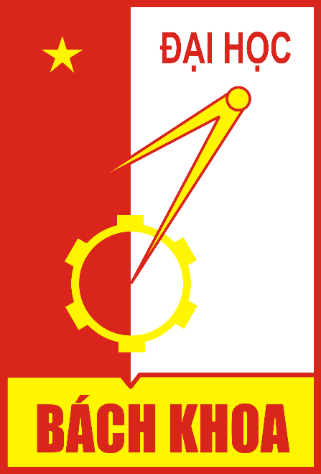
TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

─────── \* ───────



**BÀI TẬP LỚN**

MÔN: Hệ Cơ Sở Tri Thức – IT4361

**Đề Tài: Robot lau nhà**

|  |  |  |
| --- | --- | --- |
| **Nhóm** | | 13 |
| **Thành viên** | Nguyễn Trung Nam  Vũ Đức Nghĩa  Trần Gia Nghĩa  Trần Thị Hải Hà | |
| **Mã lớp:** | | 95033 – K59 |
| **Giảng viên hướng dẫn:** | | **TS.Phạm Văn Hải** |

***Hà Nội, tháng 5 năm 2017***

LỜI NÓI ĐẦU

Thuật ngữ “robot” được Karel Capek sử dụng lần đầu tiên trong vở kịch khoa học viễn tưởng R.U.R. (Rossum's Universal Robots) để mô tả một con người nhân tạo. Nó bắt nguồn từ một tiếng Séc, “Robotnik”, nghĩa là người công nhân. Nhưng giờ đây, “robot” đã không còn là một khái niệm mang tính viễn tưởng, “robot” đã đang và sẽ trở thành một nguồn nhân lực có khả năng hỗ trợ con người trong nhiều lĩnh vực.

Chính vì nhận thấy được tầm quan trọng của “robot”, nhóm chúng em đã lựa chọn đề tài “Robot lau nhà”. Tuy đây là một đề tài đã rất quen thuộc, nhưng vì mới tiếp xúc với các kiến thức mới này nên chúng em có thể vẫn còn có sai xót và nhầm lẫn. Chúng em đã cố gắng học hỏi và áp dụng một phần kiến thức đã học để hoàn thành đề tài này, hi vọng chúng em sẽ nhận được ý kiến đánh giá và góp ý từ thầy và các bạn.

Để hoàn thành được bài tập lớn này, nhóm chúng em xin được gửi lời cảm ơn chân thành đến Thầy giáo hướng dẫn đề tài – **Phạm Văn Hải**, Giảng viên Khoa Công nghệ Thông tin Trường Đại học Bách Khoa Hà Nội - đã hết lòng giúp đỡ, hướng dẫn, chỉ dạy tận tình để nhóm em hoàn thành được đề tài này.

***Hà Nội, tháng 05 năm 2017***

**Chương I: Tổng quan đề tài**

## Giới thiệu đề tài

Robot là một phát minh vĩ đại của loài người. Robot không chỉ là công dụ đắc lực giúp chúng ta thực hiện những công việc khó khăn, mà robot còn đang dần trở nên quen thuộc và gần gũi hơn với chúng ta, giúp đỡ chúng ta trong nhiều công việc hàng ngày như học tập, dọn dẹp, chăm sóc trẻ nhỏ, người già,...Chúng không chỉ có khả năng làm việc không mệt mỏi, tốc độ làm việc nhanh chóng mà còn ngày càng thông minh và có khả năng xử lý công việc chinh xác mà không cần sự điều khiển trực tiếp của con người,…

Từ những công việc khó khăn:



**Hình 1: Robot phẫu thuật**



**Hình 2: Máy bay không người lái**

Cho đến những công việc hàng ngày:



**Hình 3: Robot nấu nướng**

Càng ngày, robot càng trở nên có giá trị hơn trong cuộc sống cũng như trong công việc của chúng ta. Vì vậy, hiểu biết về robot là cần thiết. Để áp dụng kiến thức đã học cũng như tìm hiểu nhiều hơn về đề tài lý thú này, nhóm chúng em đã lựa chọn đề tài “Robot lau nhà”. Tuy đây không phải là một bài toán mới nhưng chúng em sẽ cố gắng để học hỏi cũng như nắm được nhiều kiến thức hữu ích từ đề tài này.

## Mô tả yêu cầu đề tài

Bài toán tìm đường đi bao phủ là bài toán xác định một đường đi qua tất cả các điểm thuộc một khu vực nào đó kết hợp vòng tránh chướng ngại vật nếu có. Đây là vấn đề nghiên cứu quan trọng đối với rất nhiều hệ thống robot có ứng dụng trong thực tế như robot hút bụi, robot cắt cỏ, robot rà phá bom mìn, robot thu hoạch mùa vụ, robot lau cửa kính, robot sơn tự động...

Lấy ví dụ robot lau nhà, khi hoạt động phải đảm bảo làm sạch toàn bộ diện tích sàn làm việc, đồng thời phải phát hiện và tránh được các vật cản như bàn ghế, đồ đạc trong phòng.

Để giải quyết bài toán này cần xây dựng các thuật toán tìm đường đi bao phủ, hay còn được gọi ngắn gọn hơn là thuật toán bao phủ.

Các thuật toán bao phủ cần thỏa mãn các yêu cầu cơ bản như sau:

1. Robot phải đi qua tất cả các điểm trong vùng hoạt động, nói cách khác, đường đi của, robot phải bao phủ toàn bộ vùng hoạt động.

2. Robot không được thực hiện các đường đi chồng lấn lẫn nhau.

3. Robot phải hoạt động liên lục và tuần tự theo các đường đi không lặp lại.

4. Robot phải tránh các vật cản xuất hiện trong vùng hoạt động.

5. Quỹ đạo chuyển động của robot cần đơn giản (hình tròn hoặc đường thẳng) để đơn giản hóa phần điều khiển robot.

6. Đường đi của robot cần được tối ưu hóa dựa vào những điều kiện nhất định.

**Chương II: Thuật toán**

## 2.1 Lựa chọn thuật toán

Các thuật toán bao phủ có thể được phân loại thành thuật toán bao phủ tối ưu hoặc bao phủ đầy đủ. Nếu khả năng bao phủ hoàn toàn vùng làm việc của thuật toán được chứng minh chặt chẽ thì thuật toán được gọi là bao phủ đầy đủ. Trong trường hợp ngược lại, nếu thuật toán nhằm tối đa hóa diện tích bao phủ trong điều kiện robot chịu các ràng buộc như thời gian hoạt động, nguồn năng lượng, kích thước và không thể đảm bảo bao phủ hoàn toàn vùng làm việc, thì thuật toán được gọi là bao phủ tối ưu.

Các thuật toán bao phủ cũng có thể được phân thành hai loại online và offline. Thuật toán offline hoạt động dựa vào các thông tin tĩnh và các thông tin về môi trường cần bảo phủ phải được biết trước khi robot hoạt động. Ngược lại, các thuật toán online h không cần biết trước các thông tin này, mà robot sẽ tự xác định các thông tin về môi trường theo thời gian thực dựa vào các cảm biến được gắn trên robot. Do đó, các thuật toán online cho phép robot hoạt động linh hoạt ngay cả với các môi trường mà robot hoàn toàn không biết trước.

Các phương pháp giải quyết bài toán tìm đường đi:

* Phương pháp phân chia vùng làm việc cổ điển
* Phương pháp phân chia vùng làm việc dựa trên hàm Morse
* Phương pháp phân chia vùng làm việc dựa trên các điểm mốc
* Phương pháp dựa trên lưới
* Phương pháp sử dụng nhiều robot

Phương pháp tìm đường đi dựa trên lưới( grid-based) là hướng tiếp cận mạnh và được lựa chọn nhiều nhất để giải quyết bài toán tìm đường đi bao phủ. Phương pháp này biểu diễn toàn bộ môi trường làm việc của robot dưới dạng ô vuông (grid-map). Mỗi ô trên lưới sẽ nhận một giá trị cho biết tại ô đó là chướng ngại vật, vết bẩn hay vùng trống. Không gian lưới sẽ tương đương với không gian hoạt động của robot và kích thước một ô vuông thường bằng kích thước của robot.

Với phương pháp tiếp cận này, bài toán tìm đường đi bao phủ trở thành bài toán tìm đường đi tối ưu để duyệt qua tất cả các ô không chứa vật cản trên lưới, mỗi ô một lần. Một lớp thuật toán dựa trên cây bao trùm đã được phát triển để giải quyết bài toán này, tiêu biểu trong đó là các thuật toán STC (Spanning Tree Coverage)

STC offline là thuật toán giải quyết 1 bản đồ có sẵn (vị trí đi được và vị trí của các vật cản cố định). Robot sẽ đi bao phủ cả map theo thuật toán STC như ở trên.

STC online giải quyết bản đồ chưa biết trước (không biết vị trí nào có vật cản có định và di động). Thuật toán STC online có thể xử lý tình huống gặp vật cản di động bằng cách áp dụng các cơ sở tri thức vào giải quyết vấn đề.

Để giải quyết bài toán như đề bài cho, em quyết định lựa chọn thuật toán

STC tìm đường đi offline. Trong đó, bản đồ có kích thước xác định và vị trí vật cản cố định được xác định trước. Sử dụng thuật toán STC để tìm đường đi.

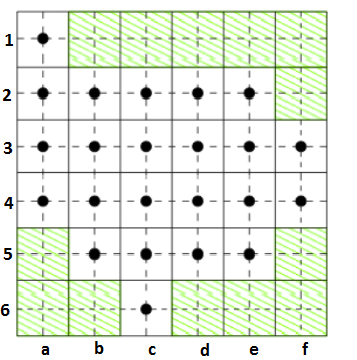
## 2.2 Mô tả thuật toán STC offline

* DxD là kích thước robot.
* Bản đồ là một lưới ô vuông được chia thành các mega-cell kích thước 2x2.

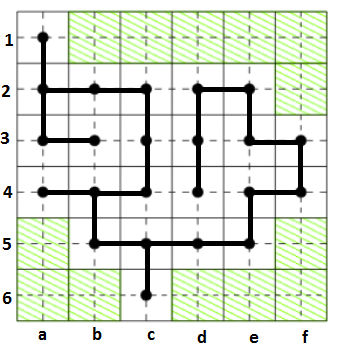
Thuật toán STC-offline là thuật toán tìm đường đi của Robot theo hình xoắn ốc. Đường đi của robot được hình thành từ mega-cell chứa vị trí xuất phát S, rồi đi sang các maga cell còn trống bên cạnh theo cây khung bao trùm.

Thuật toán:

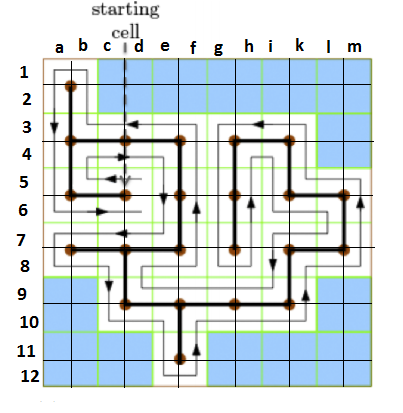
* Bước 1: Chia môi trường thành các lưới có kích thước 2x2. Xác định các điểm khung.



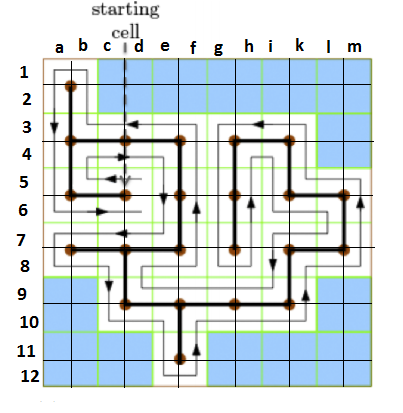
* Bước 2: Sử dụng thuật toán DFS để dụng cây khung



* Bước 3: Chia môi trường thành lưới các ô vuông có kích thước D. Tức là mỗi ô to chia thành 4 ô nhỏ bằng nhau



* Bước 4:Chọn một ô con của S( ô 5-d ) là điểm bắt đầu. Di chuyển giữa các ô con với nhau, theo con đường mà cây bao trùm đã tạo ra đến khi gặp ô bắt đầu thì dừng lại.



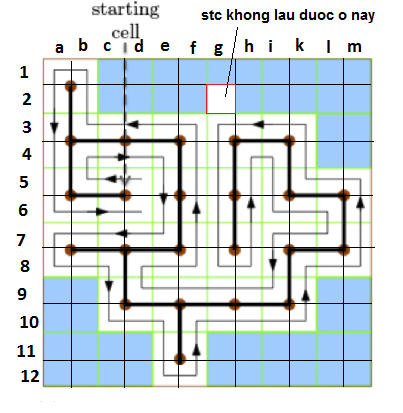
Thuật toán DFS tìm cây khung bao trùm:

Gọi T là cây con bao trùm.

1. Chọn một đỉnh *s* bất kỳ của đồ thị làm gốc của cây **T**. Lúc này cây **T** chỉ có một đỉnh *s*là gốc của **T**., (*s* có mức *0*) và chưa có cạnh nào. Tất cả các đỉnh trong **G** chưa được xét.
2. Lần lượt xét tất cả các đỉnh trong **T** có mức cao nhất nhất chưa xét xong. Mỗi lần xét đỉnh *u*: Tìm một cạnh nối đỉnh *u* với một đỉnh ngoài T.
   1. Nếu không có các cạnh như vậy thì đỉnh u đã được xét xong. Ta quay về đỉnh đứng ngay trước đỉnh u.
   2. Nếu có cạnh *e*=*(u,v)* nối *u* với *v* ngoài **T** thì bổ sung vào **T** cạnh *e* và đỉnh *v*. Nếu*u* có mức *k* thì đỉnh mới bổ sung *v* có mức *k+1*. Đỉnh mới bổ sung v chỉnh là đỉnh có mức cao nhất mới được bổ sung vào **T**.
   3. Quá trình dừng lại khi tất cả các đỉnh nằm trong **T** đã được xét.
3. T là cây bao trùm cần tìm.

## 2.3 Xử lí hạn chế của thuật toán STC:

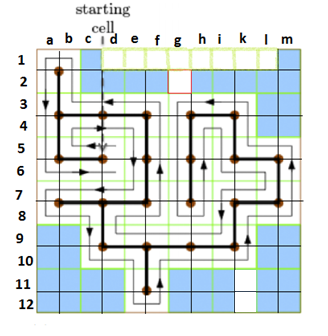
Một vấn đề dễ thấy của hệ thống các thuật toán STC là nó chỉ xử lí được bài toán trong trường hợp các vật cản cố định có kích thước các cạnh bằng 2 lần kích thước của robot lau nhà. Tức là robot không thể lau được những ô không thuộc megacell không năm trong cây bao trùm (như hình dưới). Điều này là không tối ưu vì trong thực tế robot cần lau được tất cả các vị trí có kích thước bằng cạnh của nó miễn là ô đó có thể đi đến được từ vị trí xuất phát ban đầu của robot.



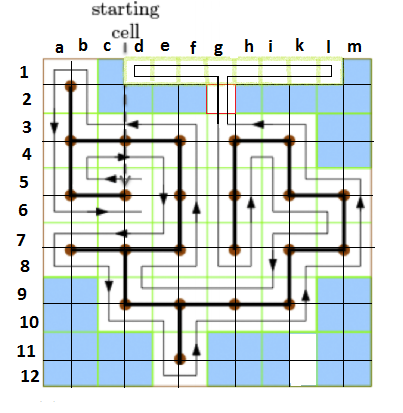
Để lau những ô nằm trong megacell không nằm trong cây bao trùm như ô 2g ở hình trên (gọi là những ô ngoại lệ), nhóm đề xuất cách làm như sau:

Trước hết quy ước những ô nằm mà robot có thể đi đến nhờ thuật toán STC cơ bản là những ô bẩn loại 1 (trong hình dưới là những ô màu trắng trừ ô 2g), những ô không có vật cản và nằm cạnh những ô loại 1 gọi là những ô bẩn loại 2 (trong hình dưới là ô 2g và 11k), những ô bẩn còn lại là những ô bẩn loại 3 (trong hình dưới là ô 1d - 1l và 12k)

Thực hiện thuật toán stc như bình thường, nhưng khi lau đến ô bẩn loại 2 thì sẽ kiểm tra xem nó có liền kề với ô bẩn loại 3 nào không, nếu có thì sẽ lau hết các ô bẩn loại 3 có thể đi đến từ ô bẩn loại 2 này (dùng dfs để lau theo chiều kim đồng hồ), sau đó quay trở lại ô bẩn loại 2 này, quay lại tiếp ô bẩn loại 1 và lau tiếp theo STC cơ bản.

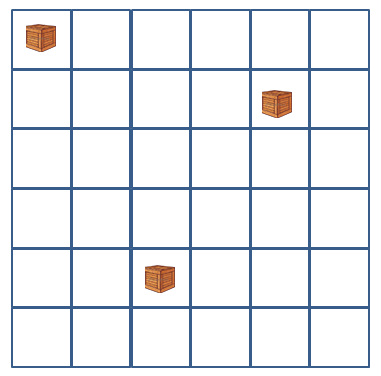


Trong trường hợp như trên, robot khi lau đến ô 3g sẽ lau tiếp như sau:

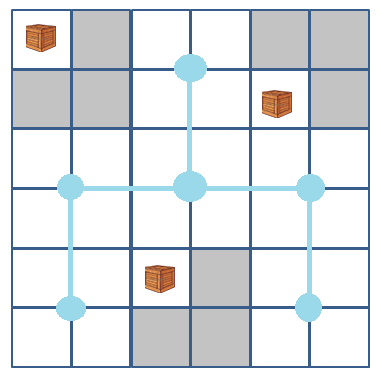


**2.4 Mô tả giải thuật tổng quát bằng ví dụ:**

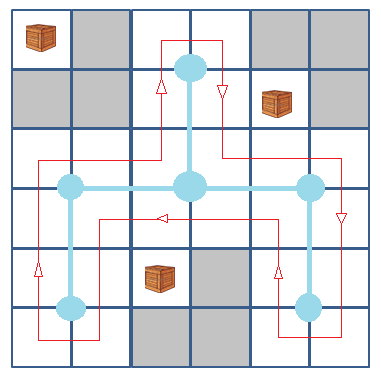
Để miêu tả chi tiết về thuật toán, ở phẩn này sẽ xét 1 ví dụ nhỏ về bản đồ có kích thước 6x6 với các vật cản có vị trí như sau:



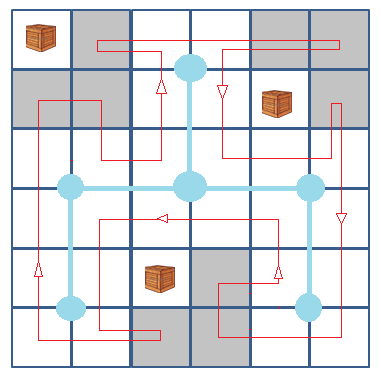
Loại tất cả các megacell chứa vật cản ta có sơ đồ cây bao trùm như hình sau:



Tìm đường đi theo thuật toán STC cơ bản được đường đi như hình sau:



Bước cuối cùng là tìm đường đi vào các ô bẩn loại 2 và loại 3 (những ô tô màu xám):



**2.5 Các luật suy diễn cơ bản:**

Tập luật của chương trình gồm 104 luật suy diễn. Các luật được lưu trong file luat.txt, các sự kiện được lưu trong file sukien.txt (các ô bẩn loại 2 trong file sukien.txt được gọi là “o ngoai le”)

Các luật suy diễn được dùng trong quá trình tạo đường đi cho robot. Một số ví dụ về luật suy diễn cơ bản:

+ 21 2 6 > 103 :nếu ô hiện tại là ô có hàng chẵn và cột chẵn (trong trường hợp cột và hàng được đánh số từ 0) , ô trước ô này là ô bên trên, có megacell ở bên trái -> đi sang bên trái

+ 21 2 17 9 39 > 115 101 : nếu ô hiện tại là ô có hàng chẵn và cột chẵn (trong trường hợp cột và hàng được đánh số từ 0) , ô trước ô này là ô bên trên, không có megacell ở bên trái, có ô bẩn loại 2 ở bên trái, không có ô bẩn loại 2 ở trái dưới -> đi sáng trái rồi quay lại, sau đó đi xuống dưới

**2.6 Mở rộng:**

Lúc robot sắp hết năng lượng, nó sẽ tìm đường đi ngắn nhất từ ô hiện tại đến ô đích để sạc pin.

Lúc đó ta sẽ dụng thuật toán A\* để tìm đường đi cho Robot.

A\* lưu giữ một tập các lời giải chưa hoàn chỉnh, nghĩa là các đường đi qua đồ thị, bắt đầu từ nút xuất phát. Tập lời giải này được lưu trong một hàng đợi ưu tiên (priority queue). Thứ tự ưu tiên gán cho một đường đi x được quyết định bởi hàm f(x) = g(x) + h(x).

Trong đó, g(x) là chi phí của đường đi cho đến thời điểm hiện tại, nghĩa là tổng trọng số của các cạnh đã đi qua. h(x) là hàm đánh giá heuristic về chi phí nhỏ nhất để đến đích từ x. Ví dụ, nếu "chi phí" được tính là khoảng cách đã đi qua, khoảng cách đường chim bay giữa hai điểm trên một bản đồ là một đánh giá heuristic cho khoảng cách còn phải đi tiếp.

Hàm f(x) có giá trị càng thấp thì độ ưu tiên của x càng cao (do đó có thể sử dụng một cấu trúc heap tối thiểu để cài đặt hàng đợi ưu tiên này).

Trong đó, các\_đường\_đi\_tiếp\_theo(p) trả về tập hợp các đường đi tạo bởi việc kéo dài p thêm một nút kề cạnh. Giả thiết rằng hàng đợi được sắp xếp tự động bởi giá trị của hàm f.

"Tập hợp đóng" (đóng) lưu giữ tất cả các nút cuối cùng của p (các nút mà các đường đi mới đã được mở rộng tại đó) để tránh việc lặp lại các chu trình (việc nàycho ra thuật toán tìm kiếm theo đồ thị). Đôi khi hàng đợi được gọi một cách tương ứng là "tập mở". Tập đóng có thể được bỏ qua (ta thu được thuật toán tìm kiếm theo cây) nếu ta đảm bảo được rằng tồn tại một lời giải hoặc nếu hàm các\_đường\_đi\_tiếp\_theo được chỉnh để loại bỏ các chu trình.

Mã giả A\*:

function A\*(điểm\_xuất\_phát,đích)

var đóng := tập rỗng

var q := tạo\_hàng\_đợi(tạo\_đường\_đi(điểm\_xuất\_phát))

while q không phải tập rỗng

var p := lấy\_phần\_tử\_đầu\_tiên(q)

var x := nút cuối cùng của p

if x in đóng

continue

if x = đích

return p

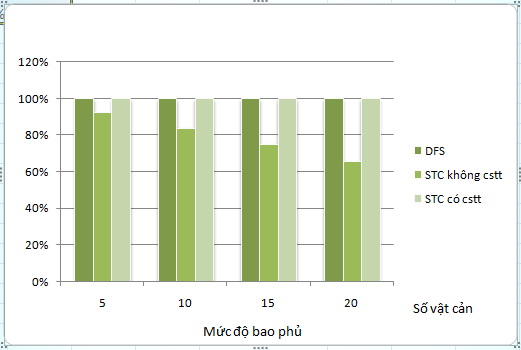
bổ sung x vào tập đóng

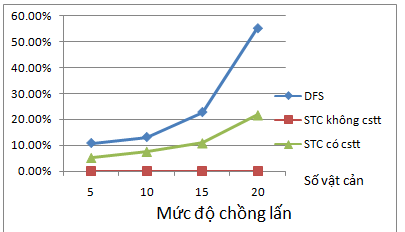
foreach y in các\_đường\_đi\_tiếp\_theo(p)

đưa\_vào\_hàng\_đợi(q, y)

return failure

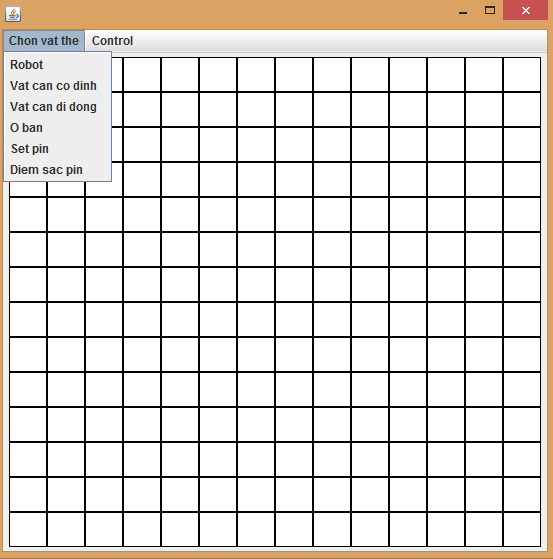
**2.7 So sánh hiệu quả của việc sử dụng stc có cơ sở tri thức so với thuật toán STC thông thường và so với thuật toán dfs**

****

**Chương III: Xây dựng chương trình**

**1.Giao diện cơ bản của chương trình:**

Giao diện khi chạy chương trình:



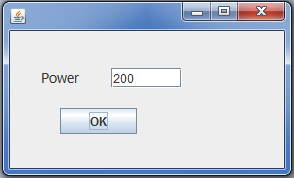
Trong đó:

* Robot: chọn vị trí cho robot
* Vat can co dinh : Chọn vị trí vật cản cố định
* Vat can di dong : Chọn vị trí cho vật cản di động
* O ban: Thiết lập vết bẩn
* Set pin : Nạp năng lượng cho robot
* Diem sac pin: Vị trí nạp năng lượng

Cách sử dụng chương trình:

Chọn lần lượt: robot, các vật cản cố định, các vật cản di động, ô bẩn, điểm sạc pin, sau đó chọn lượng pin cho robot như hình dưới đây, chú ý là sau khi chọn pin xong không tắt giao diện này vì nó sẽ hoạt động như thang đo lượng pin còn lại của robot.

Giao diện chọn năng lượng cho robot:



Hình 1. Giao diện chọn pin, cũng là đồng hồ pin

Chọn Run tại Control, cho robot hoạt động.

***\*Chú ý:***

- Trong control có lựa chọn cho phép vẽ đường đi của robot, lựa chọn này chỉ có tác dụng trong trường hợp lau không pin và không có vật cản di động (để giảm tính phức tạp khi người dùng muốn quan sát đường đi của robot, vì đường đi khi có pin hay có vật cản di động cũng không thay đổi so với không có)

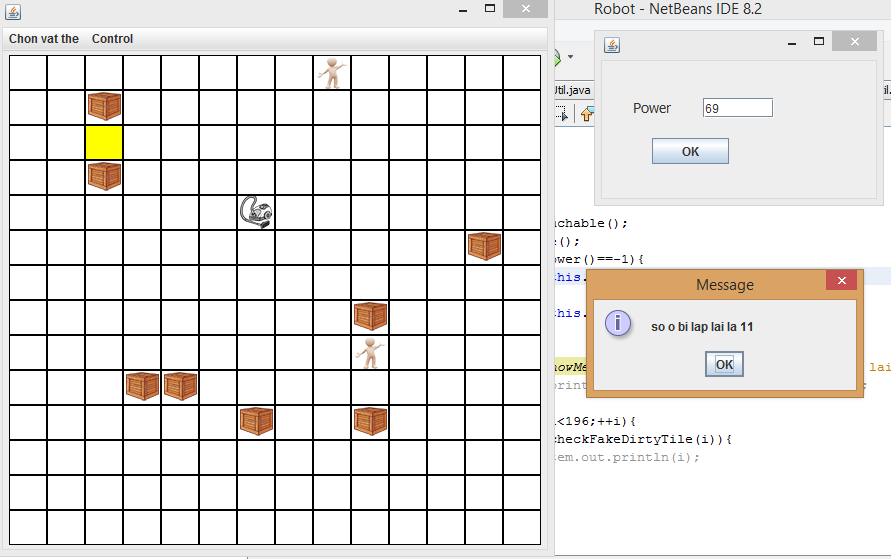
- Chỉ chọn 1 robot và 1 điểm sạc pin.

- Thứ tự lựa chọn các vật thể cần đúng thứ tự.

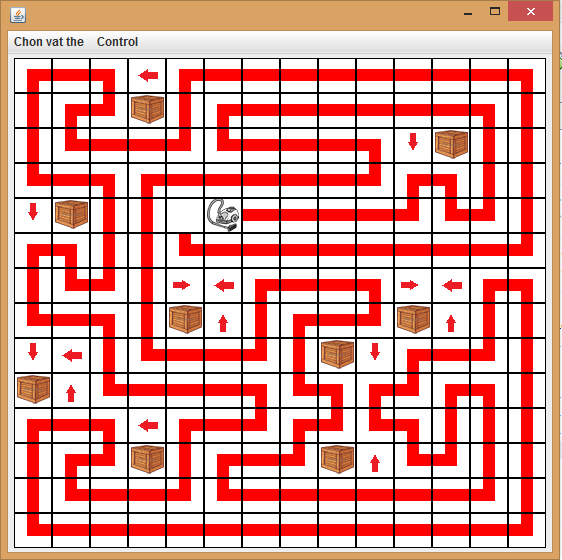
- Khi lau nhà mỗi bước đi robot tốn 2 năng lượng, khi quay về điểm sạc pin robot không lau mà chỉ di chuyển nên chỉ tốn 1 năng lượng, khi pin yếu thì sẽ có thông báo pin yếu (trong giao diện chọn pin) và robot tự dộng quay về sạc.

Giao diện kết quả:

Trong trường hợp có tính đến pin:



Trong trường hợp không tính đến pin:



ở đây những ô có mũi tên nhỏ là những ô ngoại lệ như trong miêu tả thuật toán, mũi tên hướng lên trên có nghĩa là nó được đi tới từ ô bên dưới, các mũi khác cũng tương tự.