

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ GIAO THÔNG VẬN TẢI**

**KHOA CÔNG NGHỆ THÔNG TIN**

-----🙞🙜🕮🙞🙜-----

**Ảnh có chứa văn bản, mẫu họa

Mô tả được tạo tự động**

**BÁO CÁO**

***HỌC PHẦN: TRÍ TUỆ NHÂN TẠO***

**ĐỀ TÀI: QUA SÔNG**

|  |  |  |
| --- | --- | --- |
| **Nhóm sinh viên thực hiện** | **:** | 01 |
| **Thành viên** | **:** | Vũ Mạnh Chiến |
|  |  | Phan Thị Phương Anh |
|  |  | Lê Minh Tâm |
|  |  | Hoàng Anh Tiến |
|  |  | Nguyễn Chí Hải Anh |
| **Lớp** | **:** | 72DCHT21 |
| **Giảng viên hướng dẫn** | **:** | Đoàn Thị Thanh Hằng |

**Hà Nội, 2023**

**MỤC LỤC**

[MỞ ĐẦU 3](#_bookmark0)

[CHƯƠNG I: PHÂN TÍCH BÀI TOÁN 4](#_bookmark1)

1. [Phát biểu bài toán 4](#_bookmark2)
2. [Mô tả bài toán 4](#_bookmark3)

[CHƯƠNG II. THUẬT TOÁN 5](#_bookmark4)

1. [Gioi thiệu thuật toán 5](#_bookmark5)
2. [Mô tả thuật toán của bài toán 5](#_bookmark6)

[4. Ý tưởng 6](#_bookmark7)

1. [GIẢI THUẬT BÀI TOÁN 8](#_bookmark8)
   1. [Cài đặt thuật toán 8](#_bookmark9)

[KẾT LUẬN 14](#_bookmark10)

# MỞ ĐẦU

Bài toán qua sông được giới thiệu trong cuốn “Những bài toán đố Matcova. Chủ đề của bài toán này đã có từ nhiều thế kỷ trước. Cùng với sự phát triển của xã hội, việc di chuyển cả về con người, hàng hóa và thông tin,... càng ngày càng gia tăng. Cùng cới sự gia tăng đó thì yêu cầu cực tiểu về chi phí, khoảng cách, thời gian,.. cũng trở lên quan trọng hơn. Theo lý thuyết đồ thị đó là bài toán tìm đường đi ngắn nhất giữa các đỉnh của 1 đồ thị. Chẳng hạn bài toán tìm đường đi ngắn nhất giữa các thành phố, Bài toán tìm tuyến xe buýt từ địa điểm này đến địa điểm kia sao cho tiết kiệm chi phí nhất,... Bài toán tìm đường đi ngắn nhất càng trở nên cấp thiết và quan trọng đối với cả hiện tại và tương lai. Do sự quan trọng và cấp thiết của bài toán tìm đường đi ngắn nhất nên đã có nhiều nhà toán học đã đưa ra nhiều thuật toán để giải và được vận dụng vào để giải quyết nhiều bài toán thực tế. Trong số các thuật toán được đưa ra thì có thuật toán Prim - tìm cây khung nhỏ nhất.

# CHƯƠNG I: PHÂN TÍCH BÀI TOÁN

## Phát biểu bài toán

Tại bến sông nọ có 3 thầy tu và 3 con quỷ muốn qua sông. Biết rằng tại một thời điểm thuyền chỉ chở tối đa được 2 khách. Nếu bất cứ ở trên bờ nào, bên này hoặc bên kia thì số con quỷ phải bé hơn hoặc bằng số thầy tu, ngược lại quỷ sẽ ăn thịt thầy tu.­

## Mô tả bài toán

Bài toán qua sông có input là 3 thầy tu và 3 con quỷ, trong đó điều kiện ở đây là chỉ có 2 người được qua và số thầy tu ở mỗi bờ không ít hơn số quỷ. Mục tiêu là đưa được cả 3 thầy tu và 3 con quỷ sang bên kia sông. Dựa vào điều kiện có 3 trường hợp khi đưa từ bờ trái sang bờ phải và có 2 trường hợp khi đưa về lại bờ trái. Khi đó, bài toán qua sông trở thành bài toán tìm kiếm lựa chọn tốt nhất.

# CHƯƠNG II. THUẬT TOÁN

## Gioi thiệu thuật toán

Trong khoa học máy tính, thuật toán Best-first Search là một thuật toán tìm kiếm theo bề rộng (Breadth First Search) được hướng dẫn bởi hàm đánh giá. Tư tưởng của thuật toán này là việc tìm kiếm bắt đầu tại nút gốc và tiếp tục bằng cách duyệt các nút tiếp theo có giá trị của hàm đánh giá là thấp nhất so với các

nút còn lại nằm trong hàng đợi.

## Mô tả thuật toán của bài toán

* Bài toán có hai trạng thái:
  + Trạng thái bờ trái {số thầy tu, số quỷ).
  + Trạng thái bờ phải {số thầy tu, số quỷ).
* Mỗi trạng thái có một cách thay đổi:
  + Trạng thái bờ trái: Di chuyển thầy tu hoặc quỷ qua bờ phải.
  + Trạng thái bờ phải: Di chuyển thầy tu hoặc quỷ về lại bờ trái.
* Mỗi cách thay đổi trạng thái sẽ có cách thay đổi cụ thể:
  + Di chuyển thầy tu hoặc quỷ qua bờ phải có ba trường hợp thay đổi trạng

thái:

thái:

+ Đưa 2 thầy tu qua bờ phải.

+ Đưa 2 quỷ qua bờ phải.

+ Đưa 1 thầy tu hoặc 1 quỷ qua bờ phải.

* Di chuyển thầy tu hoặc quỷ về lại bờ trái có hai trường hợp thay đổi trạng

+ Đưa 1 quỷ về bờ trái.

+ Đưa 1 thầy tu và lquỷ về bờ trái.

* + Mỗi lần thực hiện thay đổi trạng thái thì cần phải kiểm tra trạng thái đó có thay đổi đúng hay không?
    - Số thầy tu phải lớn hơn bằng số quỷ hoặc số thầy tu bằng 0 ở mỗi trạng

thái

## 4. Ý tưởng

Bài toán tập trung vào qui tắc : Nếu ở 1 "nơi" nào đó, số Quỷ (Q) nhiều hơn số người (N) thì quỷ sẽ ăn người - nói theo ứng xử trò chơi, đây là trạng thái "thua"

* + còn nói theo tập trạng thái ta đang muốn xây dựng thì đây là "trạng thái vị phạm", và nó sẽ không được xem xét trong tập trạng thái của ta ( sẽ gọi là 1 trạng thái sai, nếu có phát sinh trong quá trình tìm kiếm cũng sẽ bị bỏ qua, không bao giờ được xét đến )

Cách thức biến đổi từ trạng thái này đến trạng thái khác là: Từ bờ bên có thuyền chất 1, 2 N/Q lên thuyền và cho chạy qua bờ bên kia. Trạng thái xung đột sẽ chỉ được xem xét sau khi N/Q đã sang được bên kia.

Xét về "nơi" có thể xảy ra xung đột thì có 3 nơi cả thảy:

- "Bờ 1"

* + "Thuyền"

- "Bờ 2"

Dễ thấy đối với "thuyền" thì theo mô tả bài toán này, sẽ không có xung đột trên thuyền. Đơn giản vì thuyền phải có ít nhất 1 N/Q ( Người hay quỷ) điều khiển mới chạy, và trên thuyền cho phép chở nhiều nhất 2 N/Q. Xét mọi trường hợp, không có xung đột trên thuyền.

Cho nên ta chỉ quan tâm đến tình trạng N/Q 2 bờ, và vị trí của thuyền ở đâu ( Bờ 1/ Bở 2 ?). Một nhận xét nữa là nếu đã biết bờ 1 có x người, y quỉ, ta cũng biết luôn bên bờ 2 có 3-x người, 3-y quỷ. Vậy để phân biệt các trạng thái của bài này ta chỉ cần biết 3 yếu tố:

* + Số người bên bờ 1 (x)
  + Số quỷ bên bờ 1 (y)
  + Thuyền đang ở bên bờ 1 (b) ( true/false)

Như vậy, chỉ cần dùng 1 mảng [3][3][2]- theo thứ tự [số người bờ 1][số quỷ bờ 1][thuyền đang ở bờ 1 ?] là có thể mô tả tất cả các trạng thái có thể có.

Qui tắc xung đột đối với trạng thái [x,y,b], trạng thái là xung đột nếu:

* + Hoặc là (x>0 và y>x) ( xung đột bên bờ 1 )
  + Hoặc là (3-x>0 và 3-y>3-x) ( xung đột bên bờ 2 )

Tập các trạng thái có thể đến từ [x,y,true] (tổng quát ): [x-1,y,false], [x- 2,y, false], [x-1,y-1,false], [x,y-1,false], [x,y-2,false]

( Ngoài ra, thực tế thì phải xem xét các giá trị x, y là bao nhiêu đã, x = 0 hay 1 thì lấy gì mà x-1, x-2 ! )

Tuơng tự, ta có thể xây dựng tập các trạng thái có thể đến từ [x,y,false]

* + Trạng thái đầu : [3,3,true]
  + Trạng thái đích: [0,0,false] Bây giờ chỉ cần dùng BFS, 1 Queue, 1 con trò quay lui để lưu vết.

# III. GIẢI THUẬT BÀI TOÁN

## Cài đặt thuật toán

Sử dụng ngôn ngữ C++

1. **Chương trình.**

#include <conio.h> #include <iostream> using namespace std;

//Define state begin and state end. int brinkLeft[2];

int brinkRight[2]; static int count = 0;

void init(int n) {

// brinkLeft[0]: monk, brinkLeft[1]: devil.

// brinkRight[0]: monk, brinkRight[1]: devil. brinkLeft[0] = n;

brinkLeft[1] = n; brinkRight[0] = 0;

brinkRight[1] = 0;

}

bool check(int monkLeft, int devilLeft, int monkRight, int devilRight, char mode) { if(monkLeft == 0 && devilLeft == 2 && monkRight == 2 && devilRight == 0

&& mode == 'Q')

return false;

else if((monkLeft >= devilLeft || monkLeft == 0) && (monkLeft >= 0 && devilLeft >= 0)

&& (monkRight >= devilRight || monkRight == 0) && (monkRight >= 0 && devilRight >= 0))

return true; return false;

}

void brinkLeftToRight(int n) {

// Have three case: 2 monk; 2 devil; 1 monk and 1 devil pass over. int tempLeft[2], tempRight[2];

for(int i = 0; i < 3; i++) { tempLeft[0] = brinkLeft[0]; tempLeft[1] = brinkLeft[1]; tempRight[0] = brinkRight[0]; tempRight[1] = brinkRight[1]; switch(i) {

case 0: {

tempLeft[0] -= 2;

tempRight[0] += 2; break;

}

case 1: {

tempLeft[1] -= 2;

tempRight[1] += 2; break;

}

case 2: {

tempLeft[0]--; tempLeft[1]--; tempRight[0]++; tempRight[1]++; break;

}

default: break;

}

if(check(tempLeft[0], tempLeft[1], tempRight[0], tempRight[1], 'Q')) { count++;

brinkLeft[0] = tempLeft[0]; brinkLeft[1] = tempLeft[1]; brinkRight[0] = tempRight[0]; brinkRight[1] = tempRight[1]; switch(i) {

case 0: {

cout << count << ". Two Monk : Left --->>>

Right." << endl;

break;

}

Right." << endl;

case 1: {

cout << count << ". Two Devil : Left --->>>

break;

-->>> Right." << endl;

}

break;

}

}

case 2: {

cout << count << ". One Monk and One Devil: Left -

break;

}

default: break;

}

}

void brinkRightToLeft(int n) {

// Have two case: 1 devil; 1 monk and 1 devil return brink Left. int tempLeft[2], tempRight[2];

for(int i = 0; i < 2; i++) { tempLeft[0] = brinkLeft[0]; tempLeft[1] = brinkLeft[1]; tempRight[0] = brinkRight[0]; tempRight[1] = brinkRight[1]; switch(i) {

case 0: {

tempLeft[1]++; tempRight[1]--; break;

}

case 1: {

tempLeft[0]++; tempLeft[1]++; tempRight[0]--; tempRight[1]--; break;

}

default: break;

}

if(check(tempLeft[0], tempLeft[1], tempRight[0], tempRight[1], 'V')) { count++;

brinkLeft[0] = tempLeft[0]; brinkLeft[1] = tempLeft[1]; brinkRight[0] = tempRight[0];

Left." << endl;

brinkRight[1] = tempRight[1]; switch(i) {

case 0: {

cout << count << ". One Devil : Right --->>>

break;

--->>> Left." << endl;

}

break;

}

}

}

}

case 1: {

cout << count << ". One Monk and One Devil: Right

break;

}

default: break;

bool checkComplete(int n, int monk, int devil) { if(monk == n && devil == n)

return true; return false;

}

void display() {

for(int i = 0; i < 2; i++)

cout << brinkLeft[i] << " "; cout << endl;

for(int i = 0; i < 2; i++)

cout << brinkRight[i] << " "; cout << endl;

}

void monkToRiver(int n) { while(1) {

display(); brinkLeftToRight(n);

if(checkComplete(n, brinkRight[0], brinkRight[1])) {

cout << endl << "-------------------------COMPUTER WIN!----------

" << endl;

break;

}

display(); brinkRightToLeft(n);

}

}

int main() {

int n;

cout << "PLEASE ENTER N (MONK - DEVIL): "; cin >> n;

init(n); monkToRiver(n); getch();

return 0;

}

\* Tổ chức dữ liệu

* + n: là biến cho biết số thầy tu và quỷ - monk: là thầy tu
  + devil: là quỷ
  + brinkLeft: là trạng thái bờ trái
  + brinkRight: là trạng thái bờ phải
  + monkLeft là trạng thái di chuyển của thầy tu sang bờ trái
  + devilLeft: là trạng thái di chuyển của con quỷ sang bờ trái
  + monkRight: là trạng thái di chuyển của thầy tu sang bờ phải - devilRight: là trạng thái di chuyển của con quỷ sang bờ phải
  + tempLeft: biến trung gian lưu giá trị khi thay đổi trạng thái
  + tempRight: biến trung gian lưu giá trị khi thay đổi trạng thái

# KẾT LUẬN

Thuật toán Best-first Search được vận dụng để giải quyết bài tốn qua sông đã cho thấy trong thực tế các thuật toán rất cấp thiết và quan trọng trong hiện tại và tương lai. Sử dụng các thuật toán để xử lý các vấn đề cách tối ưu nhất. Qua bài toán qua sông này đã cho chúng em hiểu rất nhiều về môn học trí tuệ nhân tạo và vận dụng vào cuộc sống hàng ngày để sử lý một vấn đề nào đó. Và đặc biệt là chúng em được tiếp thu mọi kiến thức được truyền đạt từ cô Đoàn Thị Thanh Hằng rất hay và bổ ích. Chúng em đã biết được cách tự lập thu thập nhiều thơng tin bỗ ích từ nhiều nguồn để hoàn thành bài tập lớn này. Từ đó học được nhiều bài học ý nghĩa.