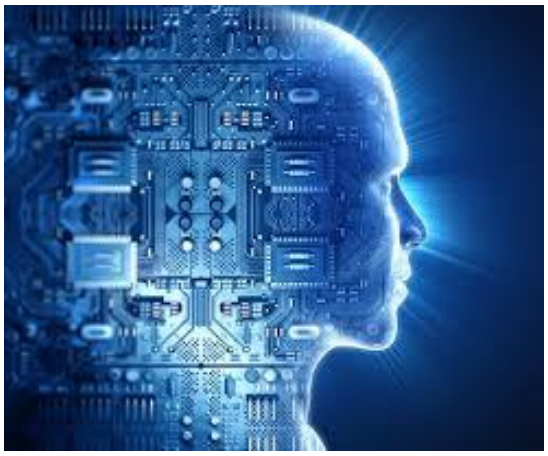
Trường Đại học Khoa học Tự nhiên – Đại học Quốc gia Thành phố Hồ Chí Minh

Khoa Công nghệ thông tin

MÔN CƠ SỞ TRÍ TUỆ NHÂN TẠO

BÁO CÁO LAB-01:

TÌM KIẾM HEURISTIC VỚI A*



Giảng viên lý thuyết:

TS. Lê Hoài Bắc

Giảng viên thực hành:

ThS. Lê Ngọc Thành

ThS. Nguyễn Ngọc Thảo

ThS. Châu Ngọc Phương

MỤC LỤC

DANH MỤC HÌNH

I. Thông tin thành viên nhóm:

STT	Họ tên	MSSV	Email
1	Nguyễn Thị Tình	1612703	1612703@student.hcmus.edu.vn
2	Phan Minh Sơn	1612888	1612888@student.hcmus.edu.vn

II. Tổng quan đồ án:

- Cho vị trí bắt đầu (Start) và vị trí kết thúc (Goal), bài toán tìm đường đi là chỉ ra một đường thẳng nối hai điểm này.
- Trong đồ án, tìm đường đi ngắn nhất bằng thuật toán A* với heuristic là khoảng cách Euclidian và khoảng cách lớn hơn giữa dx và dy so với điểm Goal.
- Sử dụng thuật toán ARA* để giảm thời gian tìm kiếm dù không tìm được đường đi tối ưu nhất.
- Xây dựng giao diện đồ họa người dùng (GUI).

III. Chi tiết thực hiện:

III. 1 . Cấu trúc dữ liệu:

- **Class Node:** *# lưu trữ dữ liệu cho một ô trên bản đồ đường đi.*

  Node.eps; *# static variable lưu giá trị epsilon cho tất cả các Node.*

  Node.Goalxy; *# static variable lưu tọa độ Goal cho tất cả các Node.*

  Node.ID; *# static variable lưu loại heuristic sử dụng:*

  0: Max (dx, dy), 1: Khoảng cách Euclidean, 2: Khoảng cách Manhattan.

  x; *# hoành độ.*

  y; *# tung độ.*

  parent; *# lưu vị trí cha dẫn tới vị trí đang xét.*

  gValue; *# khoảng cách từ Start đến vị trí đang xét.*

  heuristic(); *# phương thức tính heuristic cho khoảng cách từ vị trí tới Goal.*

  fValue0(); *# gValue + heuristic().*

  fValue(); *# gValue + epsilon * heuristic().*

  isSamePosition (other); *# kiểm tra vị trí đang xét và other có cùng tọa độ hay không (trả về true / false).*

  isVisited (List); *# kiểm tra vị trí đó đã được viếng thăm trong List chưa (trả về true / false), nếu đã viếng thăm, cập nhật gValue cho vị trí.*

  positionVisited(List); *# trả về vị trí xuất hiện trong List, nếu chưa xuất hiện trả về -1.*

  operator >, <, >=, <=, = ; *# override theo giá trị fValue().*

- **Class GUI(frame):** *# class chính để xây dựng giao diện đồ họa người dùng.*
  Map[]; *#lưu bản đồ đường đi*
  filepath; *#lưu đường dẫn tập tin để đọc file hoặc lưu file.*
  Start; *#lưu Node cho vị trí Start.*
  Goal; *#lưu Node cho vị trí Goal.*
  way[]; *#lưu đường đi tìm được.*
  Các button, Label, ComboBox để xử lý vẽ giao diện, bản đồ, đường đi, …
  Các phương thức bắt sự kiện …
  AStar(); # phương thức tìm đường đi bằng thuật toán A.*
  ARAStar(); *# phương thức tìm đường đi bằng thuật toán ARA*.*
  Các phương thức hỗ trợ khác …
  Ở đây class GUI là class lớn bao hàm cả chương trình nên nhóm chỉ trình bày sơ lược các thuộc tính và phương thức chính trong class. Giải thích phương thức hay các thuộc tính dễ được trình bày thêm trong các phần sau: phần yêu cầu 2 (III. 3), phần thuật toán đã thực hiện (III. 4), phần giao diện GUI (III. 5) và hướng dẫn sử dụng (V).

III. 2 . Sơ đồ biểu diễn hệ thống phần mềm:

III.2.1. Hàm tính heuristic:

- Là phương thức trong class Node.

- Phương thức có dạng: def heuristic()

- Chức năng: Tính heuristic cho thuật toán.

- Output: Trả về một trong ba giá trị heuristic: Max (dx, dy), Khoảng cách Euclidean, Khoảng cách Manhattan.

III.2.2. Hàm tìm kiếm A*:

- Là phương thức trong class GUI.

- Phương thức có dạng: def AStar().

- Chức năng: Tìm kiếm đường đi ngắn nhất dựa trên thuật toán A* với heuristic.

III.2.3. Hàm tìm kiếm ARA*:

- Là phương thức trong class GUI.

- Phương thức có dạng: def ARAStar().

- Chức năng: Tìm kiếm đường đi theo thuật toán ARA* với heuristic.

III.2.4. Hàm improvePath:

- Là phương thức trong class GUI.

- Phương thức có dạng: def improvePath(Open, Closed, Incons).

- Chức năng: tìm ra đường đi tối ưu hơn ứng với mỗi epsilon khác nhau.

- Input: 3 list Node trong đó Open là dạng hàng đợi ưu tiên dựa trên fValue của Node.

- Output: True nếu tìm ra đường đi tối ưu hơn, ngược lại trả về False.

III.2.5. Hàm đọc file:

- Là hàm trong chương trình.

- Hàm có dạng: def readFile(fname).

- Chức năng: Nhận input từ file.

- Input: tên file input (fname).

- Output: Node Start, Goal và bản đồ đường đi Map.

III.2.6. Hàm ghi kết quả ra file:

- Là hàm trong chương trình.

- Hàm có dạng: def writeFile(fname, way, Map, id).

- Chức năng: ghi kết quả ra file, nếu không tìm thấy đường đi thì kết quả là -1, ngược lại ghi số bước thực hiện, danh sách vị trí đường đi, và bản đồ đường đi được mô hình hóa lại (o: chướng ngại vật, -: không có vật cản, x: đường đi, S: vị trí Start, G: vị trí Goal).

III. 3 . Yêu cầu 2:

III.3.1. Đề xuất Heuristic mới:

- **Heuristic mới đề xuất:** Khoảng cách lớn hơn giữa dx và dy của điểm đang xét so với điểm Goal.

  s.heuristic() = max (abs (Goal.x – s.x), abs (Goal.y – s.y));

  với s là một Node bất kì.

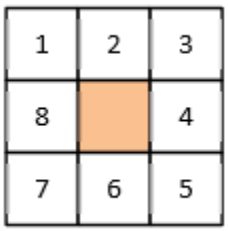
  Hay **Heuristic = Max (dx, dy).**

- **Chứng minh Heuristic chấp nhận được:**

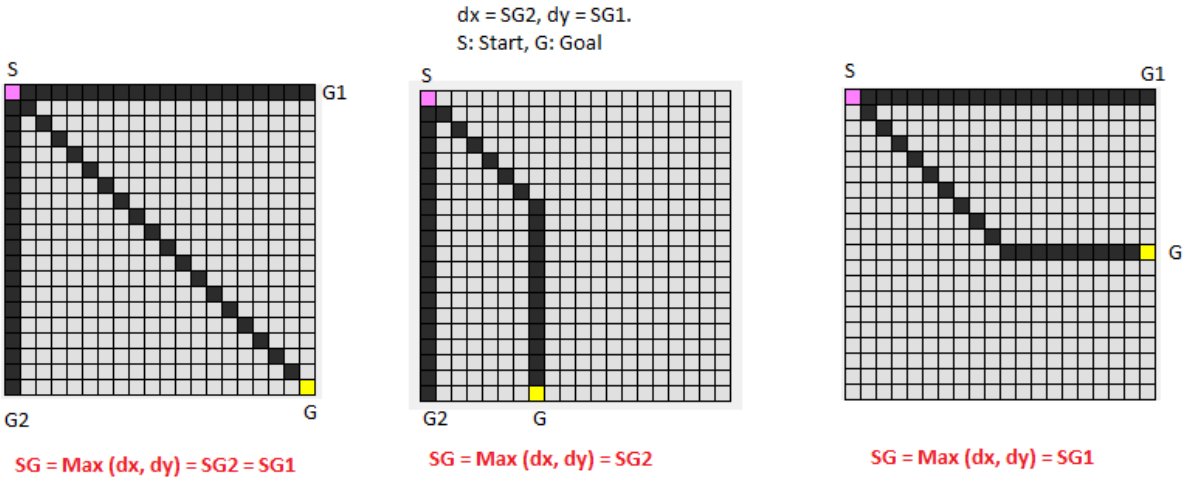
  + Trường hợp đường đi ngắn nhất giữa Start và Goal không có chướng ngại vật:

  Ta có: heuristic bằng với đường đi.

Giải thích: Vì chi phí đi theo đường chéo (đến các ô 1, 3, 5,7) bằng chi phí đi theo đường thẳng (đến các ô 2, 4, 6, 8) nên trong trường hợp này đường đi chính bằng heuristic. Hình vẽ minh họa để thấy rõ ràng hơn:



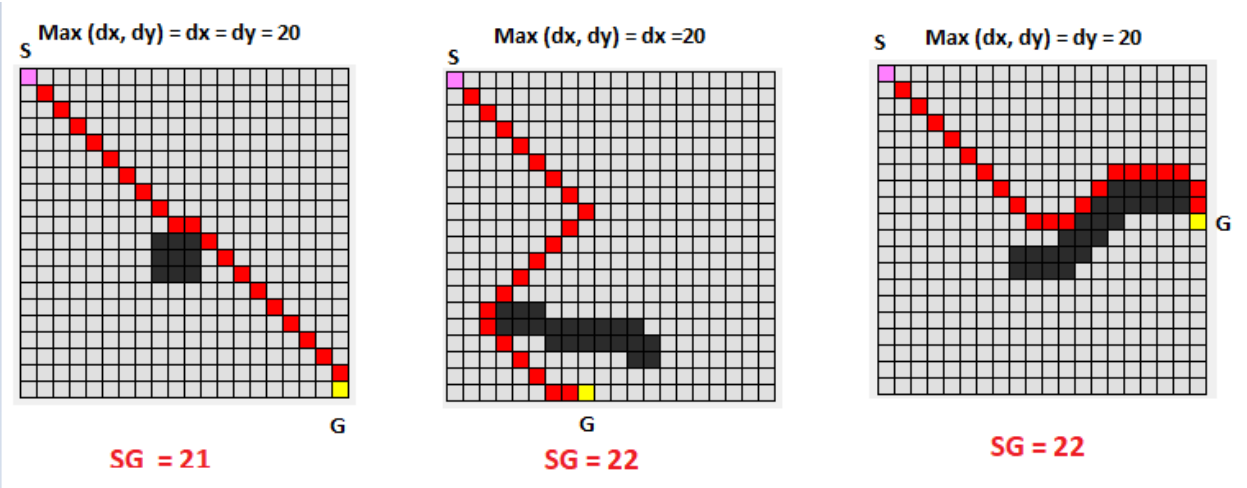
Hình 1. Thứ tự mở rộng các ô



Hình 2. Chứng minh heuristic chấp nhận được 1

+ *Trường hợp đường đi ngắn nhất giữa Start và Goal có chướng ngại vật:*

Ta có trong trường hợp có chướng ngại vật thì đường đi từ Start đến Goal bao giờ cũng lớn hơn đường đi từ Start đến Goal cùng vị trí trong trường hợp không có chướng ngại vật (Dễ dàng thấy theo bất đẳng thức tam giác).



Hình 3. Chứng minh heuristic chấp nhận được 2

Mà heuristic luôn bằng đường đi trong trường hợp không có chướng ngại vật.

⇨ Heuristic < chi phí đường đi thực tế.

⇨ Theo 2 trường hợp, heuristic <= chi phí thực tế.

⇨ Heuristic chấp nhận được.

III.3.2. Tìm hiểu thuật toán ARA*:

- **Thuật toán A***: nhận một heuristic s.heuristic() hợp lý là khoảng cách từ vị trí đang xét đến Goal. Xét mỗi điểm s1 là lân cận của s. Ta có s.heuristic() <= s1.heuristic() + 1 nếu s ≠ Goal và s.heuristic() = 0 nếu s = Goal. Heuristic gọi là chấp nhận được thì giá trị của heuristic không lớn hơn giá trị thực tế của khoảng cách từ vị trí đang xét tới Goal. Tăng cường trọng số cho heuristic (bằng cách sử dụng ε * s.heuristic() với $\varepsilon > 1$ thì số điểm được mở rộng ít hơn và và tìm kiếm nhanh hơn, tuy nhiên khi đó heuristic được chấp nhận một cách không hợp lệ chính thức, vì vậy đường đi tìm được không chắc là tối ưu nhất. Điều kiện dừng của A* là điểm Goal được mở rộng.
- **ARA* (Anytime Reparing A*)**: đây là thuật toán hiệu quả hơn với bất kì thời gian thực thi nào với bài toán tìm kiếm sử dụng heuristic bằng giải thuật A* với hệ số ε tăng cường cho heuristic (*inflated heuristic*) bằng việc sử dụng lại tìm kiếm đường đi trước đó để tối ưu hóa chi phí tìm kiếm đường đi ở các lần thực thi tiếp theo.
- ARA* thực thi A* nhiều lần với giá trị ban đầu của ε và sau đó giảm dần giá trị của ε cho đến khi $\varepsilon = 1$. Nếu mỗi lần giảm ε thực thi A* thì thời gian tìm kiếm lớn. Bên cạnh đó, ARA* sử dụng lại kết quả tìm kiếm trước đó để giảm thời gian, chi phí tìm kiếm và tối ưu bài toán hơn.

- Những điểm gọi là bất hợp lý địa phương (*local inconsistency*) là những điểm khi mở rộng lại thì gValue của nó giảm dần, những điểm này được đưa vào tập Open và sau đó nếu được mở rộng thì xóa khỏi Open. A* với $\varepsilon > 1$ thì không đảm bảo mỗi điểm được mở rộng tối đa một lần do đó, ở ARA* có tập Closed để chứa những điểm đã mở rộng để hạn chế việc mở rộng một điểm nhiều lần. Một điểm local inconsistency chỉ được thêm vào Open khi nó chưa được mở rộng trước đó. Tuy nhiên Open lại không chứa tất cả các điểm local inconsistency vì vậy có thêm tập Incons để chứa các điểm local inconsistency đã được mở rộng rồi để truy vết và sử dụng lại ở lần tìm kiếm tiếp theo.
- Điều kiện dừng ở mỗi lần tìm kiếm là giá trị gValue Goal nhỏ hơn hoặc bằng giá trị fValue nhỏ nhất của mọi điểm trong tập Open (khác với điều kiện dừng của A*).

III.3.3. Giao diện GUI:

- Chương trình hỗ trợ giao diện đồ họa người dùng GUI cho cả thuật toán A* và ARA* với cả 3 heuristic: Max (dx, dy) (heuristic nhóm đề xuất), Khoảng cách Euclid, Khoảng cách Manhattan.
- Giao diện cụ thể được trình bày trong phần Hướng dẫn sử dụng.

III. 4 . Thuật toán đã thực hiện:

III.4.1. Thuật toán A*:

Mã giả cho thuật toán A*:

- B1: Gán Start.gValue = 0, Open = [].
- B2: Thêm Start vào tập Open với fValue = gValue + ε * heuristic().
- B3: While (Goal chưa được mở rộng)
  Lấy điểm s thuộc Open sao cho fValue của s là nhỏ nhất.
  Xóa s khỏi Open.
  Xét mỗi điểm s1 lân cận của s:
  Nếu s1 chưa được viếng thăm trước đó:
  Gán s1.gValue = ∞.
  Nếu s1.gValue > s.gValue + 1:
  Gán s1.gValue = s.gValue +1.
  Thêm s1 vào Open dạng hàng đợi ưu tiên với fValue().

II. 3. 2. Thuật toán ARA*:

Mã giả cho ARA*:

- **Hàm improvePath():**

While Goal.fValue() > min (s thuộc Open. fValue())
Lấy s min ra khỏi Open.
Thêm s vào Closed.
Xét mỗi điểm s1 là lân cận của s:
    Nếu s1 chưa được viếng thăm:
        Gán s1.gValue = ∞.
    Nếu s1.gValue > s.gValue +1:
        Gán s1.gValue = s.gValue + 1.
        Nếu s1 không ∈ Closed:
            Thêm s1 vào Open dạng hàng đợi ưu tiên với fValue().
        Ngược lại Thêm vào Incons.

- **ARA* (main function):**

- B1: Gán Goal.gValue = ∞, Start.gValue = 0.
- B2: Open = Closed = Incons = ∅.
- B3: Thêm Start vào Open dạng hàng đợi Ưu tiên với fValue().
- B4: improvePath().
- B5: $\varepsilon 1$ = min (ε, Goal.gValue() / (min (s ∈ Open, Incons). (gValue + heuristic)).
- B6: Xuất đường đi với ε_0.
- B6: while ($\varepsilon 1 > 1$)ε
    Giảm ε: ε -= $\Delta\varepsilon$.
    Chuyển các điểm thuộc Incons qua Open dạng hàng đợi ưu tiên.
    Closed = ∅.
    improvePath().
    $\varepsilon 1$ = min (ε, Goal.gValue() / (min (s ∈ Open, Incons). (gValue + heuristic)).
    Xuất đường đi với ε.

IV. Hướng dẫn sử dụng:

IV. 1. Đối với giao diện Console:

- **Tên chương trình: 1612703_1612888_Lab01_Console.exe**
- **Câu lệnh:**

- Thuật toán A*: <tên chương trình> <input> <output> <Heuristic>
- Thuật toán ARA*:
<tên chương trình> <input> <output> <Heuristic> <Epsilon> <DeltaEpsilon> <tmax>

- **Trong đó:**

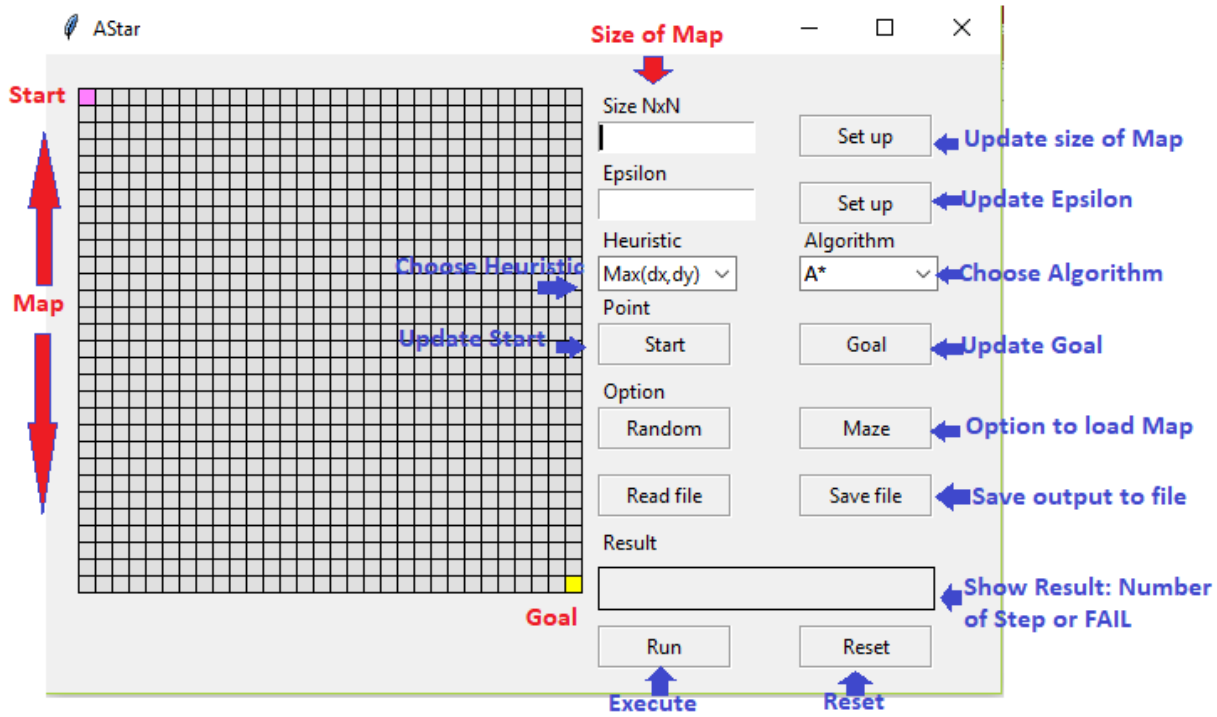
+ input.txt: đường dẫn tập tin input.
+ output.txt: đường dẫn tập tin output để lưu kết quả.
+ Heuristic: 0: Max (dx, dy), 1: Khoảng cách Euclid, 2: Khoảng cách Manhattan.
+ Epsilon: số thực > 1 giá trị epsilon ban đầu.
+ DeltaEpsilon: số thực, độ giảm epsilon, nên thuộc khoảng 0.1 – 0.5.
+ tmax: Thời gian tối đa để tìm kiếm với ARA, tính theo ms.

- **Kết quả:**

- A*: ghi kết quả xuống file output với dòng đầu tiên là số bước thực thi, dòng thứ hai là vị trí các ô trên đường đi từ Start -> Goal, các dòng tiếp theo là Map với "-" vị trí trống, "o" vị trí block, "x" đường đi, "S": Start, "G": Goal.
- ARA*: ghi kết quả xuống file ứng với mỗi epsilon, nếu tìm được đường tối ưu hơn thì ghi kết quả tương tự như A*, ngược lại ghi "No way better".

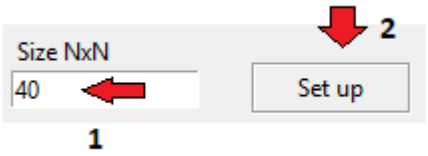
IV.2. Đối với giao diện GUI:

- **Tên chương trình: 1612703_1612888_Lab01_GUI.exe**
- **Giao diện chính:**



Hình 4. Giao diện GUI chính

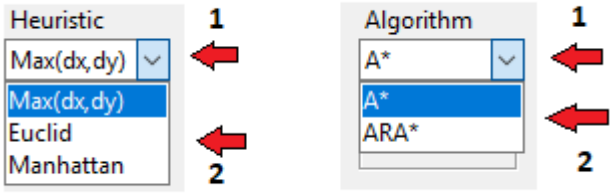
- Nhập kích thước vào Size NxN và chọn Set up để cập nhật, mặc định N = 30.



Hình 5. Update size Map

Hình 6. Update epsilon

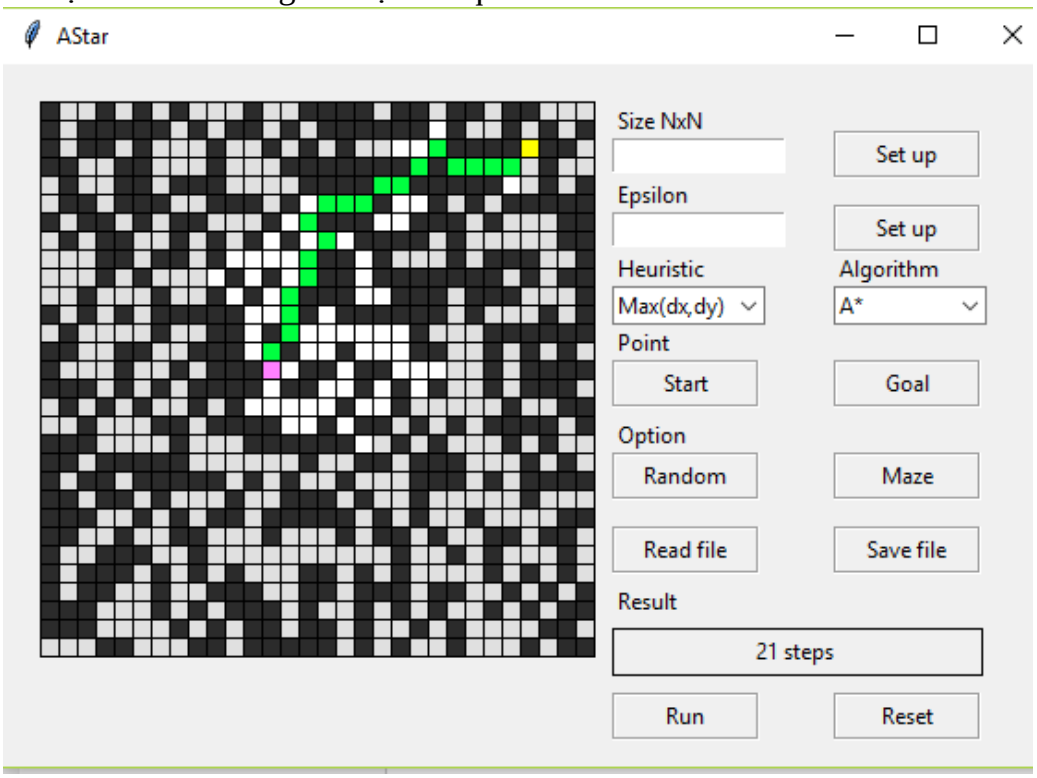
- Nhập epsilon vào ô epsilon và chọn Set up để cập nhật, mặc định epsilon = 5.0.
- Chọn Heuristic tương ứng: Max (dx, dy), Euclid, Manhattan.
- Chọn thuật toán muốn thực thi: A*, ARA*.



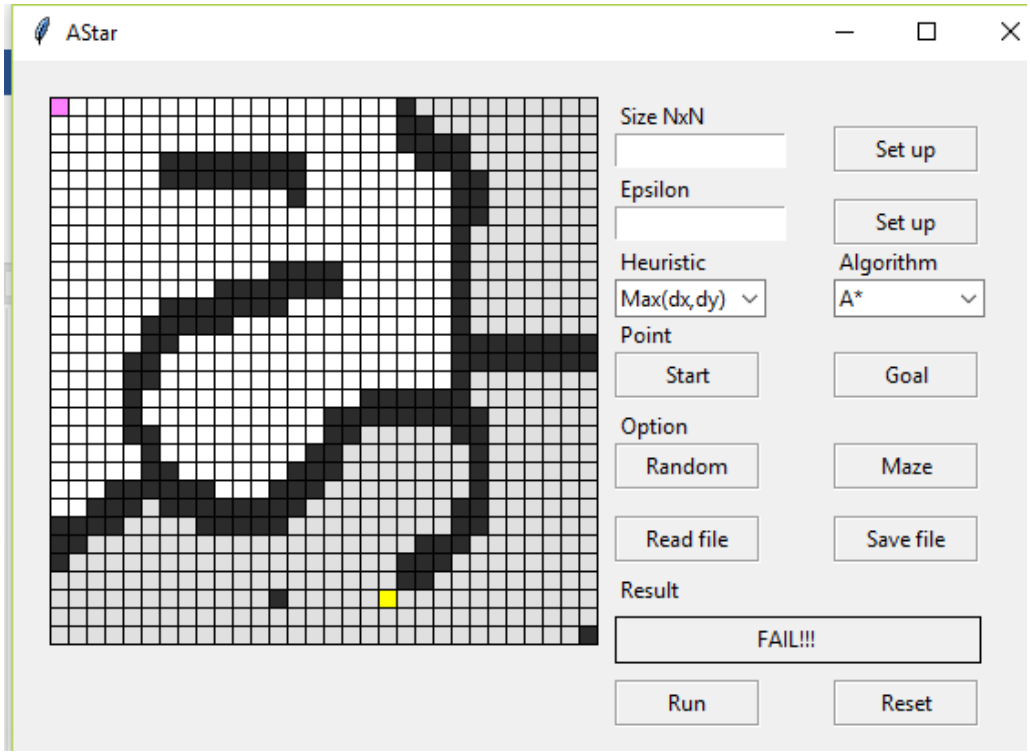
Hình 7. Chọn heuristic và thuật toán

- Cập nhật vị trí Start, Goal: Click button Start/ Goal và chọn vị trí muốn thay đổi trên Map tương ứng. (Start: màu hồng, Goal: màu vàng).
- Vẽ block: Click ô tương ứng trên Map để vẽ block (block có màu đen).
- Chọn option cho Block: Click button Random để sinh block, Start, Goal ngẫu nhiên, click button Maze để sinh Block dạng mê cung, click button Read file để đọc input từ file (truy cập đường dẫn người dùng tự chọn).
- Chọn option Save file: lưu output xuống file.
- Click Run: thực thi với các option đã chọn.
- Click Reset: load lại Map với Start, Goal, size không đổi. Các block bị xóa.
  - **Kết quả:**
- Thuật toán A*: Đường đi có màu xanh lá, Open: các điểm được mở: màu trắng, số bước đi thành công ghi trong ô Result, ngược lại hiện "FAIL!!!".
- Thuật toán ARA*: Đường đi có màu đỏ, đường đi tối ưu cuối cùng: màu xanh lá, Open: màu trắng, Incons: màu xanh dương. Mỗi lần tìm thấy đường đi thành công hiện số bước đi, sau khi thực hiện xong ghi nhận lại đường đi tối ưu nhất, ngược lại hiện "FAIL!!!".

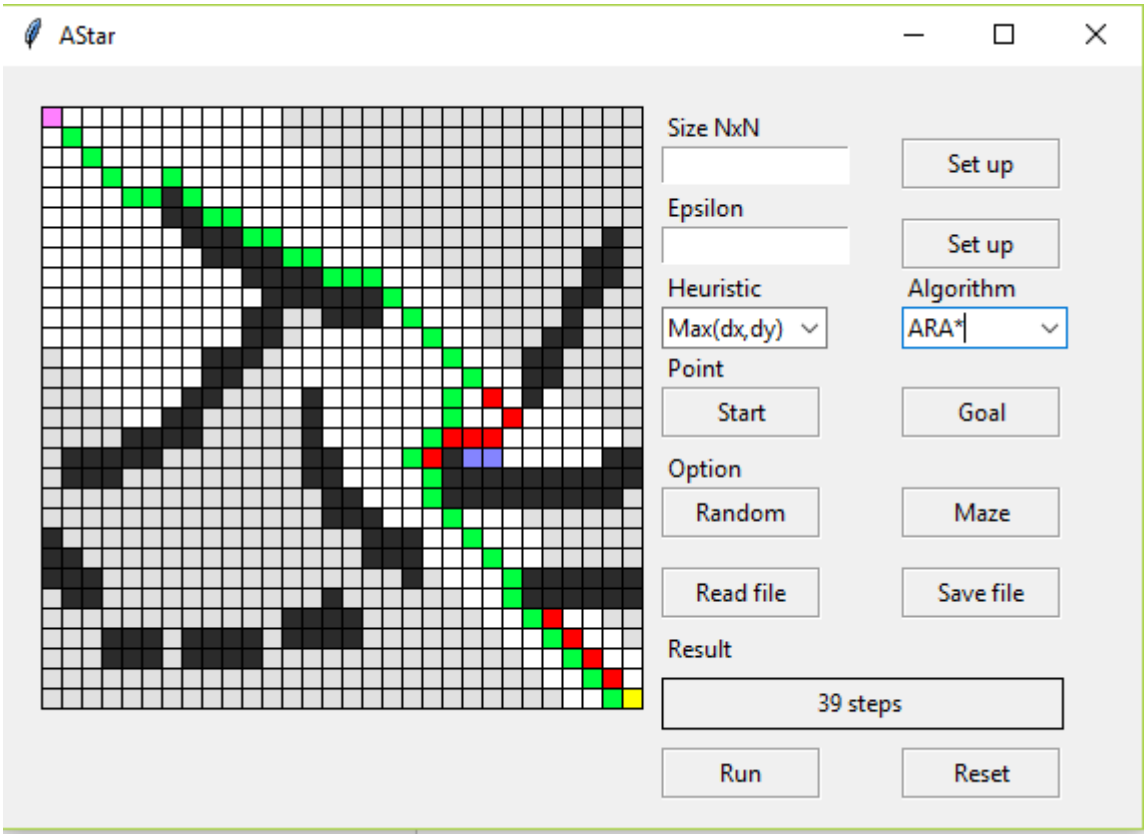
- Một vài hình ảnh giao diện kết quả:



Hình 8. Minh họa GUI A 1*



Hình 9. Minh họa GUI A 2*



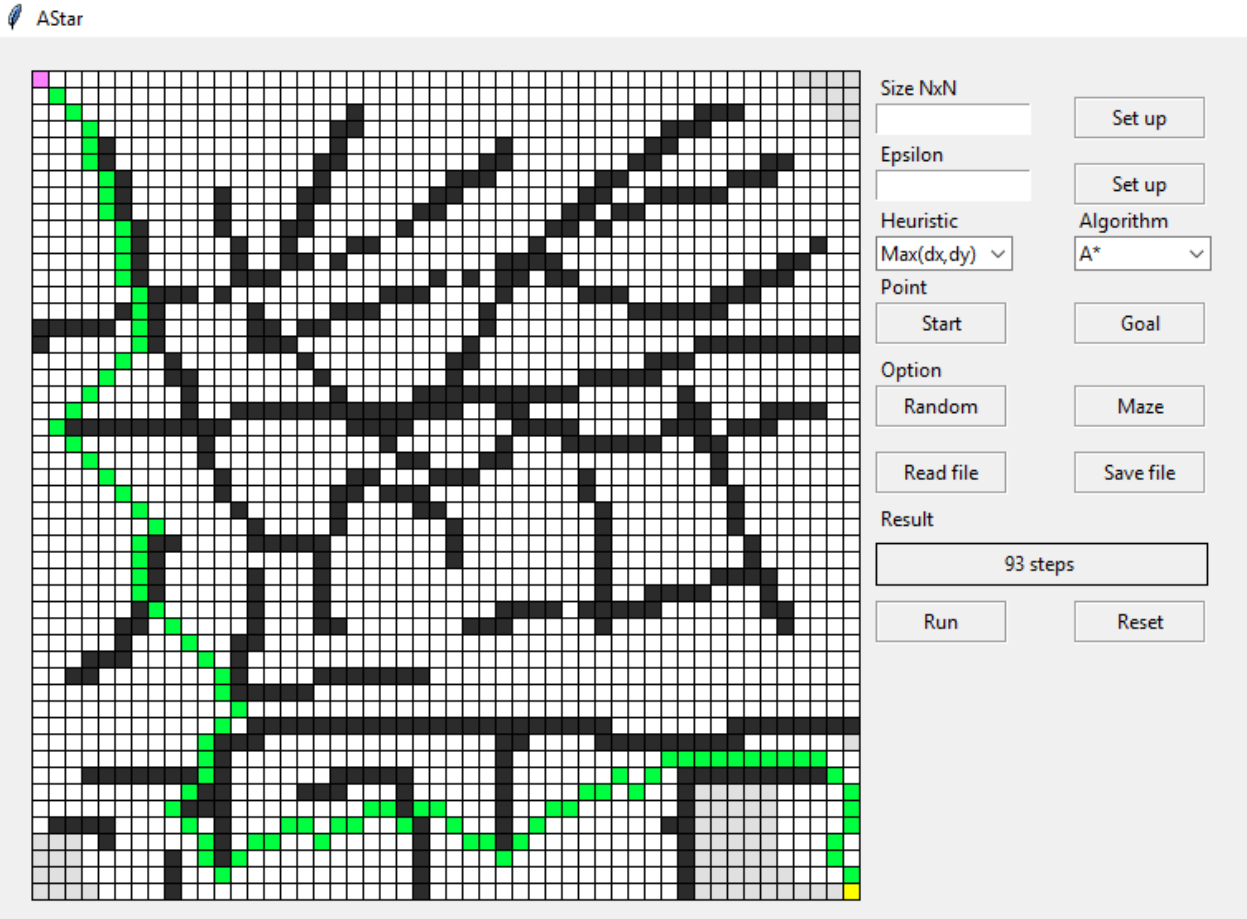
*Hình 10. Minh họa GUI ARA**

V. Các Test case:

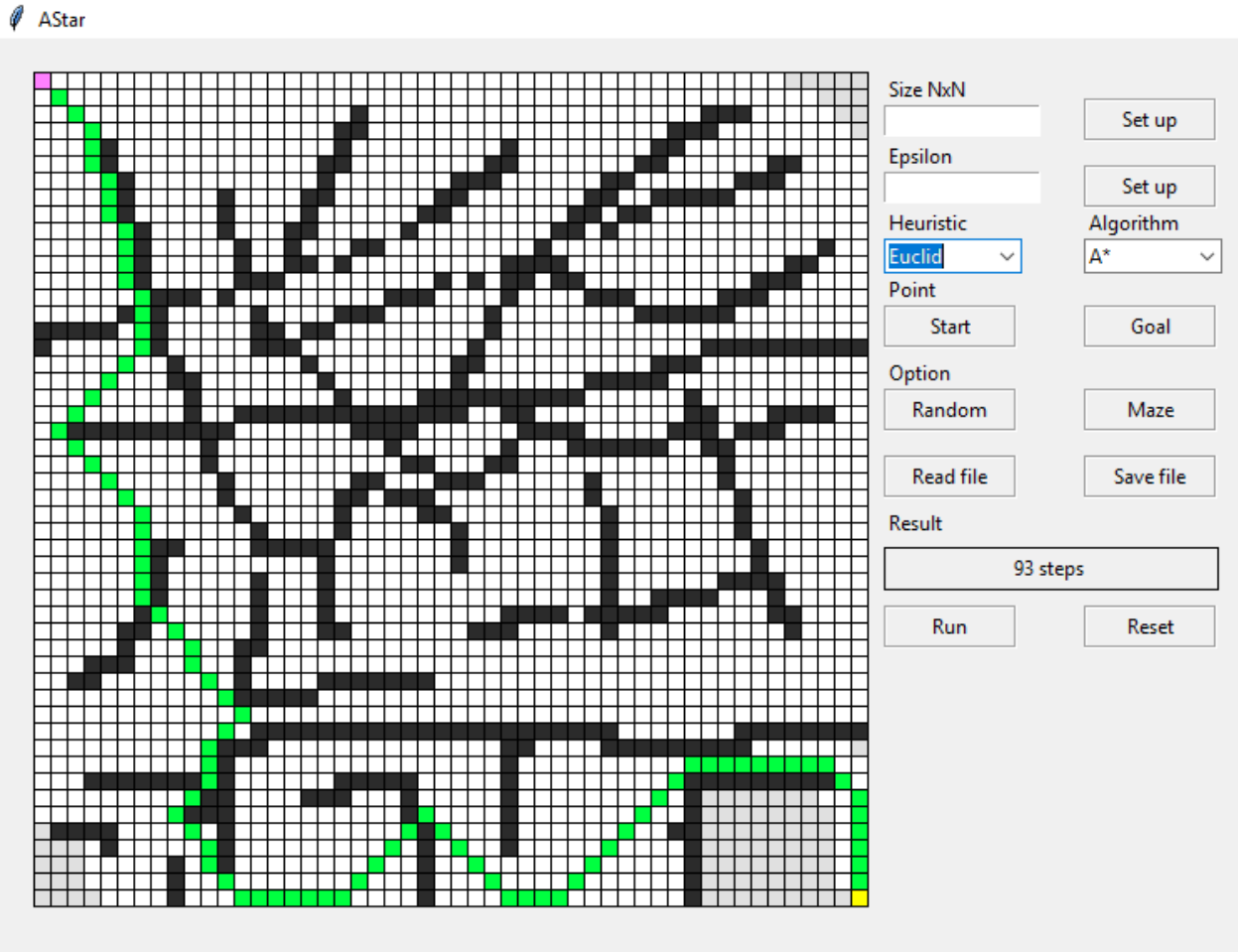
- Ở đây nhóm chỉ trình bày test case ứng với input1.txt.
- Các input còn lại nhóm gửi kèm file input và output với inputXX.txt là input thứ XX, outputXX_AM.txt, outputXX_AE.txt, outputXX_ARAM.txt, outputXX_ARAE.txt lần lượt là kết quả chạy với A* với heuristic Max (dx, dy), Euclid, ARA* với heuristic Max (dx, dy), Euclid.
- Đối với thuật toán ARA*: input1.txt:

Heuristic	Epsilon	Số bước
Max (dx, dy)	5.0	98
	4.5 – 1.5	No way better
	1.0	93
Euclid	5.0	104
	4.5 – 2.0	No way better
	1.5	99
	1.0	93

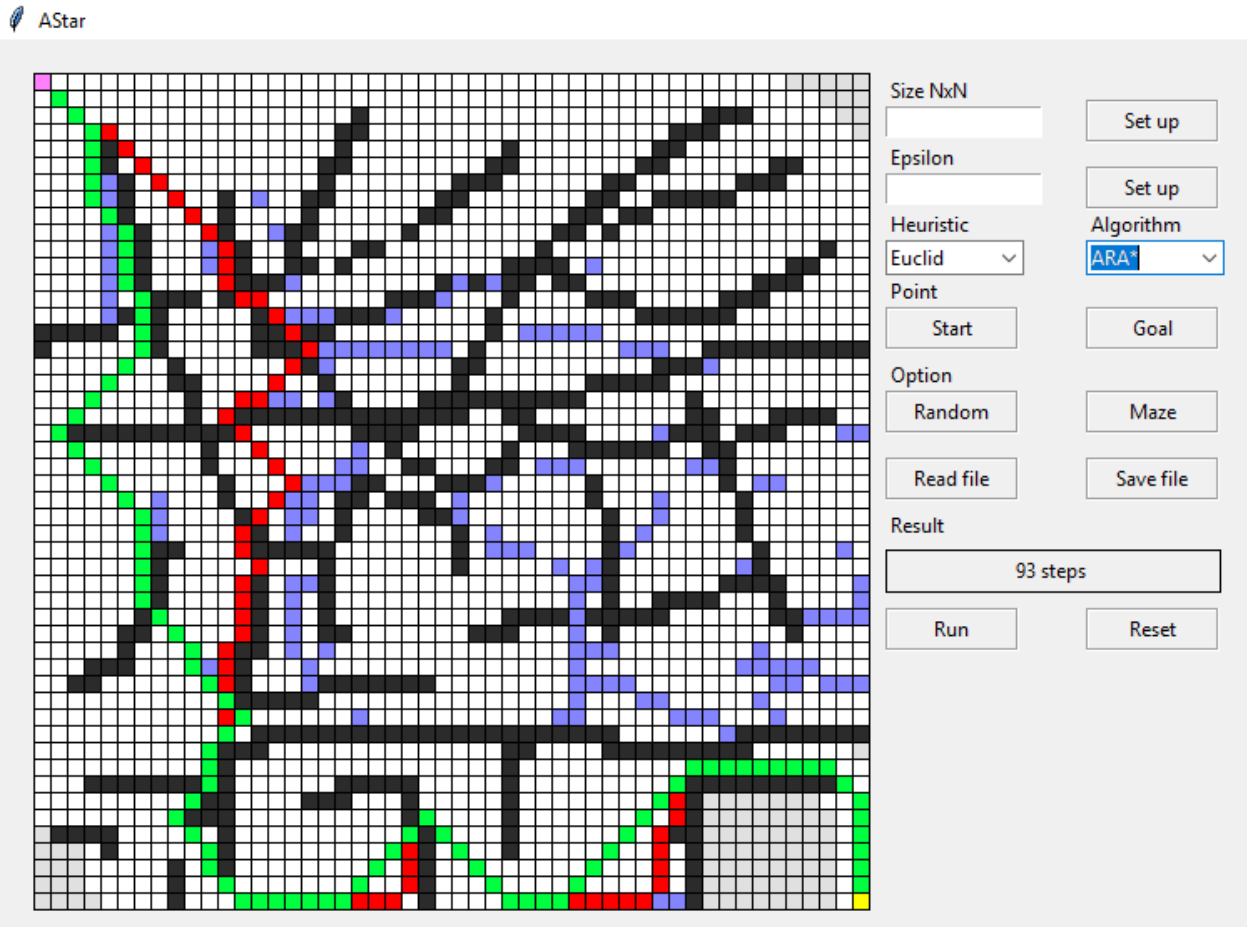
- Hình ảnh chạy bằng GUI:



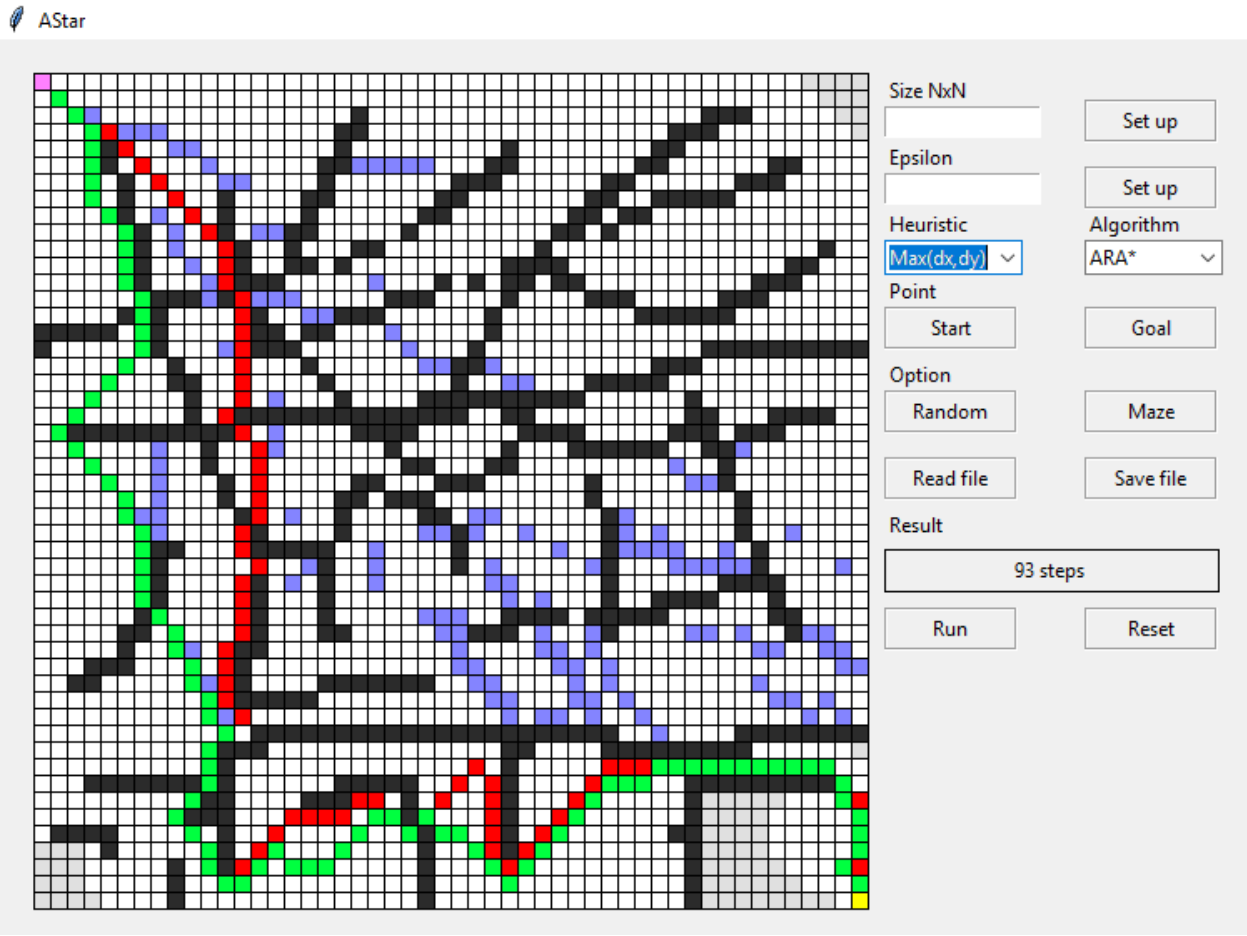
Hình 11. Test case input1. A Max (dx, dy)*



Hình 12. Test case input1. A Euclidean*



Hình 13. Test case input1. ARA Max (dx, dy)*



Hình 14. Test case input1. ARA Euclidean*

VI. Đánh giá mức độ hoàn thành:

VI.1. Bảng đánh giá mức độ hoàn thành:

STT	Nội dung	Mức độ hoàn thành
1	Yêu cầu 1: Cài đặt thuật toán A* với heuristic khoảng cách Euclidean.	Hoàn thành 100%.
2	Yêu cầu 2.1: Tìm heuristic mới và chứng minh heuristic chấp nhận được.	Hoàn thành 100%.
3	Yêu cầu 2.2: Cài đặt thuật toán ARA* với heuristic mới.	Hoàn thành 100%.
4	Yêu cầu 2.3: Cài đặt GUI minh họa thuật toán A*.	Hoàn thành 100%.
5	Cài đặt GUI minh họa thuật toán ARA*.	Hoàn thành 100%.

VI.2. Những vấn đề chưa thực hiện:

- Code cứng, chưa linh hoạt các phương thức, thuộc tính.
- Giao diện chưa hoàn chỉnh.

VII. Phân công công việc:

STT	Người thực hiện	Công việc	Nhận xét	Mức độ hoàn thành
1	Nguyễn Thị Tình	Tìm hiểu và cài đặt ARA*.	Hoàn thành công việc được giao, đúng tiến độ.	100%
2		Chứng minh heuristic chấp nhận được.		100%
3		Viết báo cáo.		100%
4	Phan Minh Sơn	Bulid chương trình ra file.exe	Hoàn thành công việc được giao, đúng tiến độ.	100%
5		Cài đặt A*.		100%
6		Cài đặt GUI.		100%
7	Cả hai	Tester.		100%

VIII. Tài liệu tham khảo:

- Ani Hsieh, “A* Search Algorithm”, course E28, Mobile Robotics.
- Maxim Likhachev, Geoff Gordon and Sebastian Thrun, "ARA*: Anytime A* with Provable Bounds on Sub-Optimality," Advances in Neural Information Processing Systems 16 (NIPS), MIT Press, Cambridge, MA, 2004.
- https://stackoverflow.com/
- https://www.python.org/