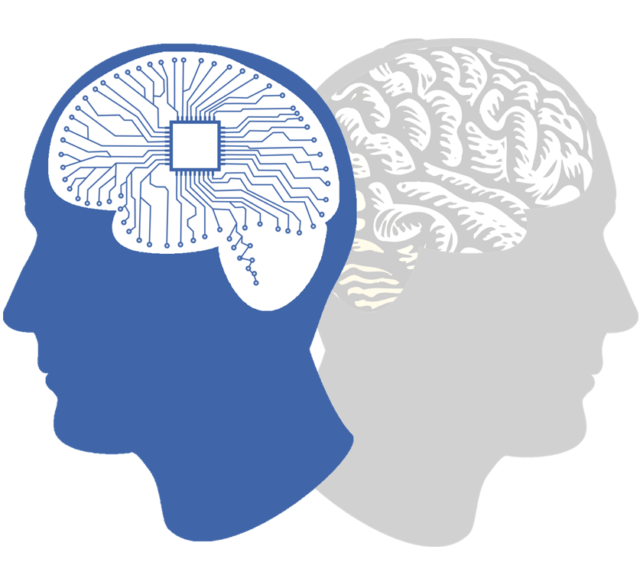
Khánh Hòa - 2022



BÁO CÁO THỰC HÀNH TRÍ TUỆ NHÂN TẠO

Sinh viên thực hiện: Võ Tấn Khuê

Mã số sinh viên: 61133814

# MỤC LỤC

[MỤC LỤC 2](#_Toc110407512)

[Chương 1. CÁC PHƯƠNG PHÁP TÌM KIẾM HEURISTIC 4](#_Toc110407513)

[1.1 TÌM KIẾM THEO CHIỀU SÂU (DEPTH FIRST SEARCH) 4](#_Toc110407514)

[1.1.1 Tổng quan về DFS 4](#_Toc110407515)

[1.1.2 Code chương trình 4](#_Toc110407516)

[1.1.3 Kết quả chạy chương trình 5](#_Toc110407517)

[1.2 TÌM KIẾM THEO CHIỀU RỘNG (BREADTH FIRST SEARCH) 6](#_Toc110407518)

[1.2.1 Tổng quan về BFS 6](#_Toc110407519)

[1.2.2 Code chương trình 7](#_Toc110407520)

[1.2.3 Kết quả chạy chương trình 8](#_Toc110407521)

[1.3 TÌM KIẾM ƯU TIÊN TỐI ƯU (BEST-FIRST SEARCH) 9](#_Toc110407522)

[1.3.1 Thuật giải AKT 9](#_Toc110407523)

[1.3.1.1 Tổng quan 9](#_Toc110407524)

[1.3.1.2 Code chương trình 9](#_Toc110407525)

[1.3.1.3 Kết quả cài đặt 11](#_Toc110407526)

[1.3.2 Thuật giải A\* 11](#_Toc110407527)

[1.3.2.1 Tổng quan 11](#_Toc110407528)

[1.3.2.2 Code chương trình và cài đặt 13](#_Toc110407529)

[Chương 2. CÁC BÀI TOÁN TRÒ CHƠI 15](#_Toc110407530)

[2.1 ĐỔI TIỀN (VÉT CẠN VÀ HEURISTIC) 15](#_Toc110407531)

[2.1.1 Code chương trình 15](#_Toc110407532)

[2.1.2 Kết quả chạy chương trình 16](#_Toc110407533)

[2.2 TIC TAC TOE THUẬT TOÁN MINIMAX 16](#_Toc110407534)

[2.2.1 Code chương trình 16](#_Toc110407535)

[2.2.2 Kết quả chạy chương trình 20](#_Toc110407536)

[2.3 BÀI TOÁN TSP 20](#_Toc110407537)

[2.3.1 Code chương trình 20](#_Toc110407538)

[2.3.2 Kết quả chạy chương trình 22](#_Toc110407539)

[Chương 3. BIỂU DIỄN TRI THỨC 22](#_Toc110407540)

[3.1 CHỨNG MINH LOGIC MỆNH ĐỀ BẰNG PHƯƠNG PHÁP SỐ 22](#_Toc110407541)

[3.1.1 Code chương trình 22](#_Toc110407542)

[3.1.2 Kết quả chạy 26](#_Toc110407543)

[3.2 DUY DIỄN TIẾN 26](#_Toc110407544)

[3.2.1 Tổng quan 26](#_Toc110407545)

[3.2.2 Code chương trình 26](#_Toc110407546)

[3.2.3 Kết quả chạy 26](#_Toc110407547)

[3.3 SUY DIỄN LÙI 26](#_Toc110407548)

[3.3.1 Tổng quan 26](#_Toc110407549)

[3.3.2 Code chương trình 27](#_Toc110407550)

[3.3.3 Kết quả chạy 29](#_Toc110407551)

[3.4 THUẬT GIẢI VƯƠNG HẠO 29](#_Toc110407552)

[3.4.1 Tổng quan 29](#_Toc110407553)

[3.4.2 Code chương trình 29](#_Toc110407554)

[3.4.3 Kết quả chạy 29](#_Toc110407555)

[3.5 THUẬT GIẢI ROBINSON 30](#_Toc110407556)

[3.5.1 Tổng quan 30](#_Toc110407557)

[3.5.2 Code chương trình 30](#_Toc110407558)

[3.5.3 Kết quả chạy 30](#_Toc110407559)

[Chương 4. THUẬT GIẢI QUINLAND TRONG WEKA 30](#_Toc110407560)

[4.1 Yêu cầu bài toán 30](#_Toc110407561)

[Chương 5. IOT 33](#_Toc110407562)

[5.1 XÂY DỰNG HỆ THỐNG ĐÈN ĐƯỜNG THÔNG MINH 33](#_Toc110407563)

[5.1.1 Tổng quan 33](#_Toc110407564)

[5.1.2 Sơ đồ thiết kế 33](#_Toc110407565)

[5.1.3 Đặc điểm của linh kiện 34](#_Toc110407566)

[5.1.4 Code chương trình 34](#_Toc110407567)

[5.1.5 Kết quả chạy chương trình 34](#_Toc110407568)

[5.2 BẬT TẮT ĐÈN LED VỚI CẢM BIẾN HC-SR04 34](#_Toc110407569)

[5.2.1 Mô tả 34](#_Toc110407570)

[5.2.2 Sơ đồ thiết kế 35](#_Toc110407571)

[5.2.3 Đặc điểm của linh kiện 35](#_Toc110407572)

[5.2.4 Code chương trình 35](#_Toc110407573)

[5.3 SMART HOME MINI 36](#_Toc110407574)

[5.3.1 Mô tả 36](#_Toc110407575)

[5.3.2 Sơ đồ thiết kế 36](#_Toc110407576)

[5.3.3 Cấu hình mạng và luật 38](#_Toc110407577)

[TÀI LIỆU THAM KHẢO 42](#_Toc110407578)

# CÁC PHƯƠNG PHÁP TÌM KIẾM HEURISTIC

## TÌM KIẾM THEO CHIỀU SÂU (DEPTH FIRST SEARCH)

### Tổng quan về DFS

Tìm kiếm ưu tiên chiều sâu hay tìm kiếm theo chiều sâu (tiếng Anh: Depth-first search - DFS) là một thuật toán duyệt hoặc tìm kiếm trên một cây hoặc một đồ thị. Thuật toán khởi đầu tại gốc (hoặc chọn một đỉnh nào đó coi như gốc) và phát triển xa nhất có thể theo mỗi nhánh.

Ý tưởng thuật toán:

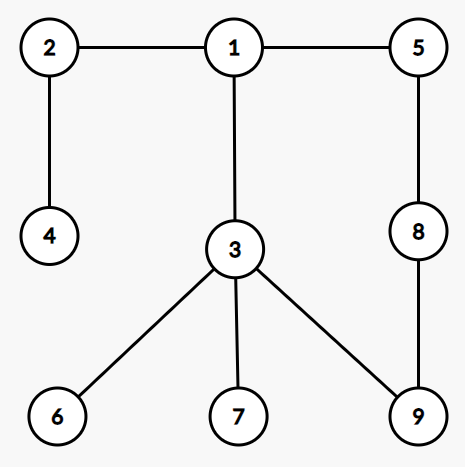
1. DFS trên đồ thị vô hướng cũng giống như khám phá mê cung với một cuộn chỉ và một thùng sơn đỏ để đánh dấu, tránh bị lạc. Trong đó mỗi đỉnh s trong đồ thị tượng trưng cho một cửa trong mê cung.
2. Ta bắt đầu từ đỉnh s, buộc đầu cuộn chỉ vào s và đánh đấu đỉnh này "đã thăm". Sau đó ta đánh dấu s là đỉnh hiện hành u.
3. Bây giờ, nếu ta đi theo cạnh (u,v) bất kỳ.
4. Nếu cạnh (u,v) dẫn chúng ta đến đỉnh "đã thăm" v, ta quay trở về u.
5. Nếu đỉnh v là đỉnh mới, ta di chuyển đến v và lăn cuộn chỉ theo. Đánh dấu v là "đã thăm". Đặt v thành đỉnh hiện hành và lặp lại các bước.
6. Cuối cùng, ta có thể đi đến một đỉnh mà tại đó tất cả các cạnh kề với nó đều dẫn chúng ta đến các đỉnh "đã thăm". Khi đó, ta sẽ quay lui bằng cách cuộn ngược cuộn chỉ và quay lại cho đến khi trở lại một đỉnh kề với một cạnh còn chưa được khám phá. Lại tiếp tục quy trình khám phá như trên.
7. Khi chúng ta trở về s và không còn cạnh nào kề với nó chưa bị khám phá là lúc DFS dừng. [1]

### Code chương trình

|  |
| --- |
| *#include <bits/stdc++.h>*  *using namespace std;*  *int n, m;*  *//mang vector de luu danh sach ke*  *vector<int> adj[1001];*  *// kiem tra mot dinh da duoc tham hay chua*  *bool visited[1001];*  *void inp(){*  *cin >> n >> m;*  *for (int i = 0; i< m; i++)*  *{*  *int x, y;*  *cin >> x >> y;*  *adj[x].push\_back(y);*  *// co huong thi bo dong nay*  *adj[y].push\_back(x);*  *}*  *memset(visited, false, sizeof(visited));*  *}*  *void dfs(int u){*  *cout << u << " ";;*  *//danh dau la u da duoc tham*  *visited[u] = true;*  *//dung foreach de duyet tung diem ke cua U*  *for(int v : adj[u]){*  *//veu diem v chua dc tham thi*  *if(!visited[v]){*  *dfs(v);*  *}*  *}*  *}*  *int main(){*  *inp();*  *dfs(1);*  *}* |

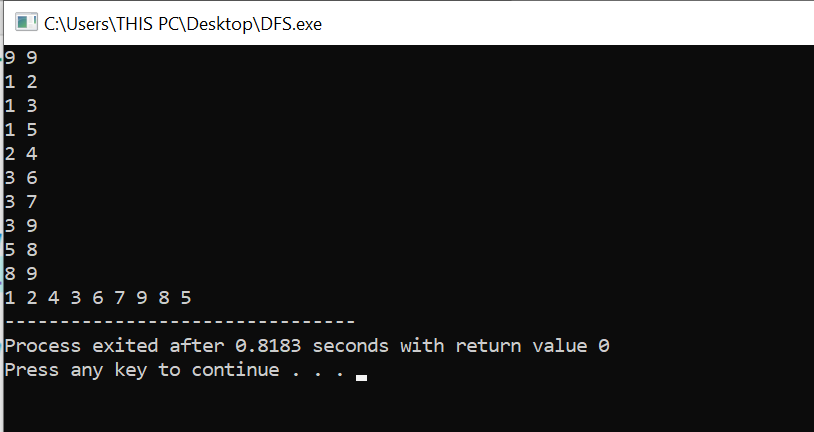
### Kết quả chạy chương trình

Ta có đồ thị như trong hình sau, ta sẽ duyệt theo chiều sâu bắt đầu từ đỉnh số 1.



Hình 1.1.3. Đồ thị gồm 9 đỉnh và 9 cạnh

Kết quả duyệt ta theo thứ tự là **1 2 4 3 6 7 9 8 5**



Hình 1.1.3. Kết quả cài đặt

## TÌM KIẾM THEO CHIỀU RỘNG (BREADTH FIRST SEARCH)

### Tổng quan về BFS

Trong lý thuyết đồ thị, 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: (a) cho trước một đỉnh của đồ thị; (b) thêm các đỉnh kề với đỉnh vừa cho vào danh sách có thể hướng tới tiếp theo. Có thể sử dụng thuật toán mục đích: tìm kiếm đường đi từ một đỉnh gốc cho trước tới một đỉnh đích, và tìm kiếm đường đi từ đỉnh gốc tới tất cả các đỉnh khác.

Ý tưởng thuật toán:

Thuật toán sử dụng một cấu trúc dữ liệu hàng đợi để lưu trữ thông tin trung gian

1. Chèn đỉnh gốc vào hàng đợi (đang hướng tới)
2. Lấy ra đỉnh đầu tiên trong hàng đợi và quan sát nó

* Nếu đỉnh này chính là đỉnh đích, dừng quá trình tìm kiếm và trả về kết quả.
* Nếu không phải thì chèn tất cả các đỉnh kề với đỉnh vừa thăm nhưng chưa được quan sát trước đó vào hàng đợi.

1. Nếu hàng đợi là rỗng, thì tất cả các đỉnh có thể đến được đều đã được quan sát – dừng việc tìm kiếm và trả về "không thấy".
2. Nếu hàng đợi không rỗng thì quay về bước 2.

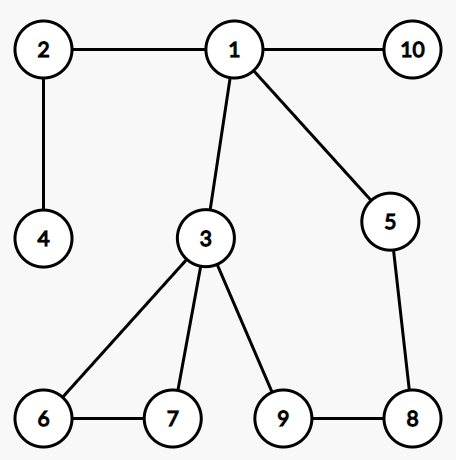
Ghi chú: Nếu sử dụng một ngăn xếp thay vì hàng đợi thì thuật toán trở thành thuật toán tìm kiếm theo chiều sâu. [2]

### Code chương trình

|  |
| --- |
| *#include <bits/stdc++.h>*  *using namespace std;*  *int n,m;*  *// mang vecto de luu danh sach ke*  *vector<int> adj[1001];*  *// nhap input va chuyen thanh ds ke*  *bool visited[1001];*  *void inp(){*  *cin >> n >> m;*  *for(int i=0; i<m; i++){*  *int x, y;*  *cin >> x >> y;*  *adj[x].push\_back(y);*  *adj[y].push\_back(x);*  *}*  *memset(visited, false, sizeof(visited));*  *}*  *void bfs(int u){*  *// Buoc khoi tao*  *queue<int> q;*  *q.push(u);*  *visited[u] = true;*    *// Buoc Lap*  *while(!q.empty()){*  *// lay dinh o dau hang doi*  *int v = q.front();*  *// lay ra va xoa no di*  *q.pop();*  *cout << v << " ";*  *for(int x : adj[v]){*  *//visited[x] == false*  *if(!visited[x]){*  *q.push(x);*  *visited[x] = true;*  *}*  *}*  *}}*  *int main(){*  *inp();*  *bfs(1);*  *return 0;*  *}* |

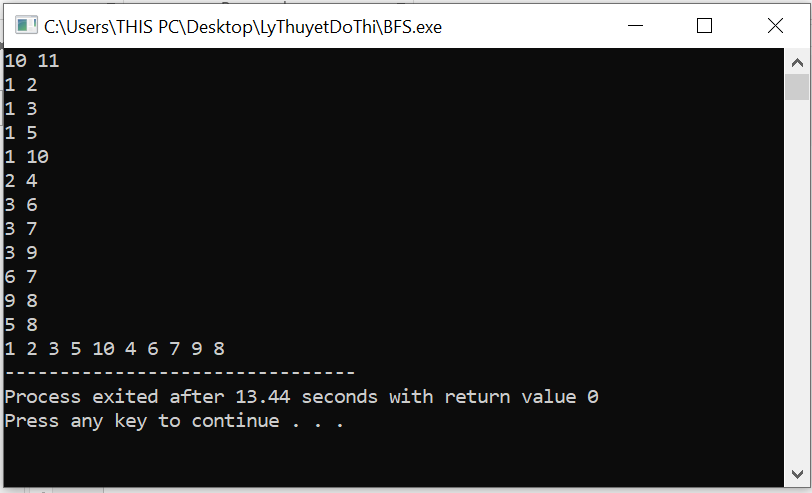
### Kết quả chạy chương trình

Ta có đồ thị như trong hình sau, ta sẽ duyệt theo chiều sâu bắt đầu từ đỉnh số 1.



Hình 1.1.3. Đồ thị gồm 10 đỉnh và 11 cạnh

Kết quả duyệt ta theo thứ tự là **1 2 3 5 10 4 6 7 9 8**



Hình 1.1.3. Kết quả cài đặt

## TÌM KIẾM ƯU TIÊN TỐI ƯU (BEST-FIRST SEARCH)

### Thuật giải AKT

#### Tổng quan

Thuật giải AKT mở rộng AT bằng cách sử dụng thêm thông tin ước lượng h’. Độ tốt của một trạng thái f là tổng của hai hàm g và h’.

***Phương pháp:***

1. Đặt OPEN chứa trạng thái khởi đầu.
2. Cho đến khi tìm được trạng thái đích hoặc không còn nút nào trong OPEN, thực hiện:
3. Chọn trạng thái (Tmax) có giá trị f nhỏ nhất trong OPEN (và xóa Tmax khỏi OPEN)
4. Nếu Tmax là trạng thái kết thúc thì thoát.
5. Ngược lại, tạo ra các trạng thái kế tiếp Tk có thể có từ trạng thái Tmax. Đối với mỗi trạng thái kế tiếp Tk thực hiện:

g(Tk) = g(Tmax) + cost(Tmax, Tk);

*Tính h’(Tk)*

*f(Tk) = g(Tk) + h’(Tk);*

Thêm Tk vào OPEN.

#### Code chương trình

Ảnh có chứa văn bản

Mô tả được tạo tự động

Ảnh có chứa văn bản

Mô tả được tạo tự động

***Ảnh có chứa văn bản

Mô tả được tạo tự động***

Ảnh có chứa văn bản

Mô tả được tạo tự động

Ảnh có chứa văn bản

Mô tả được tạo tự động

Ảnh có chứa văn bản

Mô tả được tạo tự động

Ảnh có chứa văn bản

Mô tả được tạo tự động

#### Kết quả cài đặt

Áp dụng giải thuật AKT vào bài toán sắp xếp các số trên ma trận 3x3 theo thứ tự nhất định.

### Thuật giải A\*

#### Tổng quan

<https://www.mygreatlearning.com/blog/a-search-algorithm-in-artificial-intelligence/>

***Ý tưởng:*** Sử dụng hàm heuristics chấp nhận được, áp dụng tìm kiếm theo chiều rộng và loại bỏ những đường đi có chi phí cao.

Hàm lượng giá: *f(u) = g(u) + h(u)*

Trong đó:

* *g(u)* là chi phí để đến u
* *h(u)* là lượng giá từ u đến đích
* *f(u)* là ước lượng tổng giá đến đích qua u

***Phương pháp:***

* A\* là một phiên bản đặc biệt của AKT áp dụng cho trường hợp đồ thị.
* Thuật giải A\* có sử dụng thêm tập hợp CLOSE để lưu trữ những trường hợp đã được xét đến.
* A\* mở rộng AKT bằng cách bổ sung cách giải quyết trường hợp khi "mở" một nút mà nút này đã có sẵn trong OPEN hoặc CLOSE.
* Khi xét đến một trạng thái Ti bên cạnh việc lưu trữ 3 giá trị cơ bản g, h’, f’ để phản ánh độ tốt của trạng thái đó, A\* còn lưu trữ thêm hai thông số sau:

1. Trạng thái cha của trạng thái Ti (ký hiệu là Cha(Ti))

2. Danh sách các trạng thái kế tiếp của Ti

***Mã giả:***

*Procedure A \*;*

*Begin*

1. *Khởi tạo danh sách L chỉ chứa trạng thái ban đầu*
2. *Loop do*
3. *if L rỗng then {thông báo thất bại; stop;}*
4. *Loại trạng thái u ở đầu danh sách L;*
5. *if u là trạng thái đích then {thông báo thành công; stop;}*
6. *for mỗi trạng thái v kề u do*

*{*

*g(v) g(u) + k(u, v);*

*f(v) g(v) + h(v);*

*Đặt v vào danh sách L;*

*}*

1. *Sắp xếp L theo thứ tự giảm dần của hàm f sao cho trạng thái có giá trị của hàm f nhỏ nhất ở đầu danh sách;*

*End;*

#### Code chương trình và cài đặt

Ảnh có chứa văn bản

Mô tả được tạo tự động

Ảnh có chứa văn bản

Mô tả được tạo tự động

Ảnh có chứa văn bản

Mô tả được tạo tự động

#kết quả trả về: Path found: ['A', 'E', 'D', 'G']

***Nhận xét:***

Nếu không gian các trạng thái là hữu hạn và có giải pháp để tránh việc xét lặp lại các trạng thái thì giải thuật A \* là hoàn chỉnh (tìm được lời giải), nhưng không đảm bảo tối ưu.

Nếu không gian các trạng thái là vô hạn thì giải thuật A \* có thể không cho ra lời giải.

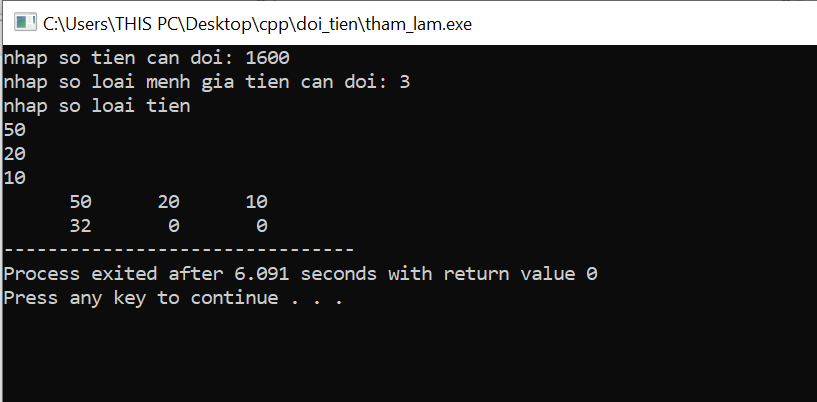
# CÁC BÀI TOÁN TRÒ CHƠI

## ĐỔI TIỀN (VÉT CẠN VÀ HEURISTIC)

### Code chương trình

|  |
| --- |
| #include <iostream>  #include <iomanip>  #include <algorithm>  using namespace std;  int main(int argc, char\*\* argv)  {  int n;  cout << "nhap so tien can doi: ";  cin >> n;  int i = 0;  int x = 8;  int smg;  cout << "nhap so loai menh gia tien can doi: ";  cin >> smg;  int sl;  int arr[100];  cout << "nhap so loai tien" << endl;  for (int i = 0; i < smg; i++)  cin >> arr[i];  sort(arr, arr + smg, greater<int>());  for (int i = 0; i < smg; i++)  cout << setw(x) << arr[i] << setw(x);  cout << endl;  while (n >= 0 && i != smg)  {  sl = n / arr[i];  n = n % arr[i];  i++;  cout << sl << setw(x);  }  } |

### Kết quả chạy chương trình



Hình 1.1.3. Kết quả

## TIC TAC TOE THUẬT TOÁN MINIMAX

### Code chương trình

|  |
| --- |
| #include <iostream>  struct move  {  int r, c, score;  };  struct TicTacToe  {  char player, computer, grid[3][3];  TicTacToe()  {  for (int i = 0; i < 3; i++)  {  for (int j = 0; j < 3; j++)  {  grid[i][j] = ' ';  }  }  }  bool win()  {  int win\_states[8][3] = {{0, 1, 2}, {3, 4, 5}, {6, 7, 8}, {0, 3, 6}, {1, 4, 7}, {2, 5, 8}, {0, 4, 8}, {2, 4, 6}};  for (int i = 0; i < 8; i++)  {  bool win = true;  int first\_r = win\_states[i][0] / 3, first\_c = win\_states[i][0] % 3;  for (int j = 0; j < 3; j++)  {  int r = win\_states[i][j] / 3, c = win\_states[i][j] % 3;  if (grid[first\_r][first\_c] == ' ' || grid[first\_r][first\_c] != grid[r][c])  {  win = false;  }  }  if (win)  return true;  }  return false;  }  bool tie()  {  if (win())  return false;  for (int i = 0; i < 3; i++)  {  for (int j = 0; j < 3; j++)  {  if (grid[i][j] == ' ')  {  return false;  }  }  }  return true;  }  void player\_move()  {  while (true)  {  std::cout << "Enter an empty cell (1 - 9): ";  int cell;  std::cin >> cell;  int r = (cell - 1) / 3, c = (cell - 1) % 3;  if (cell >= 1 && cell <= 9 && grid[r][c] == ' ')  {  grid[r][c] = player;  break;  }  }  }  void computer\_move()  {  move best\_move = minimax();  grid[best\_move.r][best\_move.c] = computer;  }  move minimax(bool maximizing\_player = true)  {  move best\_move;  if (win())  {  if (maximizing\_player)  {  best\_move.score = -1;  }  else  {  best\_move.score = 1;  }  return best\_move;  }  else if (tie())  {  best\_move.score = 0;  return best\_move;  }  best\_move.score = maximizing\_player ? -2 : 2;  for (int i = 0; i < 3; i++)  {  for (int j = 0; j < 3; j++)  {  if (grid[i][j] == ' ')  {  grid[i][j] = maximizing\_player ? computer : player;  move board\_state = minimax(!maximizing\_player);  if (maximizing\_player)  {  if (board\_state.score > best\_move.score)  {  best\_move.score = board\_state.score;  best\_move.r = i;  best\_move.c = j;  }  }  else  {  if (board\_state.score < best\_move.score)  {  best\_move.score = board\_state.score;  best\_move.r = i;  best\_move.c = j;  }  }  grid[i][j] = ' ';  }  }  }  return best\_move;  }  void play()  {  while (true)  {  std::cout << "Which symbol (X or O, X goes first)? ";  std::cin >> player;  if (player == 'X' || player == 'O')  {  break;  }  }  computer = player == 'X' ? 'O' : 'X';  if (player == 'O')  {  computer\_move();  }  print();  while (true)  {  player\_move();  print();  if (win())  {  std::cout << "Player wins!\n";  return;  }  else if (tie())  {  std::cout << "Tie!\n";  return;  }  std::cout << "Computer is making a move...\n";  computer\_move();  print();  if (win())  {  std::cout << "Computer wins!\n";  return;  }  else if (tie())  {  std::cout << "Tie!\n";  return;  }  }  }  void print()  {  std::cout << '\n';  for (int i = 0; i < 3; i++)  {  if (i)  {  std::cout << "-----------\n";  }  for (int j = 0; j < 3; j++)  {  if (j)  {  std::cout << "|";  }  std::cout << ' ';  if (grid[i][j] == ' ')  {  std::cout << 3 \* i + j + 1;  }  else  {  std::cout << grid[i][j];  }  std::cout << ' ';  }  std::cout << '\n';  }  std::cout << '\n';  }  };  int main()  {  TicTacToe game;  game.play();  } |

### Kết quả chạy chương trình

## BÀI TOÁN TSP

### Code chương trình

|  |
| --- |
| // Enter the number of villages: 4  // Enter the Cost Matrix  // Enter Elements of Row: 1  // 0 4 1 3  // Enter Elements of Row: 2  // 4 0 2 1  // Enter Elements of Row: 3  // 1 2 0 5  // Enter Elements of Row: 4  // 3 1 5 0  // The cost list is:  // 0 4 1 3  // 4 0 2 1  // 1 2 0 5  // 3 1 5 0  // The Path is:  // 1â€”>3â€”>2â€”>4â€”>1  // Minimum cost is 7  #include <iostream>  using namespace std;  int ary[10][10], completed[10], n, cost = 0;  void takeInput()  {  int i, j;  cout << "Nhap so hang so cot: ";  cin >> n;  cout << "\nNhan chi phi cua ma tran\n";  for (i = 0; i < n; i++)  {  cout << "\nNhap cac phan tu cua hang: " << i + 1 << "\n";  for (j = 0; j < n; j++)  cin >> ary[i][j];  completed[i] = 0;  }  cout << "\n\nDanh sach chi phi:";  for (i = 0; i < n; i++)  {  cout << "\n";  for (j = 0; j < n; j++)  cout << "\t" << ary[i][j];  }  }  int least(int c)  {  int i, nc = 999;  int min = 999, kmin;  for (i = 0; i < n; i++)  {  if ((ary[c][i] != 0) && (completed[i] == 0))  if (ary[c][i] + ary[i][c] < min)  {  min = ary[i][0] + ary[c][i];  kmin = ary[c][i];  nc = i;  }  }  if (min != 999)  cost += kmin;  return nc;  }  void mincost(int city)  {  int i, ncity;  completed[city] = 1;  cout << city + 1 << "--->";  ncity = least(city);  if (ncity == 999)  {  ncity = 0;  cout << ncity + 1;  cost += ary[city][ncity];  return;  }  mincost(ncity);  }  int main()  {  takeInput();  cout << "\n\nThe Path is:\n";  mincost(0); // passing 0 because starting vertex  cout << "\n\nGia nho nhat " << cost;  return 0;  } |

### Kết quả chạy chương trình

# BIỂU DIỄN TRI THỨC

## CHỨNG MINH LOGIC MỆNH ĐỀ BẰNG PHƯƠNG PHÁP SỐ

### Code chương trình

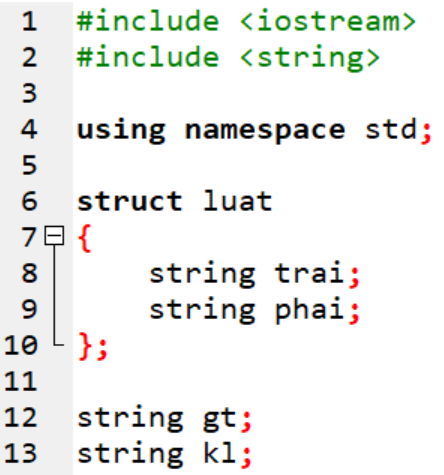
|  |
| --- |
| *#include <iostream>*  *#include <string.h>*  *#include <math.h>*  *#include <iomanip>*  *#include <vector>*  *#include <fstream>*  *using namespace std;*  *char\* ChanTri(vector<char> ch, vector<char> op, vector<char\*> in);*  *char\* UuTien(vector<char> ch, vector<char> op, vector<char\*> in);*  *char\* UuTien(vector<char> ch, vector<char> op, vector<char\*> in);*  *void add(vector<char> ch, vector<char> op, vector<char\*> in,int n);*  *int main()*  *{*  *vector<char> ch;//luu ky tu bien*  *vector<char> op;//luu toan tu |:or, &:and, >:keo theo, ~:not*  *vector<char\*> in;//luu true/false tung bien nhap vao*  *vector<char\*> in1;//luu true/false tung bien trong bang chan tri*  *fstream ifs("input.txt",ios::in);//doc file*    *ofstream ghi("output.txt");*  *string input;*  *// cin>>input;*  *getline(ifs,input);*  *for(int i=0;i<input.size()+1;i++)//dung vong lap theo lan luot ky tu vao vector ky tu, toan tu vao vector toan tu tuong ung*  *{*  *if(input[i]=='\n')//doc den phim enter thi dung*  *break;*  *if((int)input[i]==0)//doc den phim cach thi bo qua*  *continue;*  *if((int)input[i]>96 && (int)input[i]<124)*  *ch.push\_back(input[i]);*  *else*  *op.push\_back(input[i]);*  *}*  *cout<<"- Cong thuc Logic: "<<input;*  *cout<<"\n- Nhap chan tri cac menh de tham gia cong thuc:\n";*    *for(int i=1;i<ch.size();i++)*  *{*  *int u = 0;*  *if((int)op[u]==61) // loc dau bang*  *{*  *for(int j=0;j<input.size()+1;j++)//dung vong lap theo lan luot ky tu vao vector ky tu, toan tu vao vector toan tu tuong ung*  *{*  *if((int)input[i]>96 && (int)input[i]<124)*  *ch.push\_back(input[i]);*  *else*  *op.push\_back(input[i]);*  *}*  *}*  *}*    *for(int i=0;i<ch.size();i++)//nhap chan tri cho tung bien*  *{*  *char \*tmp = new char[6];*  *do*  *{*  *cout<<ch[i]<<" = ";*  *cin>>tmp;*  *}while(strcmpi(tmp,"true")!=0 && strcmpi(tmp,"false")!=0);*  *in.push\_back(tmp);*  *}*    *cout<<"\n- Ket qua menh de chung minh la: "<<UuTien(ch,op,in)<<endl;//goi ham tinh chan tri*  *ghi<<"- Cong thuc Logic: "<<input; //ghi vao file*  *ghi<<"\n- Ket qua menh de chung minh la: "<<UuTien(ch,op,in)<<endl;//goi ham tinh chan tri*  *ghi.close();*  *cout<<"\n\n\tBANG CHAN TRI CUA CONG THUC\n\n";*  *int x=8\*(ch.size()+1);*  *for(int i=0;i<=x;i++)//in header cua bang*  *{*  *if(i==0)*  *cout<<char(218);*  *else if(i==x)*  *cout<<char(191);*  *else if(i%8==0)*  *cout<<char(194);*  *else*  *cout<<char(196);*  *}*  *cout<<endl;*  *for(int i=0;i<ch.size();i++)//in title*  *cout<<left<<char(179)<<" "<<setw(4)<<ch[i];*  *cout<<char(179)<<setw(7)<<" F"<<char(179)<<endl;*  *for(int i=0;i<=x;i++)//in khung bang*  *{*  *if(i==0)*  *cout<<char(195);*  *else if(i==x)*  *cout<<char(180);*  *else if(i%8==0)*  *cout<<char(197);*  *else*  *cout<<char(196);*  *}*  *cout<<endl;*  *for(int i=0;i<ch.size();i++)//gan tat ca chan tri bang false*  *{*  *char \*tmp = new char[6];*  *strcpy(tmp,"false");*  *in1.push\_back(tmp);*  *}*  *add(ch,op,in1,0);//in dong dau tien bang chan tri*  *for(int i=1;i<pow(2,ch.size());i++)//dung vong lap in cac dong tiep theo bang tran tri theo nguyen tac cong bit*  *{*  *add(ch,op,in1,1);*  *}*  *for(int i=0;i<=x;i++)//in footer bang*  *{*  *if(i==0)*  *cout<<char(192);*  *else if(i==x)*  *cout<<char(217);*  *else if(i%8==0)*  *cout<<char(193);*  *else*  *cout<<char(196);*  *}*  *cout<<endl;*  *return 0;*  *}*  *char\* ChanTri(vector<char> ch, vector<char> op, vector<char\*> in)*  *{*  *bool result = (int)op[0]==126?(strcmpi(in[0],"true")==0?0:1):(strcmpi(in[0],"true")==0?1:0);//gan result bang chan tri bien dau tien*  *int j = (int)op[0]==126?1:0;//neu toan tu dau la not thi bo qua toan tu dau*  *for(int i=1;i<ch.size();i++)*  *{*  *if((int)op[j+1]!=126) //neu toan tu sau khac not thi tinh binh thuong*  *{*  *if((int)op[j]==124)*  *result|=strcmpi(in[i],"true")==0?1:0;*  *if((int)op[j]==38)*  *result&=strcmpi(in[i],"true")==0?1:0;*  *if((int)op[j]==62)*  *result=!result|strcmpi(in[i],"true")==0?1:0;*  *}*  *else//neu toan tu sau la not thi phu dinh chan tri cua bien do*  *{*  *if((int)op[j]==124)*  *result|=strcmpi(in[i],"true")==0?0:1;*  *if((int)op[j]==38)*  *result&=strcmpi(in[i],"true")==0?0:1;*  *if((int)op[j]==62)*  *result=!result|strcmpi(in[i],"true")==0?0:1;*  *j++;*  *}*  *j++;*  *}*  *char\* tmp = new char[6];*  *strcpy(tmp,result==1?"True":"False");*  *return tmp;//tra ve ket qua chan tri cua bieu thuc*  *}*  *char\* UuTien(vector<char> ch, vector<char> op, vector<char\*> in) {*  *for(int i=1;i<ch.size();i++)*  *{*  *int u = 0;*  *if((int)op[u]==40)*  *{*  *return ChanTri(ch,op,in);*  *if((int)op[u]==41)*  *{*  *break;*  *}*  *}*  *else*  *{*  *return ChanTri(ch,op,in);*  *}*    *}*  *}*  *void add(vector<char> ch, vector<char> op, vector<char\*> in,int n)*  *{*  *int cap = 0,tmp;*  *tmp = (strcmpi(in[in.size()-1],"true")==0?1:0)+n+cap;//gia tri tmp = chan tri cua bit cuoi + bit cong them + bien nho*  *strcpy(in[in.size()-1],tmp==1?"True":"False");//ghi gia tri cua bit cuoi sau khi cong*  *cap = tmp==2?1:0;//ghi gia tri cua bien nho*  *for(int i = in.size()-2;i>=0;i--)//dung vong lap tiep tuc cong bit cho den bit dau*  *{*  *tmp = (strcmpi(in[i],"true")==0?1:0)+cap;*  *strcpy(in[i],tmp==1?"True":"False");*  *cap = tmp==2?1:0;*  *}*  *for(int i=0;i<in.size();i++)//in ra tung dong cua bang chan tri*  *cout<<left<<char(179)<<" "<<setw(6)<<in[i];*  *cout<<char(179)<<" "<<setw(6)<<UuTien(ch,op,in)<<char(179)<<endl;*  *}* |

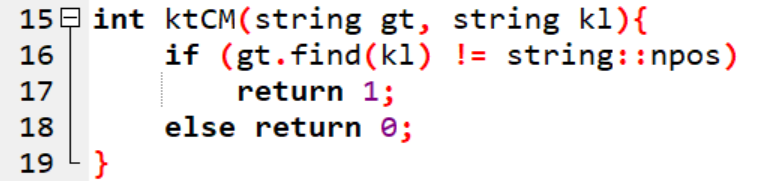
### Kết quả chạy

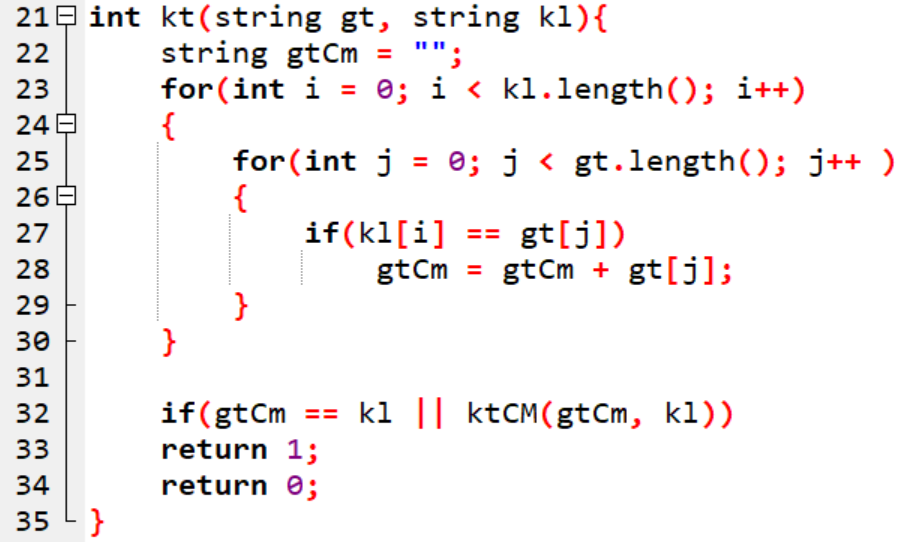
## SUY DIỄN TIẾN

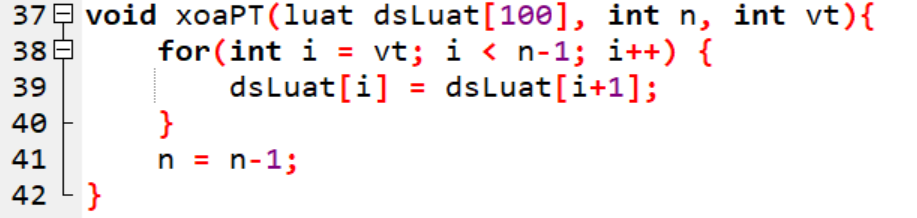
### Tổng quan

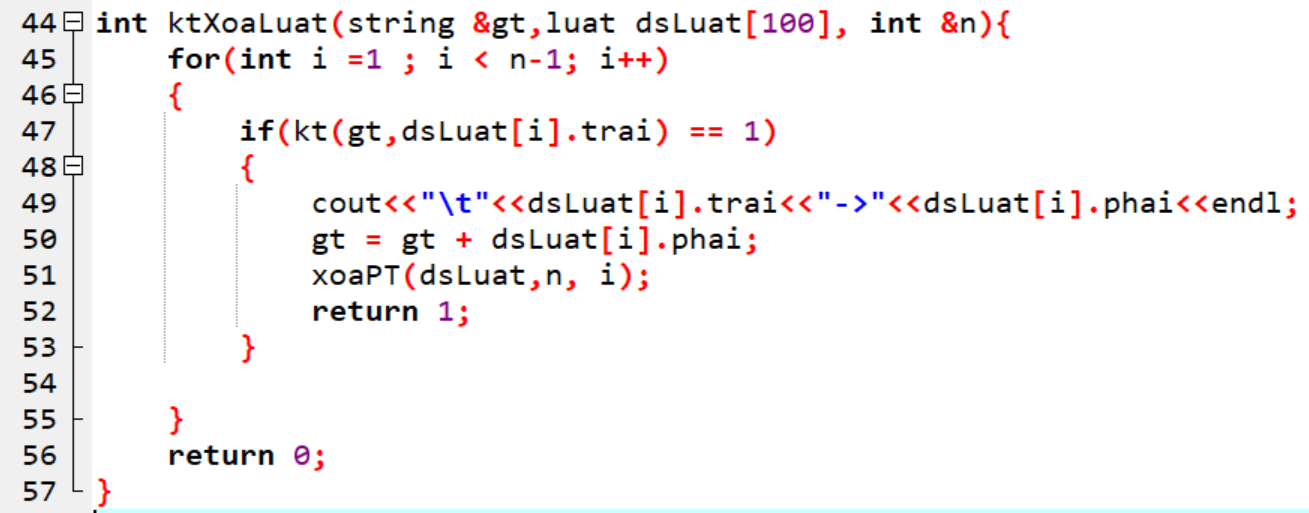
### Code chương trình

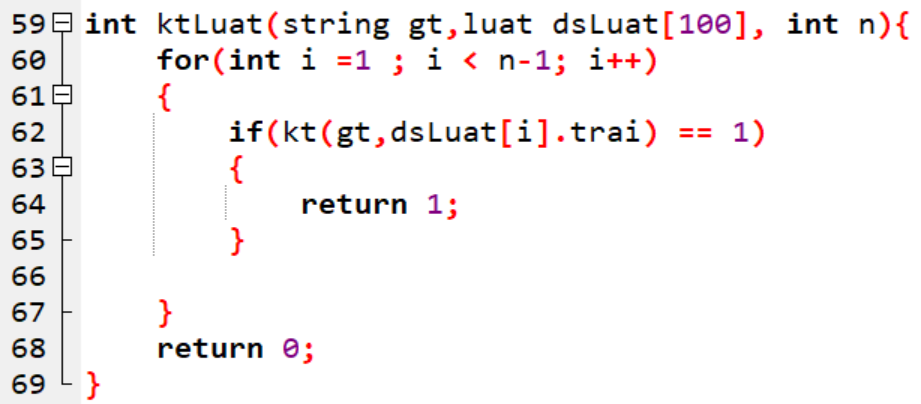


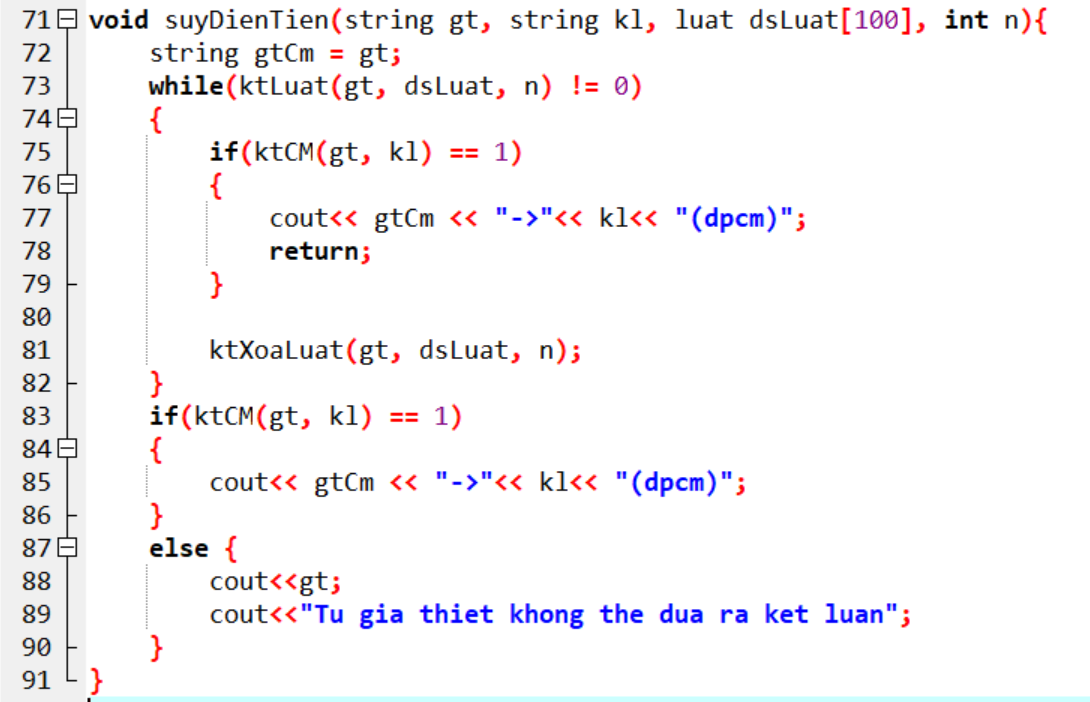


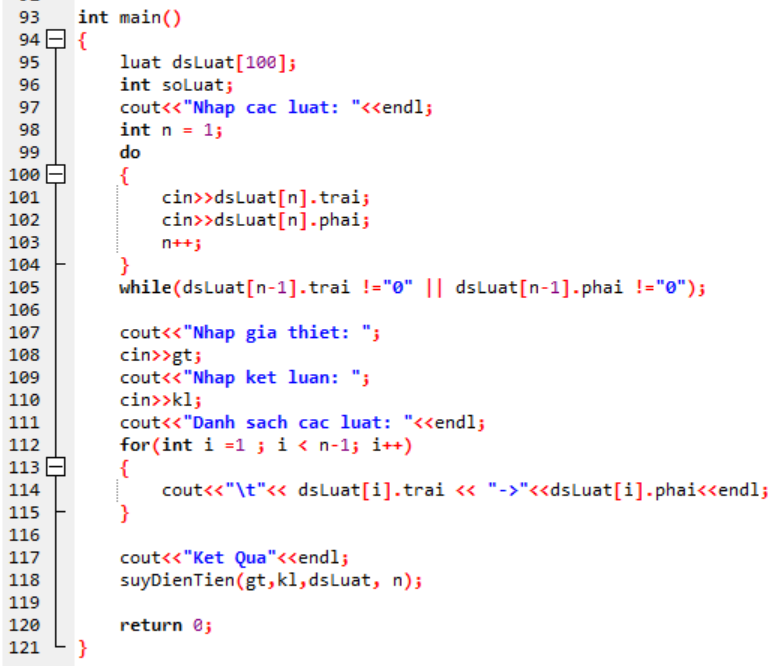




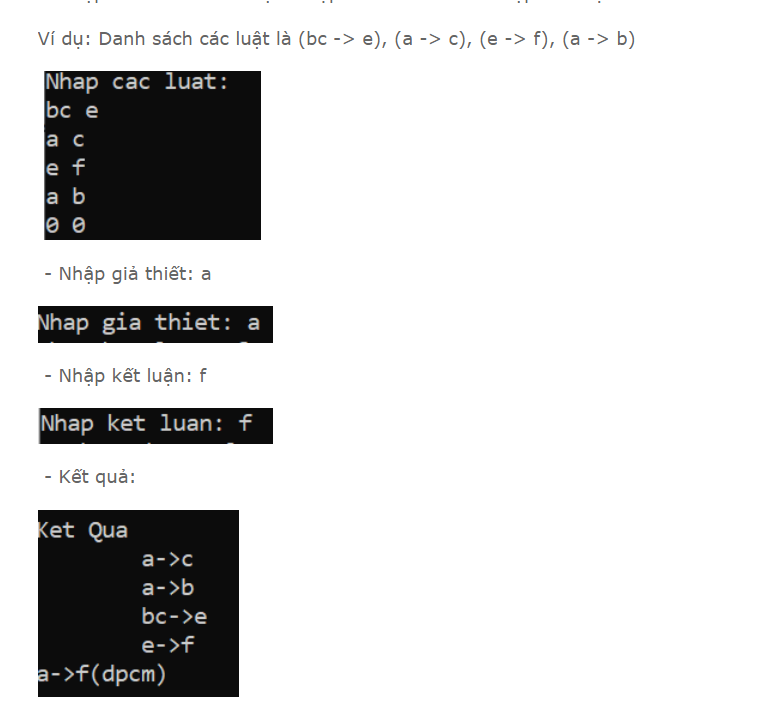








### Kết quả chạy



## SUY DIỄN LÙI

### Tổng quan

Suy diễn lùi (Backward chaining)

Suy diễn lùi bắt đầu với một danh sách các giả thiết và suy luận ngược lại từ hệ quả. Suy diễn lùi sẽ tìm kiếm các quy tắc suy luận cho đến khi tìm thấy một quy tắc có kết quả phù hợp với mục tiêu mong muốn. Nếu tiền đề trước đó của quy tắc này là đúng, thì quy tắc này sẽ được thêm vào danh sách mục tiêu.

**Mã giả:**

**Backward – Chaining(H)**

**{**

**if** H có trong tập luật R đã cho

**return true;**

**if** H không có trong bất kì tập luật nào

**then print**(“Liệu H có còn trong bất cứ luật nào bạn muốn được thêm vào?”);

or **return false;**

Duyệt qua từng luật/hệ quả/quy tắc A trong tập luật R phù hợp với H **do**

**if** Backward – Chaining(A) = **true**

**then return true;**

**return false;**

**}**

### Code chương trình

***Cài đặt:***

File input.txt

Ảnh có chứa văn bản

Mô tả được tạo tự động

Ảnh có chứa văn bản

Mô tả được tạo tự động

Ảnh có chứa văn bản

Mô tả được tạo tự động

Ảnh có chứa văn bản

Mô tả được tạo tự động

Ảnh có chứa văn bản

Mô tả được tạo tự động

Ảnh có chứa văn bản

Mô tả được tạo tự động

### Kết quả chạy

Ảnh có chứa văn bản

Mô tả được tạo tự động

## THUẬT GIẢI VƯƠNG HẠO

### Tổng quan

### Code chương trình

### Kết quả chạy

## THUẬT GIẢI ROBINSON

### Tổng quan

### Code chương trình

### Kết quả chạy

# SIMULATION IN GENERAL ALGORITHM

### Tổng quan

GAs là một kỹ thuật tối ưu hóa hoặc tìm kiếm giải pháp heuristic, ban đầu được thúc đẩy bởi nguyên tắc tiến hóa của Darwin thông qua chọn lọc (di truyền). GA sử dụng một phiên bản trừu tượng cao của các quy trình tiến hóa để đưa ra các giải pháp cho các vấn đề đã cho. Mỗi GA hoạt động trên một quần thể nhiễm sắc thể nhân tạo (artificial chromosomes). Đây là những chuỗi trong một bảng chữ cái hữu hạn (thường là nhị phân). Mỗi nhiễm sắc thể đại diện cho một giải pháp cho một vấn đề và có một thích hợp, một số thực là thước đo mức độ tốt của một giải pháp cho một vấn đề cụ thể.

GA được xây dựng từ một số thành phần riêng biệt. Đây là một điểm mạnh đặc biệt vì nó có nghĩa là các thành phần tiêu chuẩn có thể được sử dụng lại, với sự thích ứng nhỏ trong nhiều GA khác nhau, do đó, việc triển khai dễ dàng hơn. Các thành phần chính là mã hóa nhiễm sắc thể (chromosome encoding), hàm thích hợp (the fitness function), chọn lọc (selection), tái tổ hợp (recombination) và sơ đồ tiến hóa (the evolution scheme).

Một thiết kế điển hình cho GA cổ điển sử dụng thay thế hoàn toàn bằng các toán tử di truyền chuẩn có thể như sau:

(1) Tạo ngẫu nhiên một quần thể nguồn ban đầu của P nhiễm sắc thể.

(2) Tính thể trạng của từng nhiễm sắc thể c trong quần thể nguồn bằng hàm F(c).

(3) Tạo một quần thể kế thừa trống và sau đó lặp lại các bước sau cho đến khi tạo ra P nhiễm sắc thể.

(a) Sử dụng phương pháp chọn lọc theo tỉ lệ thuận, chọn hai nhiễm sắc thể là c1 và c2 từ quần thể nguồn.

(b) Áp dụng trao đổi chéo một điểm với c1 và c2 với tỷ lệ trao đổi chéo pc để thu được nhiễm sắc thể con c.

(c) Áp dụng đột biến đồng hợp với c có tỷ lệ đột biến pm để tạo ra c’.

(d) Thêm c’ vào quần thể kế thừa.

(4) Thay thế quần thể nguồn bằng quần thể kế thừa.

(5) Nếu không gặp điều kiện dừng, quay lại Bước 2.

https://machinelearningmastery.com/simple-genetic-algorithm-from-scratch-in-python/

# BÀI TOÁN SỬ DỤNG PROLOG

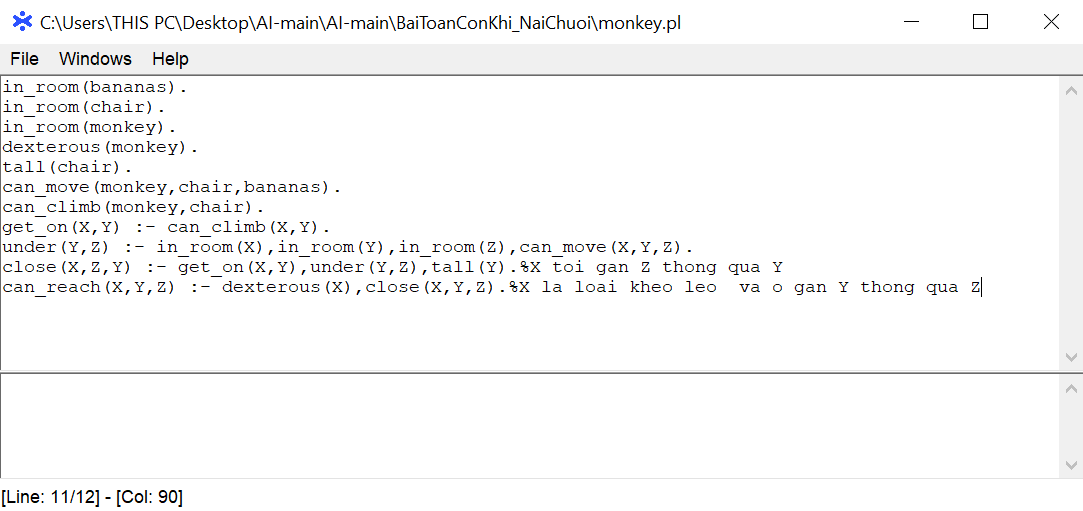
## CON KHỈ VÀ NẢI CHUỐI

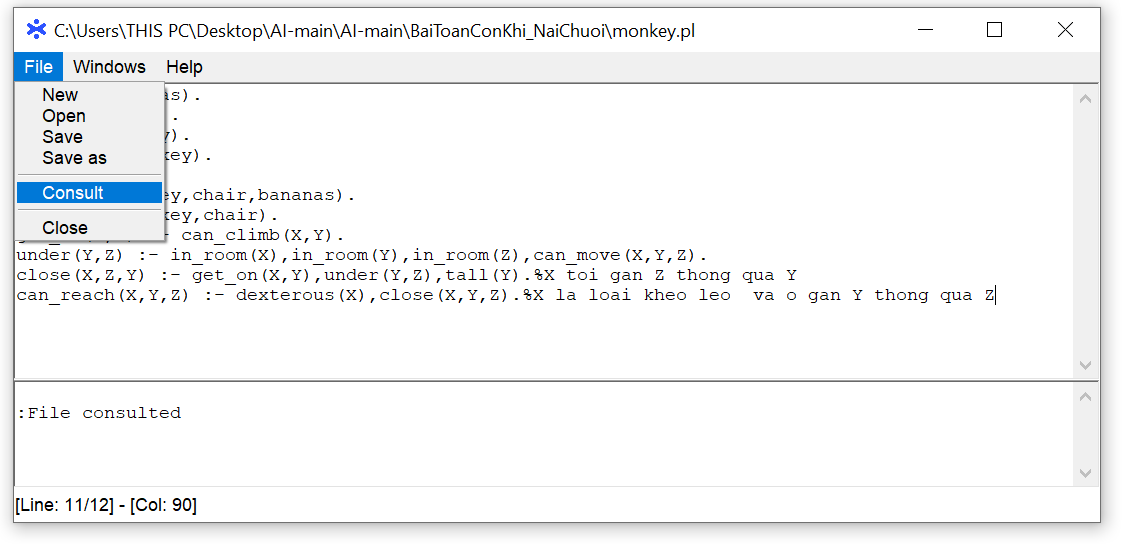
### Tổng quan

### Cài đặt



Vào file -> Open



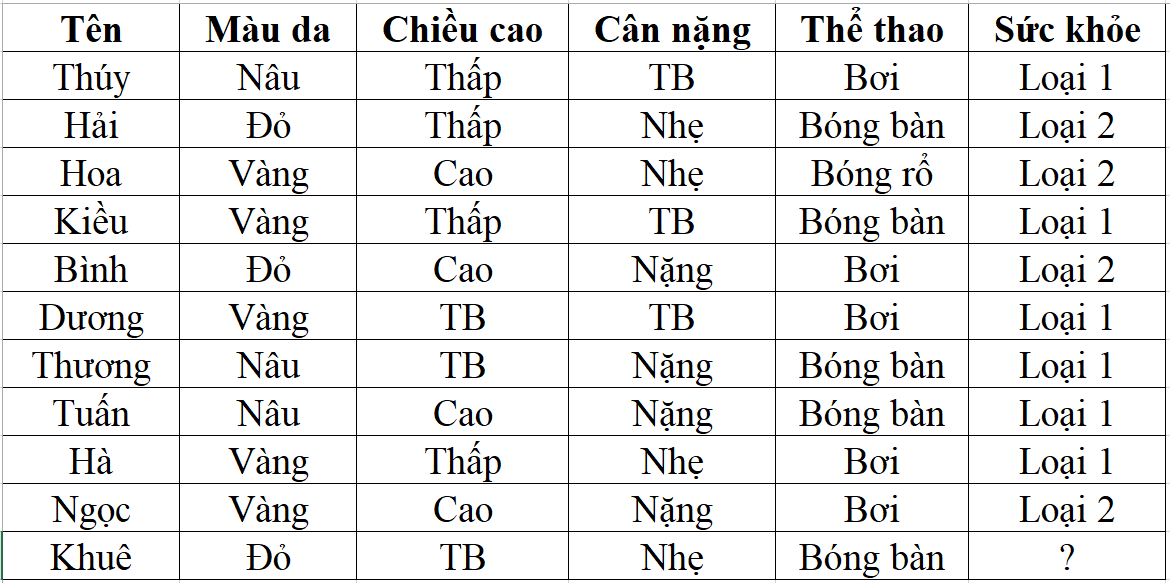


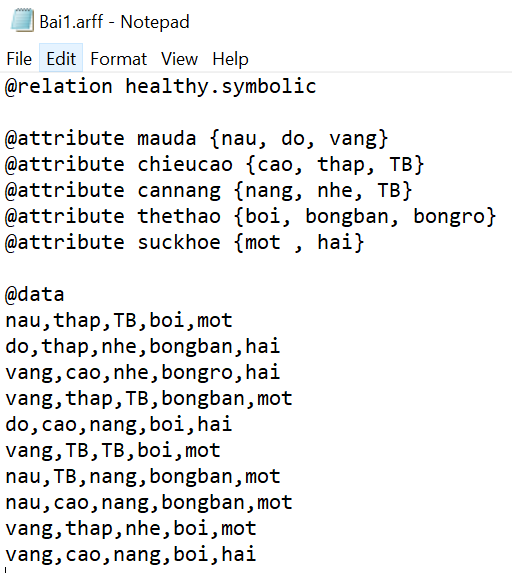
can\_reach(monkey, bananas, chair).

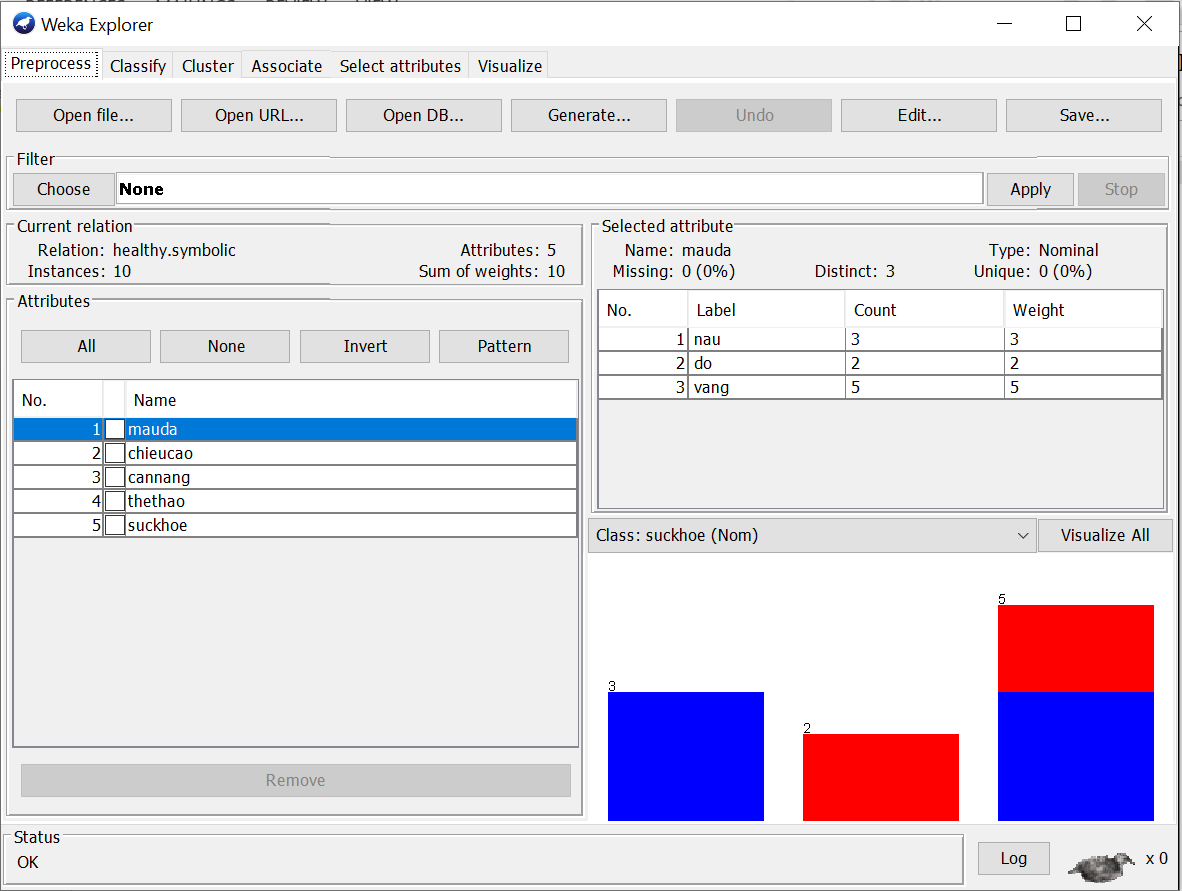


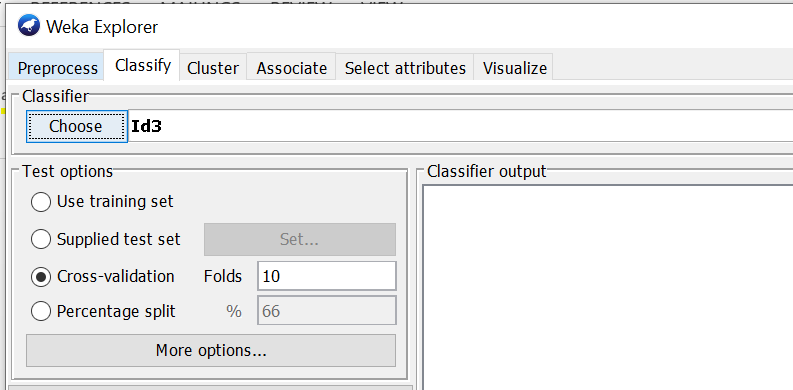
# THUẬT GIẢI QUINLAND TRONG WEKA

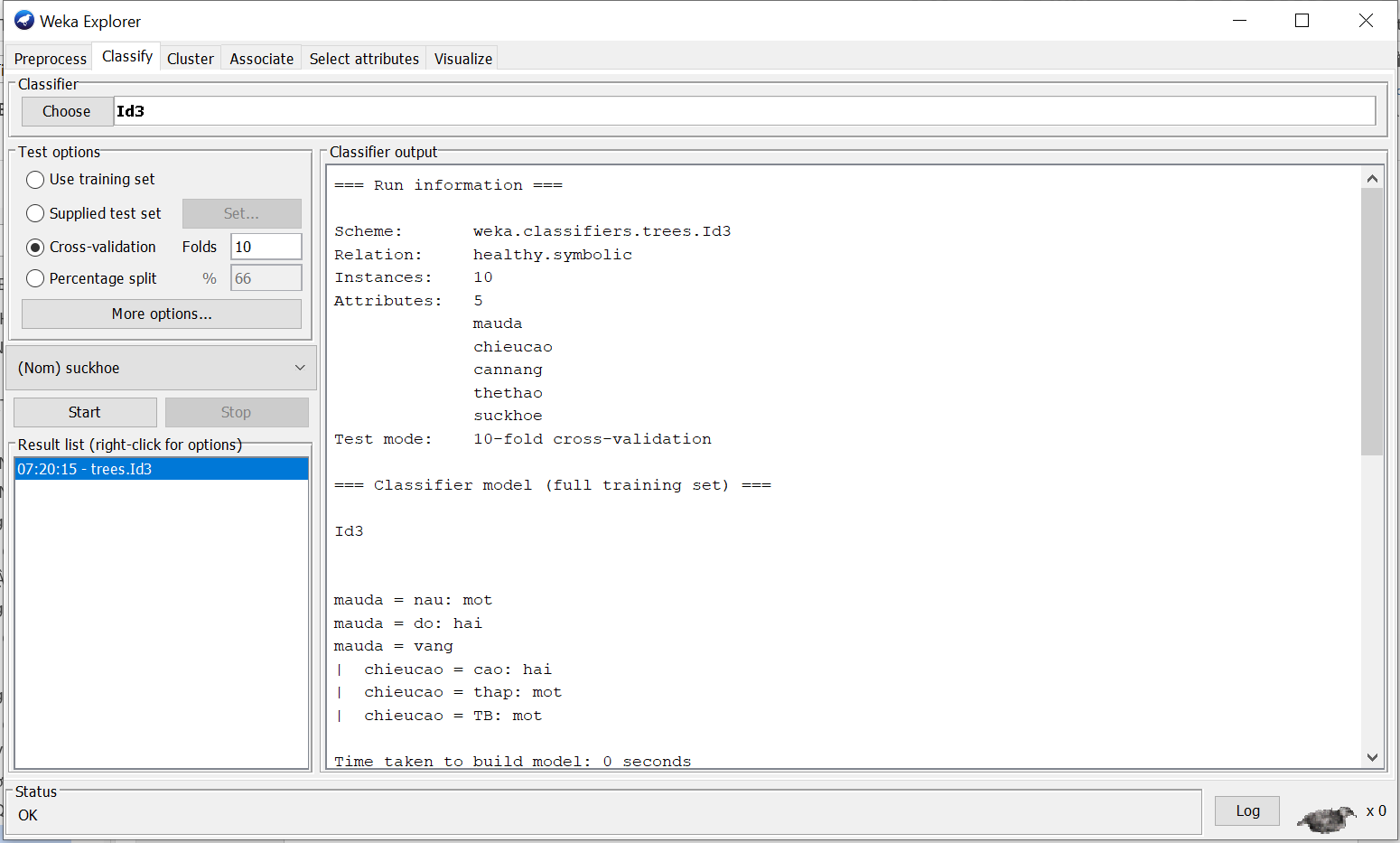
## Yêu cầu bài toán

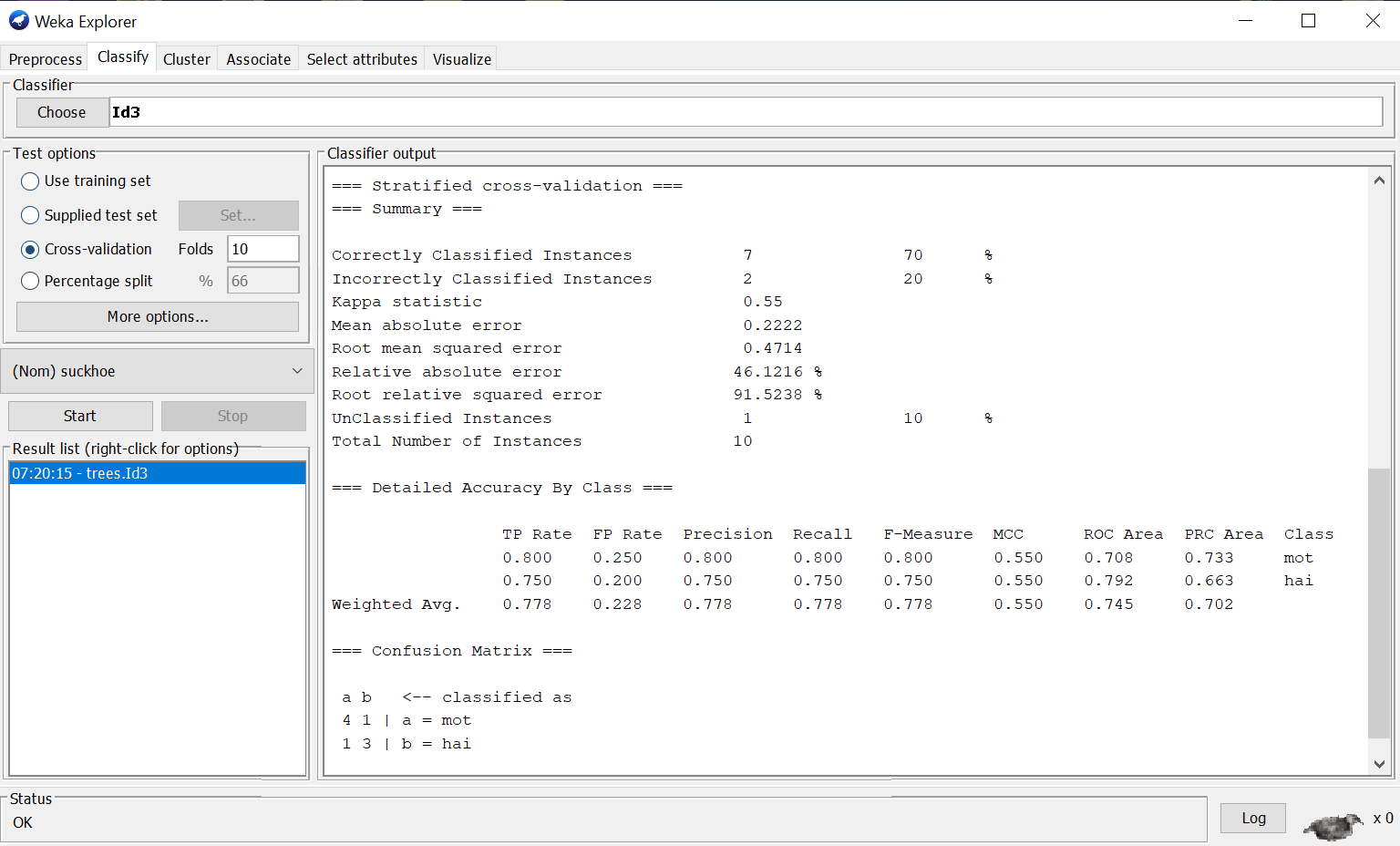












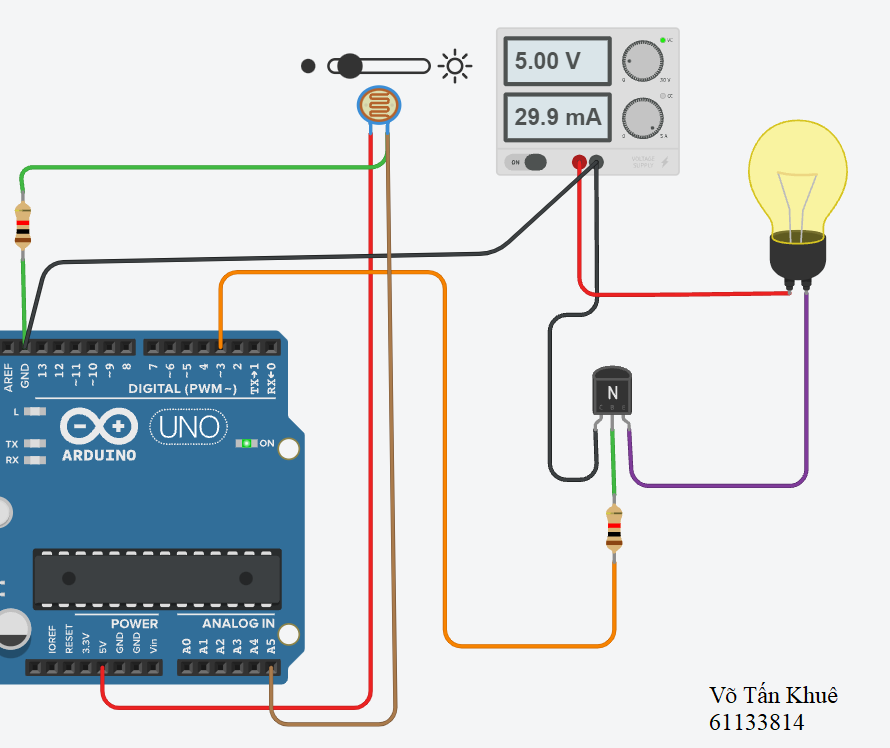
# IOT

## XÂY DỰNG HỆ THỐNG ĐÈN ĐƯỜNG THÔNG MINH

### Tổng quan

Hệ thống được thiết kế bật tắt đèn đường khi có ánh sáng chiếu vào

### Sơ đồ thiết kế



### Đặc điểm của linh kiện

* 2 điện trở: 1 kΩ
* 1 mạch Arduino
* 1 Bulb
* 1 Power supply
* 1 LDR (Light-Dependent Resistor)
* NPN Transistor

### Code chương trình

|  |
| --- |
| int ldr=A5;  int ldr\_giaTri;  void setup()  {  pinMode(3, OUTPUT);  pinMode(ldr, INPUT);  }  void loop()  {  ldr\_giaTri=analogRead(ldr);  if (ldr\_giaTri>512)  digitalWrite(3, LOW);  else  digitalWrite(3, HIGH);  } |

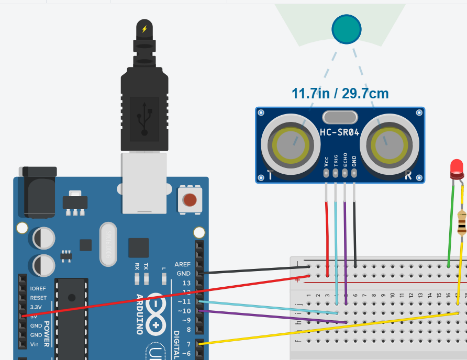
### Kết quả chạy chương trình

## BẬT TẮT ĐÈN LED VỚI CẢM BIẾN HC-SR04

### Mô tả

Hệ thống được thiết kế bật/tắt đèn Led với HC-SR04 được gắn vào chân 10, 11 của Arduino. Đèn chỉ sáng nếu cảm biến nhận thấy vật dưới 50cm.

### Sơ đồ thiết kế



Hình 74. Sơ đồ mạch Proteus

### Đặc điểm của linh kiện

* 1 đèn Led
* 1 điện trở: 100 Ω
* 1 mạch Arduino Uno
* 1 cảm biến HC-SR04

### Code chương trình

|  |
| --- |
| int trig = 11, echo = 10, led = 7; //Khai báo  float distance;  void **setup**()  {  pinMode(trig,OUTPUT);  pinMode(echo,INPUT);  pinMode(led, OUTPUT);  Serial.begin(9600);  }  float **getDistance**(){  digitalWrite(trig, HIGH);  delayMicroseconds(5);  digitalWrite(trig,LOW);  int timer = pulseIn(echo, HIGH);  return timer/58.3f;  }  void **loop**()  {  distance = getDistance();  if(distance < 50){ //Nếu khoảng cách dưới 50cm  digitalWrite(led, HIGH); // Bật đèn  }  else digitalWrite(led, LOW); // tắt đèn  } |

## SMART HOME MINI

### Mô tả

Hệ thống được thiết kế Smart Home Mini gồm có các phòng khách, nhà bếp, nhà xe, cửa ra vào. Cửa sẽ mở khóa nếu dùng RFID Card có giá trị là 225.

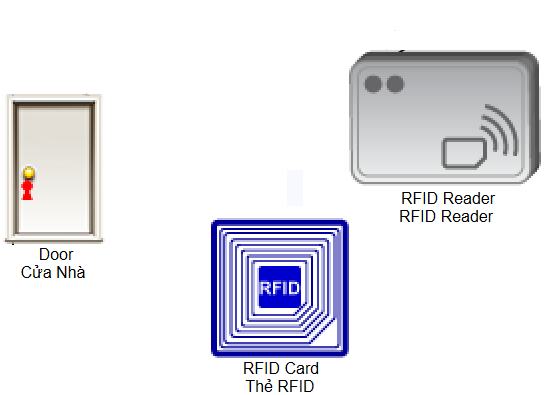
Phòng khách có hệ thống cảm biến nhiệt độ. Nếu nhiệt độ trong phòng lớn hơn 20oC thì cửa sổ sẽ mở, quạt sẽ được bật, đèn sẽ tắt. Nếu nhiệt độ phòng nhỏ hơn 20oC thì cửa sổ sẽ đóng, quạt sẽ tắt, đèn sẽ bật.

Nhà xe có hệ thống cảm biến chuyển động, khói, CO2. Nếu có chuyển động hoặc có khói lớn hơn 0.1 thì cửa nhà xe sẽ được mở. Nếu không có chuyển động và không có khói thì cửa sẽ đóng.

Nhà bếp có hệ thống cảm biến chống cháy. Nếu cảm biến phát hiện đám cháy sẽ kích hoạt vòi phun nước tự động.

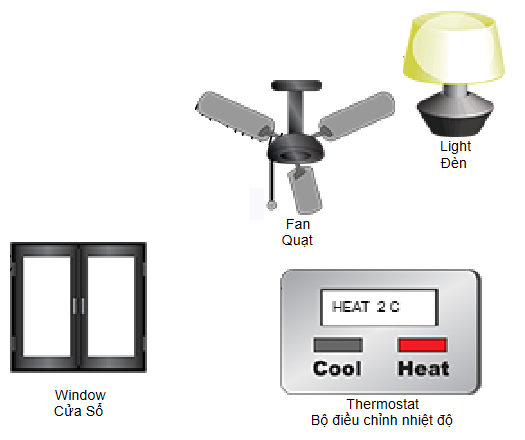
### Sơ đồ thiết kế

* Cửa ra vào



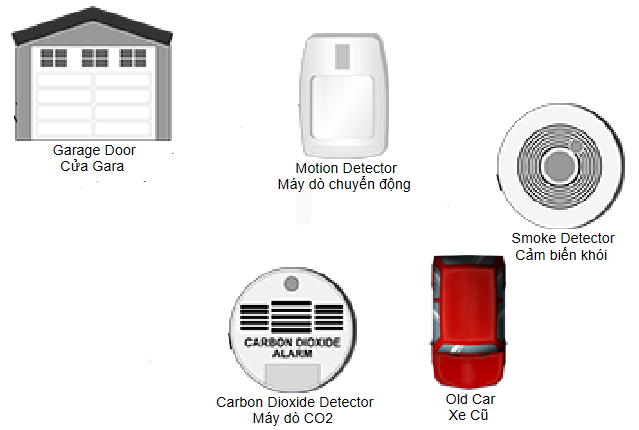
Hình 136. Sơ đồ mạch Cisco cửa ra vào

* Phòng khách



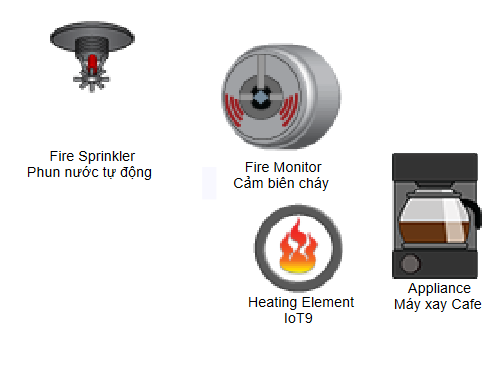
Hình 137. Sơ đồ mạch Cisco phòng khách

* Nhà xe

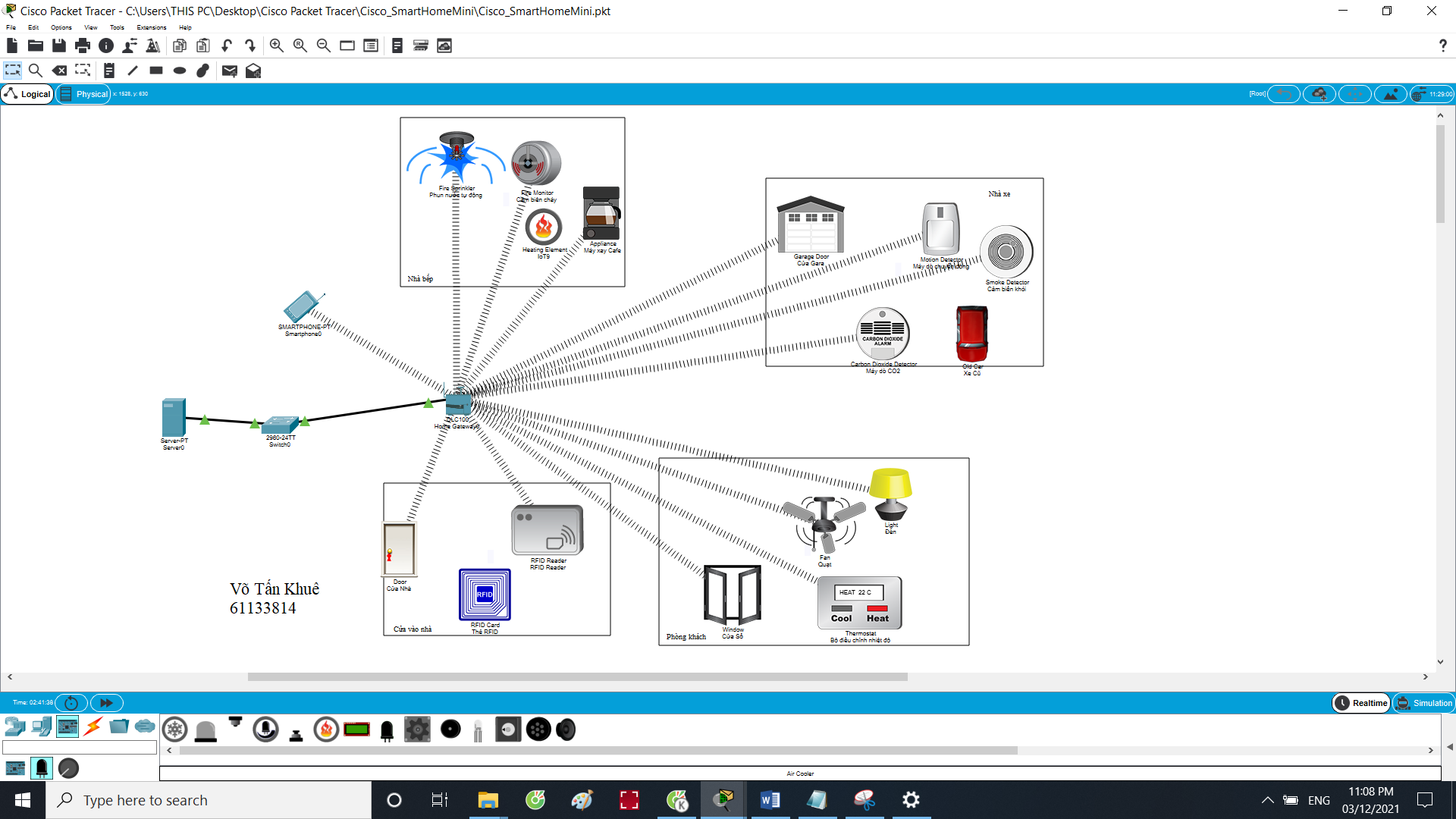


Hình 138. Sơ đồ mạch Cisco nhà xe

* Nhà bếp



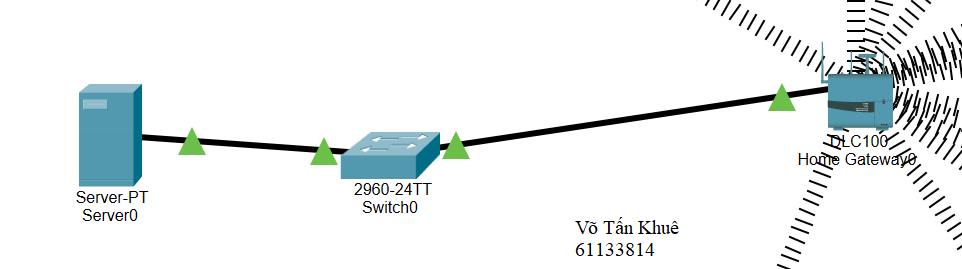
Hình 139. Sơ đồ mạch Cisco nhà bếp



Hình 140. Sơ đồ mạch Cisco Packet Tracer

### Cấu hình mạng và luật

* **Câu hình sever**

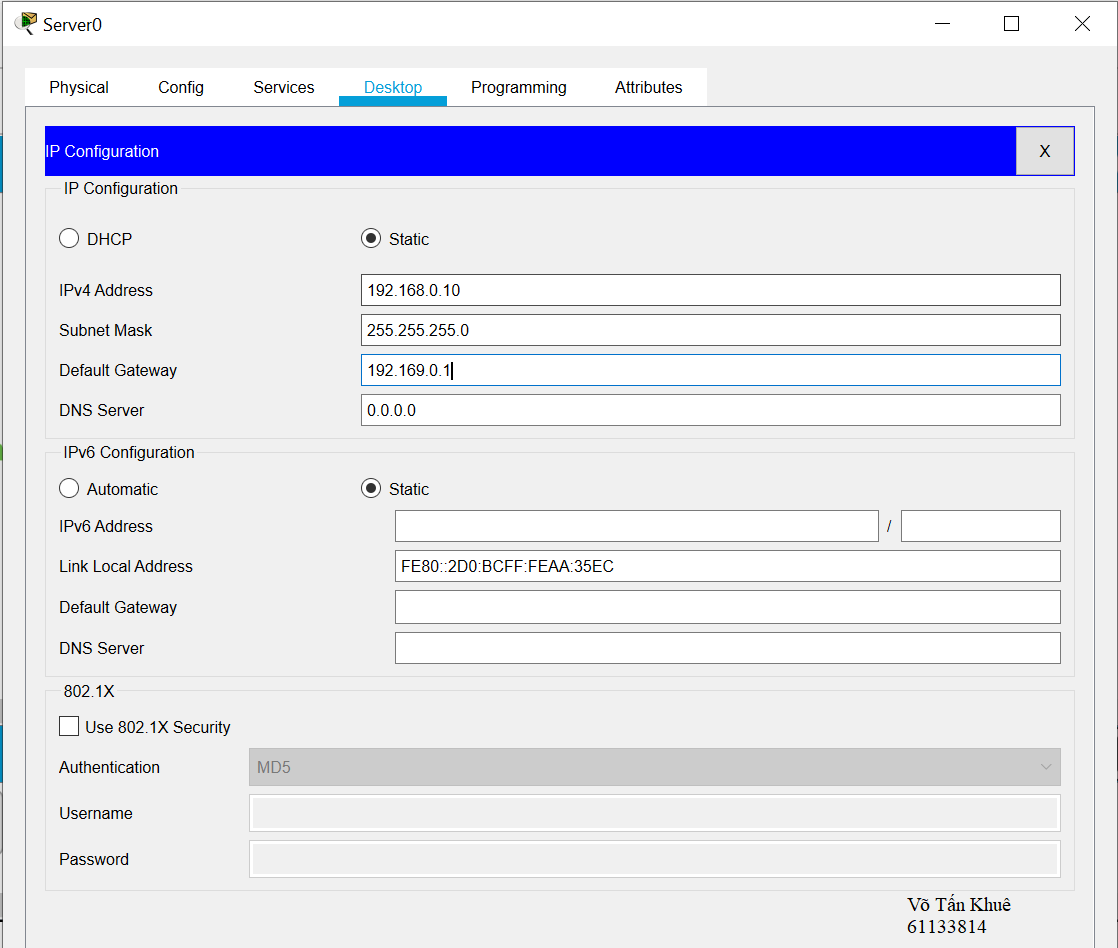


Hình 141. Sơ đồ mạch Cisco Server, Switch, Home Gateway

* Cấu hình Server-PT

+ Địa chỉ IPv4: 192.168.0.10

+ Gateway: 192.169.0.1

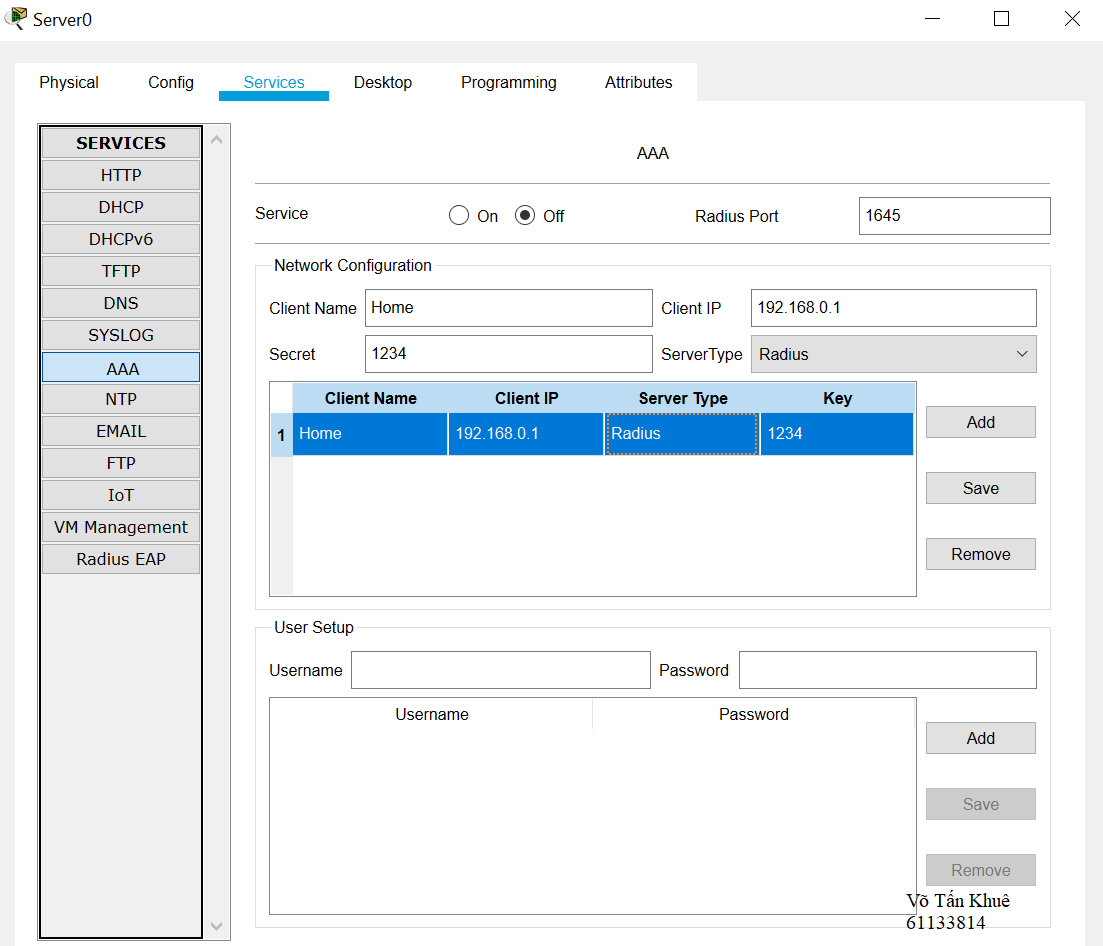


Hình 142. Cấu hình sever

+ Client: Home

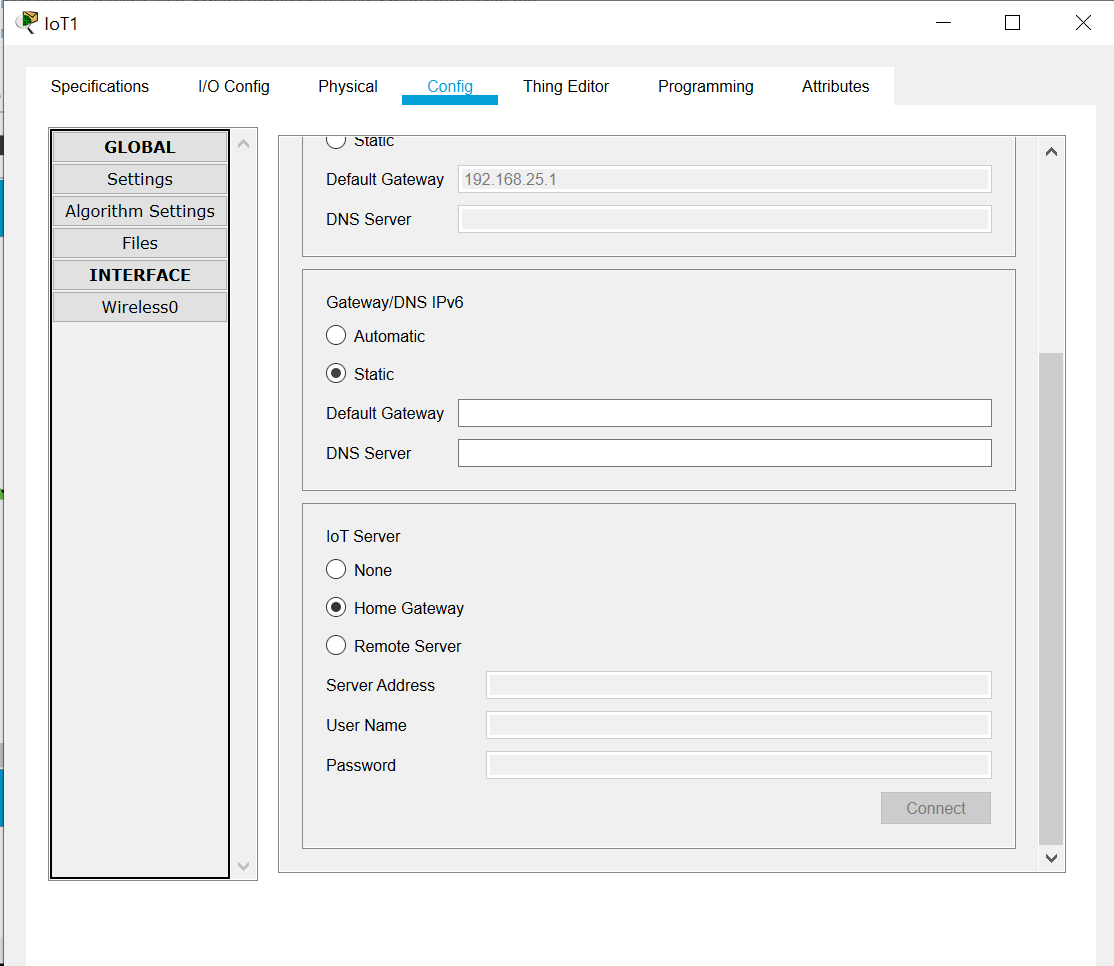
+ Client IP: 192.168.0.1

+ Secret: 1234



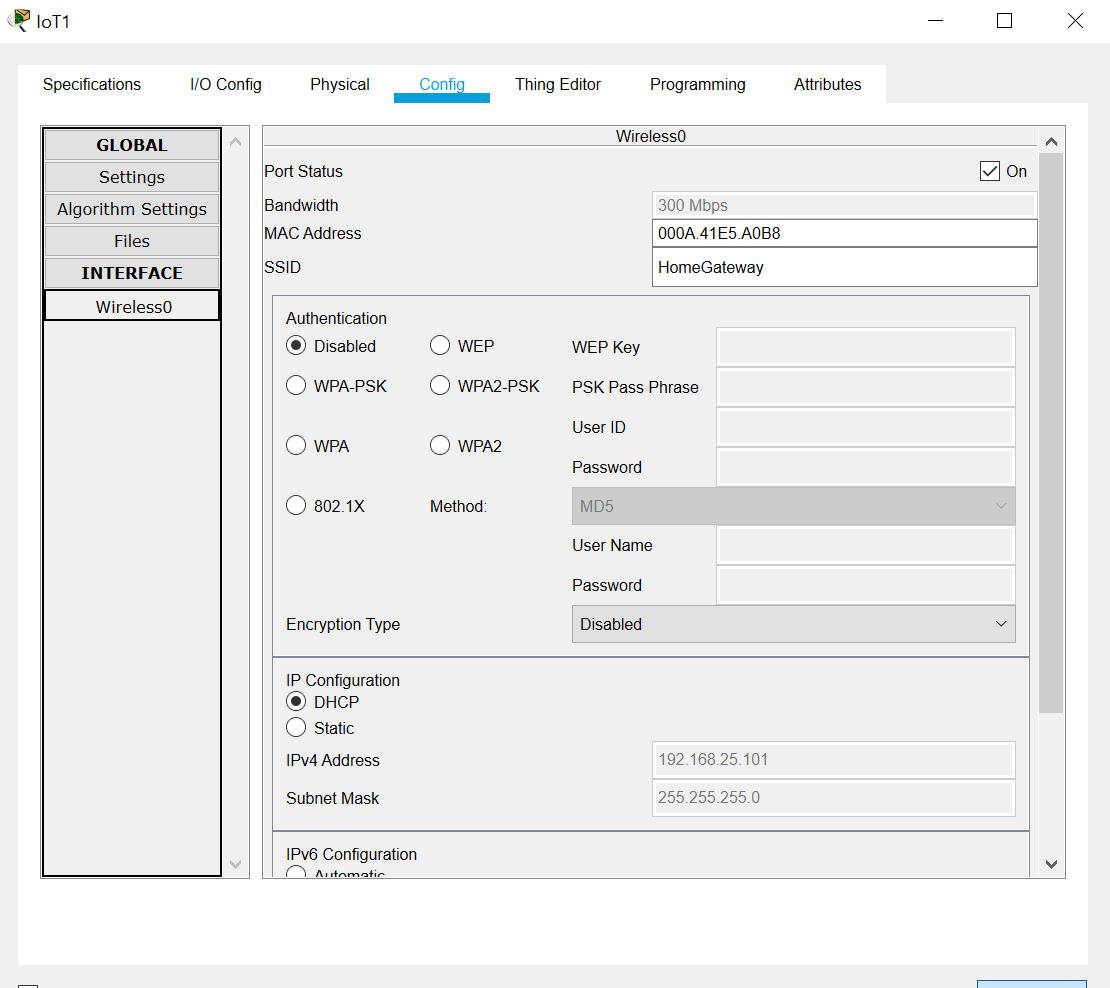
Hình 143. Cấu hình sever

* Chọn Iot Sever là **Home Gateway** cho tất cả các thiết bị trong nhà



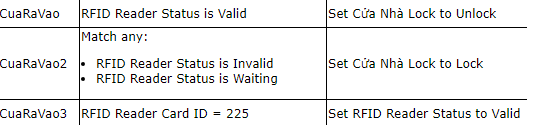
Hình 144. Chọn Home Gateway cho Iot Server

* Kết nối SSID cho tất cả thiết bị trong nhà là **HomeGateway**



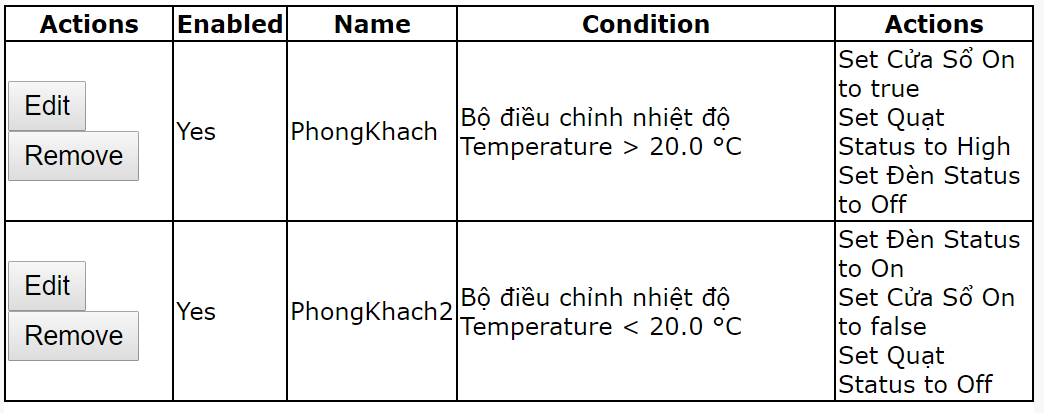
Hình 145. Kết nối SSID

* **Luật**
* Cửa ra vào



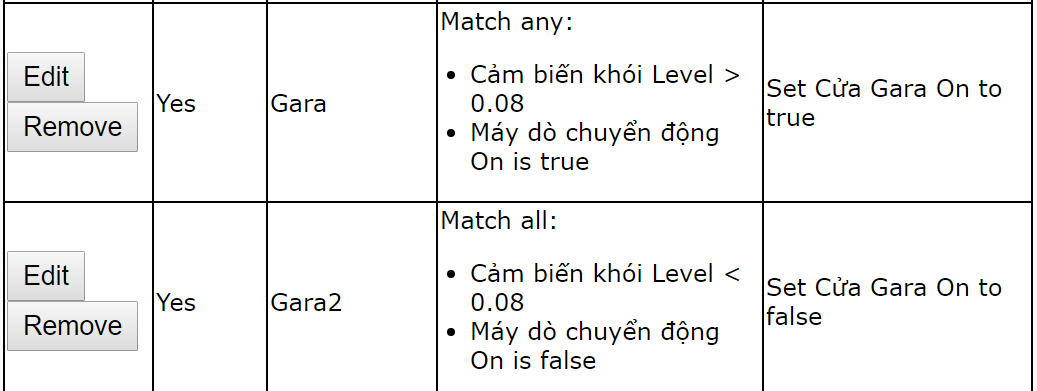
Hình 146. Luật cửa ra vào

* Phòng Khách



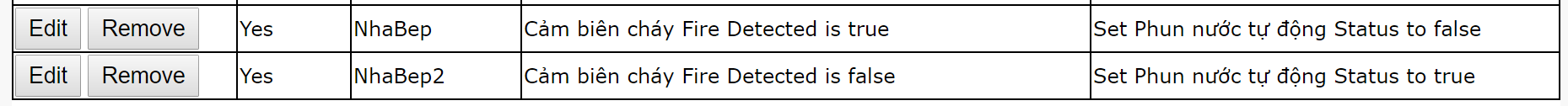
Hình 147. Luật phòng khách

* Nhà Xe



Hình 148. Luật nhà xe

* Nhà bếp



Hình 149. Luật nhà bếp

# TÀI LIỆU THAM KHẢO

|  |  |
| --- | --- |
| [1] | Wikipedia tiếng việt, “Tìm kiếm theo chiều sâu,” 26 07 2022. [Trực tuyến]. Available: https://vi.wikipedia.org/wiki/T%C3%ACm\_ki%E1%BA%BFm\_theo\_chi%E1%BB%81u\_s%C3%A2u. [Đã truy cập 26 07 2022]. |
| [2] | Wikipedia tiếng việt, "Tìm kiếm theo chiều rộng," 03 08 2022. [Online]. Available: https://vi.wikipedia.org/wiki/T%C3%ACm\_ki%E1%BA%BFm\_theo\_chi%E1%BB%81u\_r%E1%BB%99ng. [Accessed 03 08 2022]. |