**Đề bài: Nghiên cứu cơ sở lý thuyết, ứng dụng và cài đặt giải thuật di truyền để giải bài toán Người du lịch (Traveling Salesman program)**

**Mục Lục**

[Mô tả và đặc tả yêu cầu bài toán 2](#_heading=h.gjdgxs)

[Chương 1: Giới thiệu bài toán người du lịch 3](#_heading=h.30j0zll)

[1.1. Giới thiệu 3](#_heading=h.1fob9te)

[1.2. Mục tiêu nghiên cứu 3](#_heading=h.3znysh7)

[1.3. Phạm vi nghiên cứu 4](#_heading=h.2et92p0)

[1.4. Phương pháp nghiên cứu 4](#_heading=h.tyjcwt)

[Chương 2: Tổng quan về bài toán người du lịch(Traveling Salesman Problem-TSP) 5](#_heading=h.3dy6vkm)

[2.1. Khái niệm bài toán người du lịch 5](#_heading=h.1t3h5sf)

[2.2. Các phương pháp truyền thống giải bài toán người du lịch 6](#_heading=h.4d34og8)

[2.2.1. Thuật giải chính xác 6](#_heading=h.2s8eyo1)

[2.2.2. Các thuật giải gần đúng 8](#_heading=h.17dp8vu)

[Chương 3: Phân tích bài toán Người du lịch 13](#_heading=h.3rdcrjn)

[3.1 Mô hình hóa bài toán TSP. 13](#_heading=h.26in1rg)

[3.2 Các phương pháp giải quyết TSP (so sánh với các thuật toán khác). 13](#_heading=h.lnxbz9)

[3.3 Lý do chọn giải thuật di truyền cho bài toán này. 13](#_heading=h.35nkun2)

[Chương 4: Cài đặt giải thuật di truyền 14](#_heading=h.1ksv4uv)

[4.1 Thiết kế cấu trúc dữ liệu. 14](#_heading=h.44sinio)

[4.2 Chi tiết từng bước cài đặt (khởi tạo, đánh giá, lai ghép, đột biến). 14](#_heading=h.2jxsxqh)

[4.3 Tối ưu hóa tham số của giải thuật. 14](#_heading=h.z337ya)

[Chương 5: Thực nghiệm và đánh giá 14](#_heading=h.3j2qqm3)

[5.1 Thiết lập bài thử nghiệm (các trường hợp kiểm tra). 14](#_heading=h.1y810tw)

[5.2 Phân tích kết quả (so sánh với các thuật toán khác). 14](#_heading=h.4i7ojhp)

[5.3 Đánh giá hiệu quả và thời gian chạy. 14](#_heading=h.2xcytpi)

[Chương 6: Kết luận và hướng phát triển 14](#_heading=h.1ci93xb)

[6.1 Tóm tắt kết quả nghiên cứu. 14](#_heading=h.3whwml4)

[6.2 Hướng phát triển trong tương lai (cải tiến giải thuật, ứng dụng mở rộng). 14](#_heading=h.2bn6wsx)

# Mô tả và đặc tả yêu cầu bài toán

**Bước 1: Phân tích yêu cầu của bài toán**

**Yêu cầu:**

- Mô tả bài toán: Người du lịch cần đi qua N thành phố và trở về thành phố xuất phát, sao cho tổng khoảng cách di chuyển là ngắn nhất.

- Đầu vào: Danh sách các thành phố với tọa độ hoặc khoảng cách giữa các thành phố.

- Đầu ra: Một chuỗi thành phố thể hiện lộ trình tối ưu và tổng khoảng cách tương ứng.

- Giải pháp: Sử dụng giải thuật di truyền để tìm ra lộ trình tối ưu, tận dụng các khái niệm như quần thể, gen, đột biến, lai ghép và lựa chọn.

**Bước 2: Đặc tả yêu cầu**

**1. Đầu vào:**

 - Danh sách các thành phố với tọa độ hoặc ma trận khoảng cách.

   - Số lượng thành phố (N).

   - Tham số cho giải thuật di truyền (kích thước quần thể, tỉ lệ lai ghép, tỷ lệ đột biến).

**2. Đầu ra:**

 - Lộ trình tối ưu (thứ tự các thành phố).

   - Tổng khoảng cách tối thiểu của lộ trình.

**3. Yêu cầu chức năng:**

- Cài đặt giải thuật di truyền (khởi tạo quần thể, đánh giá fitness, lai ghép, đột biến, lựa chọn).

   - Tính toán và tối ưu hóa lộ trình.

   - Hiển thị kết quả (lộ trình và tổng khoảng cách).

**4. Yêu cầu phi chức năng:**

  - Thời gian thực hiện (có thể chạy trong thời gian hợp lý với N lớn).

   - Tính khả thi trong việc mở rộng cho số lượng thành phố lớn.

**Bước 3: Thiết kế đề cương**

# Chương 1: Giới thiệu bài toán người du lịch

## 1.1. Giới thiệu

Đề tài "Nghiên cứu cơ sở lý thuyết, ứng dụng và cài đặt giải thuật di truyền để giải bài toán Người du lịch (Traveling Salesman Problem - TSP)" nhằm tìm hiểu sâu sắc về bài toán TSP, một trong những bài toán nổi tiếng trong lĩnh vực tối ưu hóa và lập lịch. Bài toán này đặt ra yêu cầu tìm đường đi ngắn nhất cho một người du lịch để thăm tất cả các thành phố và trở về thành phố xuất phát. TSP có ứng dụng rộng rãi trong logistics, thiết kế mạng và nhiều lĩnh vực khác. Với sự phát triển nhanh chóng của công nghệ thông tin, nhu cầu tìm kiếm các giải pháp tối ưu cho bài toán TSP ngày càng gia tăng. Giải thuật di truyền, một trong những phương pháp tối ưu hóa heuristic mạnh mẽ, đã cho thấy tiềm năng vượt trội trong việc tìm ra các giải pháp gần đúng cho những bài toán phức tạp như TSP.

## 1.2. Mục tiêu nghiên cứu

* **Nghiên cứu lý thuyết**: Phân tích và tổng hợp các phương pháp truyền thống và hiện đại giải quyết TSP, đặc biệt là giải thuật di truyền.
* **Phát triển ứng dụng**: Khảo sát và minh họa các ứng dụng thực tiễn của giải thuật di truyền trong việc giải quyết bài toán TSP.
* **Cài đặt giải thuật**: Thiết kế và triển khai một giải thuật di truyền để giải quyết bài toán TSP, đánh giá hiệu suất và so sánh với các phương pháp khác.

## 1.3. Phạm vi nghiên cứu

* **Đối tượng nghiên cứu**: Bài toán TSP và các biến thể của nó.
* **Giới hạn về giải thuật**: Tập trung vào giải thuật di truyền, một trong những phương pháp tối ưu hóa heuristic phổ biến.
* **Dữ liệu và mô hình**: Sử dụng dữ liệu mô phỏng và thực tế từ các bài toán TSP điển hình để kiểm tra và đánh giá giải thuật.

## 1.4. Phương pháp nghiên cứu

* **Phân tích tài liệu**: Nghiên cứu các tài liệu khoa học, báo cáo và sách chuyên khảo về TSP và giải thuật di truyền.
* **Mô hình hóa**: Xây dựng mô hình toán học cho bài toán TSP và các yếu tố ảnh hưởng đến việc áp dụng giải thuật di truyền.
* **Thực nghiệm**: Cài đặt giải thuật di truyền bằng ngôn ngữ lập trình phù hợp (như Python, C++) và thử nghiệm trên các tập dữ liệu khác nhau.
* **Đánh giá và so sánh**: Sử dụng các chỉ tiêu hiệu suất (thời gian chạy, độ chính xác) để so sánh giữa giải thuật di truyền và các phương pháp khác như Brute-force, Nhánh và ràng buộc, hoặc các giải thuật heuristic khác.

Đề tài không chỉ có giá trị lý thuyết mà còn có ứng dụng thực tiễn cao, hứa hẹn đóng góp vào việc nâng cao hiệu suất giải quyết bài toán TSP bằng cách áp dụng giải thuật di truyền, mở rộng khả năng ứng dụng trong các lĩnh vực liên quan.

Bài toán Người du lịch (TSP) là một bài toán tối ưu hóa trong lĩnh vực toán học và khoa học máy tính. Cụ thể, bài toán này được định nghĩa như sau:

# Chương 2: Tổng quan về bài toán người du lịch(Traveling Salesman Problem-TSP)

## 2.1. Khái niệm bài toán người du lịch

*Bài toán Người du lịch (TSP) là một bài toán tối ưu hóa trong lĩnh vực toán học và khoa học máy tính. Cụ thể, bài toán này được định nghĩa như sau:*

**Mô tả:**

Cho một tập hợp các thành phố (hoặc điểm) và khoảng cách (hoặc chi phí) giữa mỗi cặp thành phố. Mục tiêu của bài toán là tìm ra một lộ trình ngắn nhất (hoặc có tổng chi phí thấp nhất) mà người du lịch phải đi qua tất cả các thành phố đúng một lần và quay trở lại thành phố xuất phát.

Các yếu tố chính:

* Tập hợp thành phố: Gồm n thành phố.
* Khoảng cách: Một ma trận khoảng cách biểu thị khoảng cách (hoặc chi phí) giữa thành phố
* Lộ trình: Một chuỗi các thành phố ,​​ phải được tìm ra sao cho tổng khoảng cách (chi phí) là nhỏ nhất.

**Mục tiêu:**

Tìm lộ trình tối ưu sao cho tổng khoảng cách là nhỏ nhất:

**Biến thể của bài toán:**

Bài toán TSP có thể có nhiều biến thể khác nhau, chẳng hạn như:

* TSP có ràng buộc: Thêm các ràng buộc về thời gian hoặc tài nguyên.
* TSP với nhiều người du lịch: Nhiều người du lịch cùng đi và phải phân chia thành phố.
* TSP với thời gian: Thêm yếu tố thời gian vào bài toán, yêu cầu tính toán không chỉ khoảng cách mà còn thời gian di chuyển.

Bài toán TSP là một bài toán NP-hard, nghĩa là không có phương pháp giải quyết hiệu quả cho các trường hợp lớn. Vì vậy, nó thường được giải quyết bằng các phương pháp heuristic hoặc các giải thuật tối ưu hóa như **giải thuật di truyền.**

## 2.2. Các phương pháp truyền thống giải bài toán người du lịch

**2.2.1. Thuật giải chính xác**Trong các phương pháp thì đầu tiên phải kể đến thuật toán **vét cạn** . Thuật toán này tìm tất cả các chu trình Hamilton trong đồ thị sau đó chọn một chu trình nhỏ nhất làm đáp án. Việc tìm chu trình Hamilton được thực hiện theo phương pháp duyệt chiều sâu và kết hợp quay lui . Phương pháp cho ta một kết quả tối ưu nhưng độ phức tạp quá cao (O(n!)).

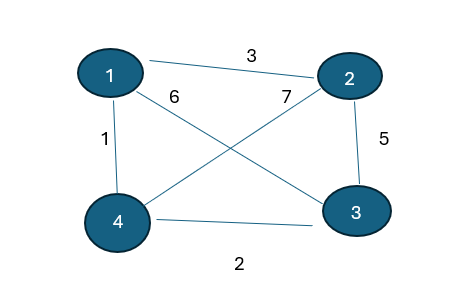
Chu trình hamilton là chu trình đi qua tất cả các đỉnh và sau đó quay về đỉnh xuất phát , do đó ta dùng một danh sách Chuaxet[] để lưu lại các đỉnh chưa xét , một biến sum để lưu lại trong số của chu trình hiện thời , do đó chu trình sẽ không quan trọng điểm xuất phát , ta chọn đỉnh xuất phát là đỉnh có chỉ số nhỏ nhất là 1 .

Quá trình duyệt như sau:

-Ban đầu đưa đỉnh 1 và 0 vào stack, Sum =0 .

- Lặp lại quá trình sau : :Lấy đỉnh từ stack ra là đỉnh i và trọng số ts, gán ChuaDuyet[i] là false và Sum +=ts, nạp 0 0 vào stack sau đó nạp các đỉnh kề j và trọng số tương ứng với đỉnh đang xét mà ChuaDuyet[j]=true, neu i không có đỉnh kề thỏa yêu cầu tức là đã duyệt hết tất cả các đỉnh . Ta cộng trọng số của cạnh i 🡪 1 vào sum và gán min bằng Min(Sum,min) . Sum -= cạnh i 🡪1 .

-Lặp lại quá trình sau : Lấy đỉnh từ stack ra , nếu đỉnh này không có đỉnh kề thỏa yêu cầu thì Pop đỉnh và trọng số này ra , nếu có đỉnh thỏa thì thoát vòng lặp và duyệt tiếp từ đỉnh này , trường hợp lấy đỉnh từ stack rá là 0 thì Pop 2 lần để lấy 2 đỉnh liên tục trong stack ra sau đó tiếp tục vòng lặp



| Stack đỉnh | Stack trọng số | Đỉnh đang xét | Sum | Min |
| --- | --- | --- | --- | --- |
| 1 | 0 |  | 0 | +∞ |
| 1,0,4,3,2 | 0,0,1,6,3 | 1 | 0 | ... |
| 1,0,4,3,2,0,4,3 | 0,0,1,6,3,0,7,5 | 2 | 3 | ... |
| 1,0,4,3,2,0,4,3,0,4 | 0,0,1,6,3,0,7,5,0,2 | 3 | 8 | ... |
| 1,0,4,3,2,0,4,3,0,4,0 | 0,0,1,6,3,0,7,5,0,2,0 | 4 | 8+2+1=11 | 11 |
| 1,0,4,3,2,0,4 | 0,0,1,6,3,0,7 | 4 | 11-1-2-5=3 | ... |
| 1,0,4,3,2,0,4,0,3 | 0,0,1,6,3,0,7,0,2 | 4 | 10 | ... |
| 1,0,4,3,2,0,4,0,3,0 | 0,0,1,6,3,0,7,0,2,0 | 3 | 10+2+6=18 | 11 |
| 1,0,4,3 | 0,0,1,6 | 3 | 18-6-2-7-3=0 | ... |
| 1,0,4,3,0,4,2 | 0,0,1,6,0,2,5 | 3 | 6 | ... |
| 1,0,4,3,0,4,2,0,4 | 0,0,1,6,0,2,5,0,7 | 2 | 11 | ... |
| 1,0,4,3,0,4,2,0,4,0 | 0,0,1,6,0,2,5,0,7,0 | 4 | 11+7+1=19 | 11 |
| 1,0,4,3,0,4 | 0,0,1,6,0,2 | 4 | 19-1-7-5=6 | ... |
| 1,0,4,3,0,4,0,2 | 0,0,1,6,0,2,0,7 | 4 | 8 | ... |
| 1,0,4,3,0,4,0,2,0 | 0,0,1,6,0,2,0,7,0 | 2 | 8+7+3=18 | 11 |
| 1,0,4 | 0,0,1 | 4 | 18-3-7-2-6=0 | ... |
| 1,0,4,0,3,2 | 0,0,1,0,2,7 | 4 | 1 | ... |
| 1,0,4,0,3,2,0,3 | 0,0,1,0,2,7,0,5 | 2 | 8 | ... |
| 1,0,4,0,3,2,0,3,0 | 0,0,1,0,2,7,0,5,0 | 3 | 8+5+6=19 | 11 |
| 1,0,4,0,3 | 0,0,1,0,2,5 | 3 | 19-6-5-7=1 | ... |
| 1,0,4,0,3,0,2 | 0,0,1,0,2,5 | 3 | 3 | ... |
| 1,0,4,0,3,0,2,0 | 0,0,1,0,2,5,0 | 2 | 3+5+3=11 | 11 |

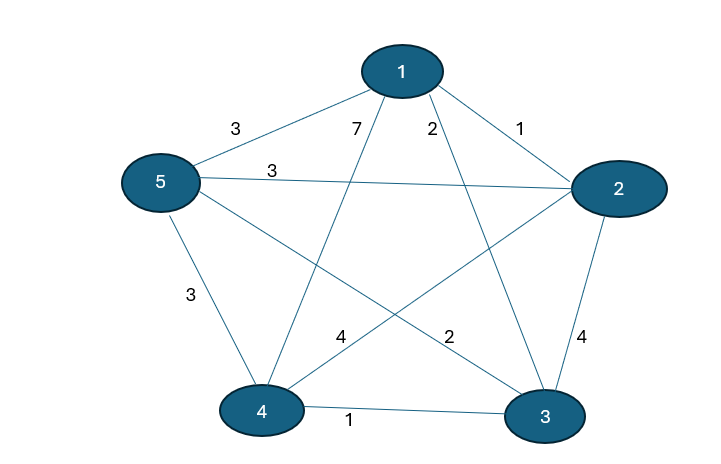
Kết thúc thuật toán ta tìm được chu trình Hamilton là 1🡪2🡪3🡪4🡪1 với độ dài là 11

### 2.2.2. Các thuật giải gần đúng

#### 2.2.2.1Thuật giải tham lam(Thuật giải láng giềng gần nhất)

Thuật giải vét cạn ở trên có mức độ phức tạp quá cao nên trong thực tế người ta chấp nhận cho các giải thuật cho kết quả tốt (nhưng không phải lúc nào cũng tốt ) bởi sự đơn giản , nhanh chóng và cài đặt dễ dàng . Một trong số đó là thuật giải tham lam . Trong thuật toán này tại mỗi bước ta chọn con đường nối với thành phố khác là ngắn nhất với hy vọng rằng n con đường ngắn nhất sẽ cho ta con đường ngắn nhất .

Dưới đây là minh họa cho thuật giải tham lam với n =5



Đáp án của thuật toán tham lam thường tốt ( tuy không phải là tối ưu ) vì đáp án tối ưu là 1🡪3🡪4🡪5🡪2🡪1 với chi phí là 10 so với kết quả của thuật toán tham làm là 14.

**Hàm minh họa cho thuật giải tham lam**

int run(int n, int id){

int min;

int b[100];//Luu lai duong di

int count =0;//So dinh da di qua

int temp;//Chi so cua dinh gan nhat

int s=0;

b[o]=k;

while(count<=n){

min=10000;

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

if(a[id][i]<min && a[id][i]>0 && check(i,b,n)==1)

{

min=a[id][i];

temp=i;

}

if(count<n-1)

s=s+min;

b[count+1]=temp;

id=temp;

count++;

}

s=s+a[id][k];

b[n]=k;

cout << " Duong di ngan nhat: ";

for(int i=0 ;i<n;i++){

cout<<b[i]<<" --> ";

cout<<b[n];

cout<<"Quang duong ngan nhat nguoi du lich nen di la: "<<s;

return 0;

}

}

Thuật giải tham lam có mức độ phức tạp nhỏ hơn nhiều so với thuật giải vét cạn nhưng như đã nói ở trên thì thuật giải này không đúng hoàn toàn .

#### 2.2.2.2.Thuật giải nhánh cận

Thuật toán nhánh cận là phương pháp chủ yếu để giải bài toán tối ưu tổ hợp . Trong quá trình tìm kiếm lời giải ,ta sẽ phân hoạch tập các phương án thành hai hay nhiều nhánh con như cây tìm kiếm ,sau đó đánh giá từng nhánh để quyết định xem nên tiếp tục ở nhánh nào và loại bỏ nhánh mà các phương án của nó không thể là phương án tối ưu .

Trong tiến trình giải bài toán TSP ta sẽ phân tập các hành trình thành 2 tập con

* + - Những hành trình chứa cạnh (i,j)
    - Những hành trình không cạnh chứa (i,j)

Trước tiên để cho việc phân nhánh dễ dàng ta cần phải rút gọn ma trận trọng số , do tổng chi phí của hành trình sẽ chứa đúng một phần tử của mỗi dòng và cột nên khi cộng hay trừ cả dòng hay cột cho một số a thì tổng trọng số cả tất cả các hành trình sẽ giảm đi a . Do đó ta có thể rút gọn ma trận trọng số sao cho trên mỗi dòng và cột đều có ít nhất một số 0 , dồng thời tổng các số a sẽ là cận dưới của tất cả các hành trình.

Ta lần lượt tìm cận dưới của 2 ma trận này , sau đó phân hoạch tiếp trên ma trận có cận dưới nhỏ hơn . Do đó khi chọn cạnh để phân nhánh ta phải chọn làm sao để cận dưới của nhánh khống chứa i,j là lớn nhất , để tìm cạnh đó ta chọn số 0 trong ma trận để khi thay nó bằng ∞ sẽ cho ta tổng hẳng số rút gọn theo cạnh và dòng lớn nhất. Lưu ý khi đã chọn cạnh i,j và j,k thì ta phải ngăn chặn không cho qua cạnh k,i để không hình thành các chu trình con.

Chúng ta sẽ tìm lời giải dựa trên ma trận trọng số .

Ví dụ :

|  | 1 | 2 | 3 | 4 | 5 |
| --- | --- | --- | --- | --- | --- |
| 1 | ∞ | 8 | 5 | 22 | 11 |
| 2 | 4 | ∞ | 9 | 17 | 27 |
| 3 | 15 | 7 | ∞ | 12 | 35 |
| 4 | 5 | 27 | 17 | ∞ | 29 |
| 5 | 23 | 21 | 19 | 7 | ∞ |
| 5 |
| 4 |
| 7 |
| 5 |
| 7 |

Đầu tiên trừ lần lượt mỗi dòng với số nhỏ nhất trên dòng đó hoặc trừ mỗi cột với số nhỏ nhất trên cột đó sao cho trên mỗi dòng hoặc một cột đều có ít nhất một số 0

|  | 1 | 2 | 3 | 4 | 5 |
| --- | --- | --- | --- | --- | --- |
| 1 | ∞ | 3 | 0 | 17 | 6 |
| 2 | 0 | ∞ | 5 | 13 | 23 |
| 3 | 8 | 0 | ∞ | 5 | 28 |
| 4 | 0 | 22 | 12 | ∞ | 24 |
| 5 | 16 | 14 | 12 | 0 | ∞ |

Ta có cận dưới của ma trận này là 5+4+7+5+7=28 . Ta chọn cạnh để phân nhánh là cạnh (5,4) , khi đó ma trận của tập phân hoạch chứa cạnh (5,4) và tập không phân hoạch không chứa cạnh (5,4) lần lượt là :

|  | 1 | 2 | 3 | 5 |
| --- | --- | --- | --- | --- |
| 1 | ∞ | 3 | 0 | 6 |
| 2 | 0 | ∞ | 5 | 23 |
| 3 | 8 | 0 | ∞ | 28 |
| 4 | 0 | 22 | 12 | ∞ |
|  | 1 | 2 | 3 | 4 | 5 |
| 1 | ∞ | 3 | 0 | 17 | 6 |
| 2 | 0 | ∞ | 5 | 13 | 23 |
| 3 | 8 | 0 | ∞ | 5 | 28 |
| 4 | 0 | 22 | 12 | ∞ | 24 |
| 5 | 16 | 14 | 12 | ∞ | ∞ |

Cận dưới 28+6=34

Cận dưới 23+28=51

Ta tiếp tục rút gọn và phân nhánh tập phân hoạch chứa cạnh (5,4)

|  | 1 | 2 | 3 | 5 |
| --- | --- | --- | --- | --- |
| 1 | ∞ | 3 | 0 | 0 |
| 2 | 0 | ∞ | 5 | 17 |
| 3 | 8 | 0 | ∞ | 22 |
| 4 | 0 | 22 | 12 | ∞ |
|  | 1 | 2 | 3 | 5 |
| 1 | ∞ | 3 | 0 | 0 |
| 2 | 0 | ∞ | 5 | 17 |
| 3 | 8 | 0 | ∞ | 22 |
| 4 | 0 | 22 | 12 | ∞ |

Cận dưới của tập phân hoạch không chứa (1,5) là 17+34=51 . Chọn cạnh (1,5) thì cận dưới của tập phân hoạch chứa (5,4) và chứa (1,5) là 34 . Đồng thời ngăn việc hình thành chu trình con bằng cách cho C[4][1]= ∞ . Tiếp tục rút gọn ta lại được ma trận sau có cận dưới là 12+34=46.

|  | 1 | 2 | 3 |
| --- | --- | --- | --- |
| 2 | 0 | ∞ | 5 |
| 3 | 8 | 0 | ∞ |
| 4 | ∞ | 22 | 12 |
|  | 1 | 2 | 3 |
| 2 | 0 | ∞ | 0 |
| 3 | 8 | 0 | ∞ |
| 4 | ∞ | 15 | 0 |

Phân nhánh cạnh (3,2) ta được ma trận 2x2 của tập phân hoạch chứa cạnh (5,4),(1,5),(3,2) có cận dưới là 46

|  | 1 | 2 | 3 |
| --- | --- | --- | --- |
| 2 | 0 | ∞ | 0 |
| 3 | 8 | 0 | ∞ |
| 4 | ∞ | 15 | 0 |
|  | 1 | 3 |
| 2 | 0 | ∞ |
| 4 | ∞ | 0 |

Cuối cùng ta thêm vào hành trình 2 cạnh cuối là (2,1) và (4,3) . Như vậy các cạnh của hành trình tìm được là (5,4),(1,5),(3,2),(2,1),(4,3) và chu trình cần tìm có chi phí là 7+11+7+4+17=46

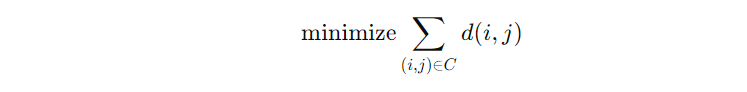
# Chương 3: Phân tích bài toán Người du lịch

## 3.1 Mô hình hóa bài toán TSP.

Bài toán Người du lịch (TSP) là một trong những bài toán tối ưu hóa tổ hợp điển hình. TSP được mô tả bằng một đồ thị hoàn chỉnh, trong đó các đỉnh biểu diễn các thành phố và các cạnh biểu diễn khoảng cách hoặc chi phí di chuyển giữa các thành phố. Nhiệm vụ là tìm một chu trình Hamilton, tức là một lộ trình đi qua tất cả các thành phố một lần và quay trở lại điểm xuất phát, sao cho tổng chi phí hoặc khoảng cách là nhỏ nhất.

* Mô hình toán học:

Cho đồ thị đầy đủ G=(V,E) với tập đỉnh V là các thành phố, và tập cạnh E là các đường nối giữa các thành phố với độ dài tương ứng.

Mục tiêu là tìm một chu trình C bao gồm tất cả các đỉnh sao cho tổng chi phí di chuyển là nhỏ nhất:  
 Trong đó, d(i,j) là khoảng cách giữa thành phố i và thành phố j.

* Các yếu tố cần thiết:

Tập hợp các thành phố: Gồm nnn thành phố cần đi qua.

Ma trận khoảng cách: Một ma trận n×n thể hiện khoảng cách hoặc chi phí giữa các thành phố.

Chu trình tối ưu: Là chuỗi các thành phố được duyệt theo thứ tự sao cho tổng chi phí là nhỏ nhất.

## 3.2 Các phương pháp giải quyết TSP (so sánh với các thuật toán khác).

Có nhiều phương pháp truyền thống và hiện đại để giải quyết bài toán TSP, bao gồm:

#### 3.2.1 Phương pháp chính xác

Phương pháp này cố gắng tìm ra lời giải tối ưu bằng cách duyệt tất cả các khả năng có thể. Điển hình của phương pháp này là thuật toán vét cạn (Brute Force) và thuật toán Nhánh và cận (Branch and Bound). Tuy nhiên, các phương pháp chính xác có độ phức tạp cao và khó áp dụng cho các bài toán có kích thước lớn.

* Brute Force: Xét tất cả các hoán vị của các thành phố để tìm chu trình có tổng khoảng cách nhỏ nhất. Độ phức tạp là O(n!)O(n!)O(n!), nghĩa là không khả thi khi số thành phố nnn lớn.
* Branch and Bound: Một cải tiến từ Brute Force nhằm cắt giảm không gian tìm kiếm bằng cách loại bỏ các nhánh không tiềm năng, nhưng vẫn có độ phức tạp cao đối với các bài toán lớn.

#### 3.2.2 Các phương pháp heuristic và gần đúng

Khi bài toán TSP quá lớn để có thể giải chính xác, các phương pháp heuristic và gần đúng có thể được sử dụng để tìm ra các giải pháp gần tối ưu trong khoảng thời gian hợp lý.

* Thuật toán Greedy (Láng giềng gần nhất): Chọn thành phố gần nhất từ thành phố hiện tại và tiếp tục cho đến khi tất cả các thành phố được đi qua. Phương pháp này đơn giản và nhanh nhưng không đảm bảo lời giải tối ưu.
* Thuật toán quay lui (Backtracking): Kết hợp giữa Brute Force và các phương pháp cắt tỉa không gian tìm kiếm.
* Các thuật toán heuristic khác như Simulated Annealing, Thuật toán kiến, và Tabu Search cũng được sử dụng rộng rãi để tìm các lời giải tiệm cận với lời giải tối ưu.

## 3.3 Lý do chọn giải thuật di truyền cho bài toán này.

Giải thuật di truyền (Genetic Algorithm - GA) là một trong những phương pháp heuristic mạnh mẽ và hiệu quả để giải quyết các bài toán tối ưu hóa tổ hợp như TSP. Lý do chọn giải thuật di truyền cho bài toán TSP có thể được giải thích qua các điểm sau:

1. Khả năng tìm kiếm đa hướng:

Giải thuật di truyền bắt đầu từ một quần thể các lời giải ngẫu nhiên, do đó có thể khám phá nhiều vùng trong không gian lời giải đồng thời. Điều này giúp tăng khả năng tìm ra lời giải tốt hơn so với các thuật toán đơn hướng như Greedy.

1. Tính linh hoạt:

Giải thuật di truyền không đòi hỏi cấu trúc toán học phức tạp của bài toán, có thể áp dụng cho nhiều biến thể của TSP, chẳng hạn như TSP với ràng buộc thời gian hoặc chi phí.

1. Cơ chế tiến hóa:

Sử dụng các khái niệm tiến hóa tự nhiên như lai ghép (crossover), đột biến (mutation) và chọn lọc (selection), giải thuật di truyền có thể cải thiện các lời giải qua từng thế hệ, hướng đến lời giải tối ưu hoặc gần tối ưu.

1. Hiệu quả với bài toán lớn:

Với các bài toán có số lượng thành phố lớn, giải thuật di truyền có thể tìm ra các giải pháp gần tối ưu trong thời gian ngắn hơn nhiều so với các thuật toán chính xác, nhờ cơ chế heuristic và khả năng tối ưu hóa toàn cục.

1. Dễ dàng tối ưu hóa tham số:

Giải thuật di truyền cho phép tùy chỉnh nhiều tham số như kích thước quần thể, tỷ lệ lai ghép, tỷ lệ đột biến, giúp người dùng có thể tối ưu hóa để đạt được hiệu suất tốt nhất cho bài toán cụ thể.

Tóm lại, giải thuật di truyền là một lựa chọn phù hợp để giải bài toán TSP nhờ tính linh hoạt, khả năng tìm kiếm toàn cục và hiệu quả khi làm việc với các bài toán lớn mà các thuật toán truyền thống khó có thể xử lý trong thời gian hợp lý.

# Chương 4: Cài đặt giải thuật di truyền

## 4.1 Thiết kế cấu trúc dữ liệu.

Trong quá trình cài đặt giải thuật di truyền để giải quyết bài toán Người du lịch (TSP), việc lựa chọn cấu trúc dữ liệu phù hợp là rất quan trọng để đảm bảo hiệu suất và tính khả thi của hệ thống. Các cấu trúc dữ liệu chính bao gồm:

* Quần thể (Population):

Quần thể là tập hợp các cá thể, mỗi cá thể đại diện cho một lộ trình khả dĩ. Mỗi cá thể sẽ được biểu diễn dưới dạng một chuỗi các thành phố. Ví dụ: một cá thể có thể được biểu diễn là một mảng các số nguyên, trong đó mỗi phần tử tương ứng với một mã thành phố.

Dữ liệu quần thể có thể được lưu trữ dưới dạng mảng hoặc danh sách liên kết (ArrayList), với mỗi phần tử trong quần thể là một cá thể.

* Cá thể (Individual):

Một cá thể được biểu diễn bằng một chuỗi hoán vị của các thành phố, ví dụ: [1, 3, 2, 5, 4]. Điều này đại diện cho một lộ trình đi qua các thành phố 1, 3, 2, 5, 4 theo thứ tự.

Dữ liệu cá thể có thể được lưu trữ dưới dạng mảng hoặc danh sách các số nguyên (mã thành phố).

* Hàm fitness:

Hàm fitness đánh giá mức độ tối ưu của một cá thể. Fitness trong bài toán TSP thường được tính bằng nghịch đảo của tổng khoảng cách lộ trình.

Cấu trúc này có thể được cài đặt dưới dạng một phương thức nhận đầu vào là một cá thể và trả về một giá trị số nguyên hoặc số thực đại diện cho fitness.

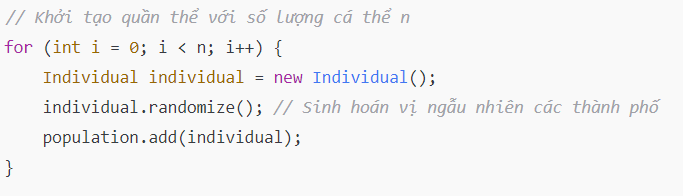
* Khoảng cách giữa các thành phố (Distance Matrix):

Khoảng cách giữa các thành phố sẽ được lưu trữ trong một ma trận hai chiều, với Distance[i][j] đại diện cho khoảng cách giữa thành phố i và j.

## 4.2 Chi tiết từng bước cài đặt (khởi tạo, đánh giá, lai ghép, đột biến).

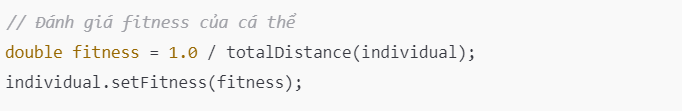
#### 4.2.1 Khởi tạo quần thể

* Quần thể ban đầu được khởi tạo bằng cách tạo ra một số lượng nhất định các cá thể ngẫu nhiên. Mỗi cá thể là một chuỗi hoán vị của các thành phố.
* Ví dụ: Khởi tạo quần thể với 100 cá thể, mỗi cá thể là một hoán vị ngẫu nhiên của danh sách các thành phố.



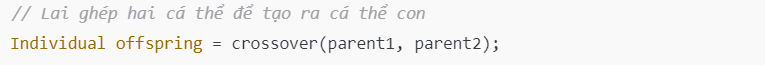
#### 4.2.2 Đánh giá (Evaluation)

* Đánh giá fitness của mỗi cá thể trong quần thể. Fitness của cá thể được tính bằng cách tính tổng khoảng cách của lộ trình mà cá thể đại diện và sau đó lấy nghịch đảo của tổng khoảng cách đó.



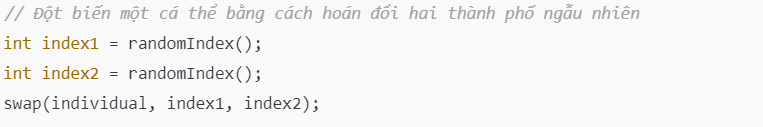
#### 4.2.3 Lai ghép (Crossover)

* Lai ghép là quá trình kết hợp hai cá thể bố mẹ để tạo ra cá thể con mới. Trong TSP, một phương pháp lai ghép phổ biến là Order Crossover (OX), đảm bảo rằng tất cả các thành phố đều xuất hiện đúng một lần trong lộ trình của cá thể con.
* Ví dụ: Lai ghép hai cá thể bố mẹ để tạo ra cá thể con mới.



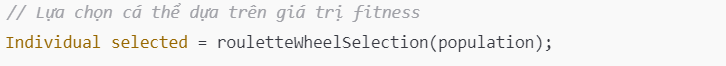
#### 4.2.4 Đột biến (Mutation)

* Đột biến là quá trình thay đổi ngẫu nhiên một số thành phần của cá thể nhằm duy trì tính đa dạng cho quần thể. Một phương pháp đột biến phổ biến là Swap Mutation, trong đó hai thành phố trong lộ trình của cá thể được hoán đổi vị trí cho nhau.



#### 4.2.5 Lựa chọn (Selection)

* Sau khi đánh giá fitness của quần thể, cần lựa chọn các cá thể tốt hơn để tiếp tục tham gia vào quá trình lai ghép và đột biến. Một phương pháp lựa chọn phổ biến là Roulette Wheel Selection, trong đó xác suất lựa chọn cá thể tỷ lệ thuận với giá trị fitness của nó.

4.3 Tối ưu hóa tham số của giải thuật.Việc tối ưu hóa các tham số của giải thuật di truyền là rất quan trọng để đảm bảo rằng hệ thống hoạt động hiệu quả và tìm được lộ trình tối ưu trong thời gian ngắn nhất. Các tham số chính cần tối ưu bao gồm:

## Kích thước quần thể (Population Size): Kích thước quần thể càng lớn thì tính đa dạng của các cá thể càng cao, nhưng điều này cũng đòi hỏi nhiều thời gian tính toán hơn. Cần tối ưu hóa để tìm được kích thước quần thể hợp lý.

## Tỉ lệ lai ghép (Crossover Rate): Tỉ lệ lai ghép quyết định bao nhiêu cá thể trong quần thể sẽ tham gia vào quá trình lai ghép. Thông thường, tỉ lệ này dao động từ 0.7 đến 0.9.

## Tỉ lệ đột biến (Mutation Rate): Tỉ lệ đột biến quy định bao nhiêu cá thể sẽ được đột biến trong mỗi thế hệ. Nếu tỉ lệ đột biến quá thấp, quần thể sẽ nhanh chóng bị đồng nhất, còn nếu tỉ lệ đột biến quá cao, hệ thống sẽ mất tính ổn định. Tỉ lệ đột biến thường được đặt từ 0.01 đến 0.1.

## Số thế hệ (Number of Generations): Số thế hệ quyết định độ dài của quá trình tiến hóa. Nếu số thế hệ quá ít, hệ thống có thể không kịp tìm ra lộ trình tốt nhất, nhưng nếu quá nhiều, hệ thống sẽ tốn nhiều tài nguyên mà không cải thiện đáng kể kết quả.

## Để tối ưu hóa các tham số này, có thể sử dụng các phương pháp như thử nghiệm nhiều giá trị khác nhau và quan sát hiệu suất của hệ thống, hoặc áp dụng các kỹ thuật tối ưu hóa tự động như Grid Search hoặc Random Search.

# Chương 5: Thực nghiệm và đánh giá

## 5.1 Thiết lập bài thử nghiệm (các trường hợp kiểm tra).

## 5.2 Phân tích kết quả (so sánh với các thuật toán khác).

## 5.3 Đánh giá hiệu quả và thời gian chạy.

# Chương 6: Kết luận và hướng phát triển

## 6.1 Tóm tắt kết quả nghiên cứu.

## 6.2 Hướng phát triển trong tương lai (cải tiến giải thuật, ứng dụng mở rộng).

**Tài liệu tham khảo**

**- Danh sách tài liệu, sách và bài báo liên quan đến TSP và giải thuật di truyền.**

Việc tối ưu hóa các tham số của giải thuật di truyền là rất quan trọng để đảm bảo rằng hệ thống hoạt động hiệu quả và tìm được lộ trình tối ưu trong thời gian ngắn nhất. Các tham số chính cần tối ưu bao gồm:

* **Kích thước quần thể (Population Size)**: Kích thước quần thể càng lớn thì tính đa dạng của các cá thể càng cao, nhưng điều này cũng đòi hỏi nhiều thời gian tính toán hơn. Cần tối ưu hóa để tìm được kích thước quần thể hợp lý.
* **Tỉ lệ lai ghép (Crossover Rate)**: Tỉ lệ lai ghép quyết định bao nhiêu cá thể trong quần thể sẽ tham gia vào quá trình lai ghép. Thông thường, tỉ lệ này dao động từ 0.7 đến 0.9.
* **Tỉ lệ đột biến (Mutation Rate)**: Tỉ lệ đột biến quy định bao nhiêu cá thể sẽ được đột biến trong mỗi thế hệ. Nếu tỉ lệ đột biến quá thấp, quần thể sẽ nhanh chóng bị đồng nhất, còn nếu tỉ lệ đột biến quá cao, hệ thống sẽ mất tính ổn định. Tỉ lệ đột biến thường được đặt từ 0.01 đến 0.1.
* **Số thế hệ (Number of Generations)**: Số thế hệ quyết định độ dài của quá trình tiến hóa. Nếu số thế hệ quá ít, hệ thống có thể không kịp tìm ra lộ trình tốt nhất, nhưng nếu quá nhiều, hệ thống sẽ tốn nhiều tài nguyên mà không cải thiện đáng kể kết quả.

Để tối ưu hóa các tham số này, có thể sử dụng các phương pháp như thử nghiệm nhiều giá trị khác nhau và quan sát hiệu suất của hệ thống, hoặc áp dụng các kỹ thuật tối ưu hóa tự động như **Grid Search** hoặc **Random Search**.