**TRƯỜNG ĐẠI HỌC KỸ THUẬT CÔNG NGHỆ CẦN THƠ**

KHOA CÔNG NGHỆ THÔNG TIN

**CHUYÊN NGÀNH KHOA HỌC MÁY TÍNH**

**★★★🕮★★★**

**A white circle with blue text and a book and a symbol

Description automatically generated**

ĐỀ TÀI ĐỒ ÁN:

**TÌM HIỂU VỀ GA VÀ ỨNG DỤNG**

Giảng viên hướng dẫn: **LÊ ANH NHÃ UYÊN**

Sinh viên thực hiện: **LÊ PHƯỚC HỮU 2100450**

**NGUYỄN MỸ HẰNG 2101166**

***Cần Thơ, tháng 12, năm 2023***

**TRƯỜNG ĐẠI HỌC KỸ THUẬT CÔNG NGHỆ CẦN THƠ**

KHOA CÔNG NGHỆ THÔNG TIN

**CHUYÊN NGÀNH KHOA HỌC MÁY TÍNH**

**★★★🕮★★★**

**A white circle with blue text and a book and a symbol

Description automatically generated**

ĐỀ TÀI ĐỒ ÁN:

**TÌM HIỂU VỀ GA VÀ ỨNG DỤNG**

Giảng viên hướng dẫn: **LÊ ANH NHÃ UYÊN**

Sinh viên thực hiện: **LÊ PHƯỚC HỮU 2100450**

**NGUYỄN MỸ HẰNG 2101166**

***Cần Thơ, tháng 12, năm 2023***

**LỜI CẢM ƠN**

Trong lần đầu tiên chúng em thực hiện đồ án 1 của học kỳ một năm thứ ba tại Trường Đại Học Kỹ Thuật Công Nghệ Cần Thơ, thật may mắn khi được giảng viên là cô Lê Anh Nhã Uyên đồng thời cũng là cố vấn học tập của chúng em đã đồng hành, hướng dẫn chúng em trong quá trình thực hiện đồ án 1 này. Với sự chân thành, nhiệt quyết của Cô chúng em muốn gửi lời cảm ơn chân thành đến cô Lê Anh Nhã Uyên vì đã cùng chúng em hoàn thành đồ án 1 này.

Tuy nhiên, kiến thức về đề tài GA (Genetic Algorithm) của chúng em vẫn còn những hạn chế nhất định. Do đó, không tránh khỏi những thiếu sót trong quá trình hoàn thành bài đồ án này. Mong cô xem và góp ý để đồ án của chúng em được hoàn thiện hơn.

Kính chúc cô hạnh phúc và thành công hơn nữa trong sự nghiệp trồng người. Kính chúc cô luôn dồi dào sức khỏe để tiếp tục dìu dắt nhiều thế hệ học trò đến những bến bờ tri thức.

Em xin chân thành cảm ơn!

**TÓM TẮT**

**ABTRACT**

**LỜI CAM ĐOAN**

**MỤC LỤC**

**DANH MỤC BẢNG**

**DANH MỤC HÌNH**

**DANH MỤC TỪ VIẾT TẮT**

# **Chương 1**

**GIỚI THIỆU**

1. **Giới thiệu tổng quan và lý do chọn đề tài**
2. **Giới thiệu tổng quan**

Giải thuật, trong bối cảnh của nền khoa học máy tính và trí tuệ nhân tạo, đang là trung tâm của sự hiểu biết và ứng dụng của chúng ta về các vấn đề phức tạp. Được xây dựng trên nền tảng của toán học và lý thuyết máy tính, giải thuật không chỉ là một công cụ mà còn là một phương pháp khoa học đối với việc giải quyết các vấn đề từ những đề tài cơ bản đến những thách thức nâng cao. Các thuật toán sắp xếp, tìm kiếm, và đồ thị, ngày càng trở nên phức tạp và hiệu quả, thúc đẩy khả năng xử lý dữ liệu và tìm kiếm thông tin nhanh chóng. Trong lĩnh vực của máy học, giải thuật đóng vai trò quyết định trong việc phát triển mô hình dự đoán và phân loại, với khả năng tự học và cải thiện qua thời gian.

Từ các nghiên cứu cơ bản đến ứng dụng thực tế, giải thuật đang là một vai trò không thể phủ nhận trong sự phát triển của nền công nghiệp 4.0. Trong thời đại của dữ liệu lớn, các nhà nghiên cứu ngày càng tập trung vào việc phát triển các thuật toán có khả năng xử lý và rút trích thông tin từ lượng lớn dữ liệu phức tạp. Ngoài ra, giải thuật còn làm nền tảng cho nhiều lĩnh vực mới nổi như trí tuệ nhân tạo, robot học, và các ứng dụng IoT (Internet of Things). Sự kết hợp giữa trí tuệ nhân tạo và giải thuật đã mở ra không gian cho việc phát triển các hệ thống tự động, từ xe tự lái đến robot hợp tác. Với tính chất động lực và tính ứng dụng cao, giải thuật không chỉ là nguồn đề xuất các phương pháp mới mẻ cho cộng đồng nghiên cứu mà còn là công cụ hữu ích cho các nhà phát triển và doanh nghiệp. Trong tương lai, sự tiếp tục của sự phát triển trong lĩnh vực này sẽ chắp cánh cho nhiều khám phá và đổi mới khoa học, đóng góp quan trọng vào sự hiểu biết và sáng tạo của con người.

1. **Lý do chọn đề tài**

Trong quá trình học tập và nghiên cứu về chuyên ngành khoa học máy tính, nhận thấy rằng những phương pháp, những kỹ thuật được ứng dụng để tìm kiếm giải pháp cho các bài toán khó, làm sao có được một kết quả tối ưu nhất, có rất nhiều phương pháp, kỹ thuật được ứng dụng thì một trong số những phương pháp, kỹ thuật được biết đến là giải thuật di truyền kỹ thuật này được dựa trên nguyên lý sinh học tiến hóa để tìm ra một kết quả tối ưu nhất và khi tìm hiểu sơ bộ về giải thuật di truyền thì chúng em thấy được sự hứng thú nhất định về giải thuật này, thấy được những ứng dụng hữu ích mà kỹ thuật này mang lại trong việc đóng góp sự phát triển các giải thuật để tìm ra lời giải cho các bài toán nan giải.

Và trong lần đầu tiên thực hiện đồ án 1 này, chúng em sẽ cùng nhau nghiên cứu, tìm hiểu, thực hành ứng dụng một giải thuật tối ưu kết quả cho bài toán đó là giải thuật di truyền.

1. **Các nghiên cứu liên quan**

Từ khi giải thuật di truyền xuất hiện đã có rất nhiều công trình nghiên cứu về giải thuật này có thể kể đến như:

* *Adaptation in Natural and Artificial Systems (1975) – John Holland*
* *Genetic Algorithms in Search, Optimization, and Machine Learning (1989) – David E. Goldberg*
* *Handbook of Genetic Algorithms (1991) – Lawrence Davis*
* *Genetic Algorithms + Data Structures = Evolution Program (1987) – Zbigniew Michalewicz*
* *Evolutionary Algorithms for Solving Multi – Objective Problems (1994) – Kalyanmoy Deb*

Còn rất nhiên công trình nghiên cứu khác chưa được nhắc đến, nhưng một trong những công trình nghiên cứu quan trọng nhất, đóng góp phần vào sự phát triển giải thuật này đó là cuốn sách *Adaptation in Natural and Artificial Systems* của *John Holland*, cuốn sách đã nêu các khái niệm cơ bản về giải thuật di truyền, John Holland đã trình bày các hệ thống mô hình hóa, khái quát về quá trình tiến hóa và một số ứng dụng thực tế, nghiên cứu này là nguồn cảm hứng rất lớn, tạo nền tảng cho các nghiên cứu sau này.

1. **Mục tiêu và phạm vi nghiên cứu**
2. **Mục tiêu**

Mục tiêu của việc tìm hiểu, nghiên cứu về giải thuật di truyền được xác định:

* Nắm vững các cơ sở lý thuyết về di truyền học và giải thuật di truyền
* Trình bày được các yếu tố trong giải thuật di truyền
* Mô hình hóa giải thuật di truyền
* Ứng dụng được giải thuật vào demo “*Tìm kiếm chuỗi mục tiêu được nhập ngẫu nhiên*”

1. **Phạm vi**

Trong đồ án 1 “Tìm hiểu về GA và ứng dụng” phạm vi nghiên cứu được xác định:

* Tập trung tìm hiểu vào cơ sở lý thuyết của giải thuật di truyền
* Nghiên cứu quá trình các bước thực hiện của giải thuật di truyền
* Tìm hiểu các phương pháp thực hiện và ứng dụng phương pháp vào demo

1. **Đối tượng nghiên cứu**

Đối tượng nghiên cứu được xác định trong đề tài này là:

* Những lĩnh vực liên quan đế đề tài
* Thuyết di truyền học
* Lý thuyết về giải thuật di truyền
* Các phương pháp: Fitness, Selection, Crossover, Mutation

1. **Phương pháp nghiên cứu**

Các phương pháp sẽ được áp dụng trong quá trình thực hiện đồ án 1 này:

* Phân tích các phương pháp thực hiện của giải thuật di truyền
* Thu thập các dữ liệu, thông tin về đề tài từ các nguồn chính thống
* Đánh giá kết quả của demo, từ đó nắm được cấu trúc hoạt động của demo trong đồ án này

1. **Bố cục của đồ** **án**

Trong đề tài giải thuật di truyền của đồ án 1 này chúng em sẽ trình bày bố cục của nội dung như sau:

* **Chương 1: GIỚI THIỆU**
* **Chương 2: CƠ SỞ LÝ THUYẾT**
* **Chương 3: PHƯƠNG PHÁP THỰC HIỆN**
* **Chương 4: KẾT QUẢ THỰC HIỆN VÀ ĐÁNG GIÁ CÁC PHƯƠNG PHÁP**
* **Chương 5: KẾT LUẬN VÀ HƯỚNG PHÁP TRIỂN**

**Chương 1**: Giời thiệu tổng quan, xác định phạm vi và đối tượng nghiên cứu, các phương nghiên cứu.

**Chương 2**: Trình bày các niệm liên quan đến giải thuật di truyền, khái niệm về giải thuật di truyền và lịch sử, lý thuyết về các phương pháp trong giải thuật di truyền.

**Chương 3**: Ứng dụng các phương pháp trong giải thuật di truyền vào demo “*Tìm kiếm chuỗi mục tiêu được nhập ngẫu nhiên*”.

**Chương 4**: Trình bày kết quả thực hiện và đánh giá hiệu quả của các phương pháp đã áp dụng.

**Chương 5**: Kết luận và đưa ra hương phát triển cho giải thuật di truyền trong tương lai.

# **Chương 2**

**CƠ SỞ LÝ THUYẾT**

1. **Khoa học máy tính**
2. **Khái niệm khoa học máy tính**

Khoa Học Máy Tính là chuyên ngành nghiên cứu về các cơ sở lý thuyết về các dữ liệu thông tin và tính toán, cùng với sự thực hiện và ứng dụng của chuyên ngành này trong các hệ thống máy tính. Khoa Học Máy Tính là cách tiếp cận khoa học và thực tiễn nhất để có thể tính toán nghiên cứu tìm ra các giải pháp thích hợp, tối ưu nhất cho các vấn đề nan giải, việc thua thập dữ liệu, phân tích, xử lý, lưu trữ, truyền thông và truy cập dữ liệu tất cả đều là một phần trong Khoa Học Máy Tính.

Một định nghĩa khác về chuyên ngành này, Khoa Học Máy Tính nghiên cứu sâu về các quy trình, cách thức để thực hiện của các giải thuật từ đó trên nền tảng của các giải thuật đã nghiên cứu, kế tiếp là áp dụng vào các bài toán thực tế chưa có lời giải tối ưu nhất.

Có rất nhiều bài toán tối ưu chưa thật sự có một giải thuật tối ưu thõa đáng, như thế dẫn đến việc một chương trình, hệ thống giảm đi tính hiệu quả, tốn nhiều tài nguyên và thời gian. Công việc của một kỹ sư Khoa Học Máy Tính là nghiên cứu, thấu hiểu vấn đề, chọn lọc các giải pháp, tích hợp các dữ liệu đã được thu thập, từ đó đưa ra các giải thuật tối ưu thõa đáng cho một bài toán.

1. **Lịch sử phát triển khoa học máy tính**

Lịch sử của khoa học máy tính đã bắt đầu từ nhiều năm trước khi các kỹ sư phát minh ra máy tính hiện đại. Những máy tính dành cho việc tính toán các bài toán số học đã tồn tại từ thời cổ đại, chẳng hạn như bàn tính. Hơn nữa, các giải thuật để thực hiện tính toán đã có kể từ thời cổ đại, ngay cả trước khi phát triển các thiết bị tính toán phức tạp.

Wilhelm Schickard đã thiết kế và hoàn thành chiếc máy tính cơ học đầu tiên năm 1623. Đến năm 1673, *Gottfried Leibniz* trình diễn một máy tính cơ học số, được gọi là *Stepped Reckoner*. Ông có thể được coi là nhà Khoa Học Máy Tính đầu tiên và nhà lý thuyết thông tin mà đã ghi lại hệ thống số nhị phân.

Blaise Pascal thiết kế và xây dựng máy tính cơ học hoạt động được mang tên Pascaline. Năm 1642 *Charles Babbage* đã thiết kế một máy tính theo hiệu (*Difference Engine*) vào thời *Victoria* và *Ada Lovelace* đã viết bản hướng dẫn sử dụng máy. Nhờ công trình này, ngày nay bà được biết đến là lập trình viên đầu tiên trên thế giới. Vào khoảng năm 1900, tập đoàn *IBM* đã bán những chiếc máy tính dùng thẻ đục lỗ. Tuy nhiên, tất cả những chiếc máy này đều chỉ thực hiện một nhiệm vụ đơn hoặc cùng lắm là một tập nhỏ các nhiệm vụ.

Trước năm 1920, công việc tính toán được thực hiện chủ yếu bởi những nhân viên chuyên nghiệp. Những nhà nghiên cứu đầu tiên về ngành mà sau này được gọi là khoa học máy tính, chẳng hạn như *Kurt Gödel*, *Alonzo Church* và *Alan Turing*, đã quan tâm đến câu hỏi về khả năng tính toán: “những gì có thể được tính toán bởi một người thủ quỹ người chỉ đơn giản dùng giấy và bút chì để làm một danh sách các bước tính toán, cho đến khi nào xong việc mà không cần đến trí thông minh hay hiểu biết?”. Một phần của động cơ này là ước muốn phát triển các máy tính có khả năng tự động hóa các công việc tính toán thường là buồn tẻ và dễ sai của một người tính toán. Vấn đề then chốt là xây dựng các hệ thống tính toán phổ dụng có khả năng thực hiện mọi nhiệm vụ tính toán có thể cần đến và nhờ đó tổng quát hóa tất cả các máy tính chuyên biệt trước kia thành một khái niệm đơn nhất về chiếc máy tính phổ dụng.

Trong những năm 1940, khi các máy tính mới hơn và mạnh hơn được phát triển, người ta thấy rõ ràng hơn rằng máy tính có thể được sử dụng cho nhiều mục đích khác ngoài các tính toán toán học, lĩnh vực khoa học máy tính được mở rộng thành ngành nghiên cứu tính toán nói chung. Từ thập kỷ 60, khoa học máy tính bắt đầu được thiết lập như là một ngành học riêng biệt, với sự ra đời của các khoa khoa học máy tính đầu tiên và các chương trình đào tạo đại học chuyên ngành khoa học máy tính. Từ khi các máy tính được sử dụng trong thực tiễn, nhiều ứng dụng của tính toán đã trở thành các lĩnh vực nghiên cứu riêng biệt.

1. **Giải thuật**
2. **Khái niệm giải thuật và lịch sử**

Giải thuật (Algorithms) là một tập hợp hữu hạn các hướng dẫn được xác định rõ ràng, có thể thực hiện được bằng máy tính, thường để giải quyết một lớp vấn đề hoặc để thực hiện một phép tính. Các giải thuật luôn rõ ràng và được sử dụng chỉ rõ việc thực hiện các phép tính, xử lý dữ liệu, suy luận tự động và các tác vụ khác. Nói cách khác, giải thuật cung cấp một bộ các quy tắc hay quy trình giải quyết một vấn đề nào đó sau một số bước hữu hạn xuất phát từ một tập hợp các dữ kiện cho trước. Trạng thái xuất phát (dữ liệu đầu vào) của giải thuật thường được kí hiệu là INPUT, trạng thái đạt được (kết quả đầu ra) là OUTPUT.

Từ tiếng Anh của giải thuật là Algorithm (là tên gọi Latin hóa của Al-Khwarizmi nhà toán học vùng Vịnh Ba Tư nửa đầu thế kỉ thứ IX) ban đầu được sử dụng để chỉ các bộ quy tắc và kỹ thuật được Al-Khwarizmi sử dụng để giải các phương trình đại số, trước khi được khái quát hóa để chỉ bất kỳ bộ quy tắc hoặc kỹ thuật nào.

Sự phát triển của lý thuyết giải thuật bắt đầu bằng chứng minh của Kurt Gödel các định lý về tính không đầy đủ của các hệ thống tiên đề (định lý đầu tiên công bố vào năm 1931) đòi hỏi cần phải chuẩn hóa khái niệm giải thuật (Định lý bất toàn Gödel). Các phiên bản chuẩn hóa đầu tiên của khái niệm này được phát triển vào những năm 1930 bởi Alan Turing, Emil Post và Alonzo Church. Sau đó, lý thuyết giải thuật đã nhận được sự phát triển hơn nữa trong các công trình của Stephen Cole Kleene, Andrey Andreyevich Markov và những người khác. Vào năm 1954 Andrey Andreyevich Markov đề xuất tinh chỉnh khái niệm giải thuật bằng phương tiện mà ông gọi là giải thuật bình thường, theo đánh giá được coi là “một trong những phiên bản tiêu chuẩn hóa thành công nhất của giải thuật”. Cách tiếp cận chung nhất khái niệm giải thuật được đề xuất bởi Andrey Nikolaevich Kolmogorov vào năm 1965. Trong những năm tiếp theo, Donald Knuth, Alfred Aho và Jeffrey Ullman đã đóng góp đáng kể cho lý thuyết giải thuật.

Giải thuật có thể được mô tả như một thủ tục hoặc công thức để giải quyết vấn đề. Chính vì vậy giải thuật được sử dụng rộng rãi trong các lĩnh vực khác nhau, đặc biệt trong toán học và khoa học máy tính, cũng như trong cuộc sống hàng ngày.

Riêng trong lĩnh vực toán học, ngoài các ứng dụng cụ thể trong đại số, giải tích, lý thuyết số, khái niệm giải thuật là cơ sở cho một trong những khái niệm trung tâm của logic toán học, cụ thể là khái niệm tính toán. Đó là lý do tại sao, định lý Gödel về tính không đầy đủ của các hệ thống tiên đề có thể được coi là hệ quả của các định lý của lý thuyết giải thuật. Lý thuyết giải thuật cũng có mối liên kết chặt chẽ với chính nền tảng của toán học, trong đó mối quan hệ giữa tính kiến thiết (Constructive) và không kiến thiết (Non-Constructive) chiếm một vị trí trung tâm. Có thể nói, lý thuyết giải thuật cung cấp các công cụ cần thiết để phát triển xu hướng mang tính kiến thiết trong toán học.

Trong Khoa Học Máy Tính, năm 1965, Andrey Nikolaevich Kolmogorov đã đề xuất sử dụng lý thuyết giải thuật làm cơ sở cho lý thuyết thông tin. Lý thuyết giải thuật cũng là nền tảng lý thuyết cho một số vấn đề của toán học tính toán và có liên quan chặt chẽ với điều khiển học, trong đó việc nghiên cứu các giải thuật điều khiển đóng vai trò quan trọng. Cuối cùng, cần phải nói, giải thuật là bước không thể thiếu trong quá trình giải quyết bất kì bài toán nào trên máy tính, do vậy khái niệm giải thuật là một trong các khái niệm cơ bản cần biết đối với bất kì ai muốn lập trình cho máy tính.

Khi một giải thuật đã hình thành, ta không xét đến việc chứng minh tính đầy đủ và tính đúng đắn của giải thuật đó mà chỉ chú trọng đến việc áp dụng các bước theo đúng trình tự quy định để thu được kết quả cần thiết. Việc chứng minh ấy phải được tiến hành trong quá trình thiết kế giải thuật (Thiết kế giải thuật). Nói cách khác, giải thuật chỉ là việc áp dụng các công thức hay quy tắc, quy trình đã được công nhận là đúng hay đã được chứng minh về mặt toán học.

1. **Các giải thuật phổ biến**
2. **Giải thuật tìm kiếm**

Giải thuật tìm kiếm là quy trình từng bước để tìm một phần tử cụ thể trong cấu trúc dữ liệu. Các giải thuật tìm kiếm được sử dụng trong nhiều ứng dụng, bao gồm tìm kiếm trên web, hệ thống cơ sở dữ liệu và trí tuệ nhân tạo. Có nhiều loại giải thuật tìm kiếm khác nhau, mỗi loại đều có điểm mạnh và điểm yếu riêng. Các giải thuật tìm kiếm phổ biến nhất bao gồm:

* **Linear search**: Giải thuật này chỉ đơn giản kiểm tra từng phần tử trong cấu trúc dữ liệu theo thứ tự cho đến khi tìm thấy phần tử đích. Tìm kiếm tuyến tính dễ thực hiện nhưng cũng là giải thuật tìm kiếm kém hiệu quả nhất, đặc biệt đối với các cấu trúc dữ liệu lớn.
* **Binary search**: Giải thuật này chỉ hoạt động trên các cấu trúc dữ liệu được sắp xếp. Giải thuật hoạt động bằng cách chia đôi cấu trúc dữ liệu và so sánh phần tử đích với phần tử ở giữa. Nếu phần tử đích nhỏ hơn phần tử ở giữa thì việc tìm kiếm sẽ tiếp tục ở nửa dưới của cấu trúc dữ liệu. Nếu không, việc tìm kiếm sẽ tiếp tục ở nửa trên của cấu trúc dữ liệu. Tìm kiếm nhị phân hiệu quả hơn nhiều so với tìm kiếm tuyến tính, đặc biệt đối với các cấu trúc dữ liệu lớn.
* **Hashing:** Giải thuật này hoạt động bằng cách tạo bảng băm, đây là cấu trúc dữ liệu ánh xạ các khóa tới các giá trị. Để tìm kiếm một phần tử, giải thuật băm chuyển phần tử đó thành một khóa và sau đó tra cứu khóa đó trong bảng băm. Nếu tìm thấy khóa thì giá trị tương ứng sẽ được trả về. Ngược lại, phần tử không được tìm thấy. Băm là giải thuật tìm kiếm hiệu quả nhất cho cấu trúc dữ liệu lớn, nhưng đòi hỏi nhiều bộ nhớ hơn các giải thuật tìm kiếm khác.

Giải thuật tìm kiếm tốt nhất để sử dụng tùy thuộc vào ứng dụng cụ thể. Ví dụ: nếu cấu trúc dữ liệu nhỏ và chưa được sắp xếp thì tìm kiếm tuyến tính có thể là giải pháp đơn giản và hiệu quả nhất. Nếu cấu trúc dữ liệu lớn và được sắp xếp thì tìm kiếm nhị phân là giải pháp hiệu quả nhất. Nếu cấu trúc dữ liệu lớn và chưa được sắp xếp thì băm là giải pháp hiệu quả nhất.

1. **Giải thuật sắp xếp**

Giải thuật sắp xếp là giải thuật sắp xếp một tập hợp các mục theo một thứ tự cụ thể. Thứ tự phổ biến nhất là thứ tự số tăng dần hoặc giảm dần, nhưng giải thuật sắp xếp cũng có thể được sử dụng để sắp xếp các loại dữ liệu khác, chẳng hạn như chuỗi hoặc đối tượng. Các giải thuật sắp xếp được sử dụng trong rất nhiều ứng dụng, bao gồm:

* + **Database systems**: Các giải thuật sắp xếp được sử dụng để sắp xếp các bản ghi cơ sở dữ liệu nhằm giúp hệ thống dễ dàng tìm kiếm và truy xuất hơn.
  + **Website search**: Giải thuật sắp xếp được sử dụng để sắp xếp kết quả tìm kiếm nhằm cung cấp kết quả phù hợp nhất cho người dùng.
  + **Machine learning**: Các giải thuật sắp xếp được sử dụng để sắp xếp dữ liệu huấn luyện nhằm huấn luyện các mô hình học máy.
  + **Data visualization**: Các giải thuật sắp xếp được sử dụng để sắp xếp dữ liệu trước khi được hiển thị trực quan để dễ hiểu hơn.

Có nhiều loại giải thuật sắp xếp khác nhau, mỗi loại đều có điểm mạnh và điểm yếu riêng. Các giải thuật sắp xếp phổ biến nhất bao gồm:

* + **Selection sort**: Giải thuật này hoạt động bằng cách tìm phần tử nhỏ nhất trong danh sách và hoán đổi với phần tử đầu tiên. Sau đó, tìm phần tử nhỏ thứ hai trong danh sách và hoán đổi với phần tử thứ hai, v.v. Sắp xếp lựa chọn rất đơn giản để thực hiện, nhưng cũng là giải thuật sắp xếp kém hiệu quả nhất.
  + **Insertion sort**: Hoạt động bằng cách chèn từng phần tử vào danh sách đã sắp xếp vào đúng vị trí. Sắp xếp chèn hiệu quả hơn sắp xếp lựa chọn, nhưng vẫn không hiệu quả lắm đối với các danh sách lớn.
  + **Merge sort**: Giải thuật này hoạt động bằng cách chia danh sách làm đôi, sắp xếp đệ quy từng nửa, sau đó hợp nhất hai nửa đã sắp xếp lại với nhau. Sắp xếp hợp nhất là một giải thuật sắp xếp rất hiệu quả nhưng yêu cầu bộ nhớ bổ sung để lưu trữ hai nửa được sắp xếp của danh sách.
  + **Quick sort**: Hoạt động bằng cách phân vùng danh sách xung quanh một phần tử trụ. Các phần tử nhỏ hơn phần tử trục được đặt ở một bên của trục và các phần tử lớn hơn phần tử trục được đặt ở phía bên kia của trục. Giải thuật quicksort sau đó được áp dụng đệ quy cho mỗi bên của trục xoay. Quicksort là giải thuật sắp xếp hiệu quả nhất cho hầu hết các đầu vào, nhưng có thể có độ phức tạp về thời gian trong trường hợp xấu nhất là O(n^2).

Việc lựa chọn giải thuật sắp xếp phụ thuộc vào một số yếu tố, bao gồm kích thước của danh sách, loại dữ liệu được sắp xếp và các đặc tính hiệu suất mong muốn. Giải thuật sắp xếp tốt nhất phụ thuộc vào ứng dụng cụ thể. Ví dụ: nếu danh sách nhỏ và chưa được sắp xếp thì sắp xếp lựa chọn hoặc sắp xếp chèn có thể là giải pháp đơn giản và hiệu quả nhất. Nếu danh sách lớn và được sắp xếp thì sắp xếp hợp nhất hoặc sắp xếp nhanh là giải pháp hiệu quả nhất. Nếu danh sách lớn và chưa được sắp xếp thì sắp xếp nhanh là giải pháp hiệu quả nhất, nhưng điều quan trọng là phải nhận thức được độ phức tạp về thời gian trong trường hợp xấu nhất. Ngoài ra còn có một số giải thuật sắp xếp khác chuyên biệt hơn cho một số loại dữ liệu hoặc ứng dụng nhất định. Ví dụ: sắp xếp cơ số là giải thuật sắp xếp rất hiệu quả cho số nguyên và sắp xếp đếm là giải thuật sắp xếp rất hiệu quả cho các danh sách nhỏ có phạm vi giá trị giới hạn. Nói chung, tốt nhất nên chọn giải thuật sắp xếp đã được nghiên cứu và triển khai kỹ lưỡng trong một thư viện tốt. Điều này sẽ giúp đảm bảo rằng giải thuật hiệu quả và chính xác.

1. **Các đặc trưng của giải thuật**

Giải thuật sắp xếp là giải thuật sắp xếp một tập hợp các mục theo một thứ tự cụ thể. Thứ tự phổ biến nhất là thứ tự số tăng dần hoặc giảm dần, nhưng giải thuật sắp xếp cũng có thể được sử dụng để sắp xếp các loại dữ liệu khác, chẳng hạn như chuỗi hoặc đối tượng. Các giải thuật sắp xếp được sử dụng trong rất nhiều ứng dụng, bao gồm:

* + **Database systems**: Các giải thuật sắp xếp được sử dụng để sắp xếp các bản ghi cơ sở dữ liệu nhằm giúp hệ thống dễ dàng tìm kiếm và truy xuất hơn.
  + **Website search**: Giải thuật sắp xếp được sử dụng để sắp xếp kết quả tìm kiếm nhằm cung cấp kết quả phù hợp nhất cho người dùng.
  + **Machine learning**: Các giải thuật sắp xếp được sử dụng để sắp xếp dữ liệu huấn luyện nhằm huấn luyện các mô hình học máy.
  + **Data visualization**: Các giải thuật sắp xếp được sử dụng để sắp xếp dữ liệu trước khi được hiển thị trực quan để dễ hiểu hơn.

Có nhiều loại giải thuật sắp xếp khác nhau, mỗi loại đều có điểm mạnh và điểm yếu riêng. Các giải thuật sắp xếp phổ biến nhất bao gồm:

* + **Selection sort**: Giải thuật này hoạt động bằng cách tìm phần tử nhỏ nhất trong danh sách và hoán đổi với phần tử đầu tiên. Sau đó, tìm phần tử nhỏ thứ hai trong danh sách và hoán đổi với phần tử thứ hai, v.v. Sắp xếp lựa chọn rất đơn giản để thực hiện, nhưng cũng là giải thuật sắp xếp kém hiệu quả nhất.
  + **Insertion sort**: Hoạt động bằng cách chèn từng phần tử vào danh sách đã sắp xếp vào đúng vị trí. Sắp xếp chèn hiệu quả hơn sắp xếp lựa chọn, nhưng vẫn không hiệu quả lắm đối với các danh sách lớn.
  + **Merge sort**: Giải thuật này hoạt động bằng cách chia danh sách làm đôi, sắp xếp đệ quy từng nửa, sau đó hợp nhất hai nửa đã sắp xếp lại với nhau. Sắp xếp hợp nhất là một giải thuật sắp xếp rất hiệu quả nhưng yêu cầu bộ nhớ bổ sung để lưu trữ hai nửa được sắp xếp của danh sách.
  + **Quick sort**: Hoạt động bằng cách phân vùng danh sách xung quanh một phần tử trụ. Các phần tử nhỏ hơn phần tử trục được đặt ở một bên của trục và các phần tử lớn hơn phần tử trục được đặt ở phía bên kia của trục. Giải thuật quicksort sau đó được áp dụng đệ quy cho mỗi bên của trục xoay. Quicksort là giải thuật sắp xếp hiệu quả nhất cho hầu hết các đầu vào, nhưng có thể có độ phức tạp về thời gian trong trường hợp xấu nhất là O(n^2).

Việc lựa chọn giải thuật sắp xếp phụ thuộc vào một số yếu tố, bao gồm kích thước của danh sách, loại dữ liệu được sắp xếp và các đặc tính hiệu suất mong muốn. Giải thuật sắp xếp tốt nhất phụ thuộc vào ứng dụng cụ thể. Ví dụ: nếu danh sách nhỏ và chưa được sắp xếp thì sắp xếp lựa chọn hoặc sắp xếp chèn có thể là giải pháp đơn giản và hiệu quả nhất. Nếu danh sách lớn và được sắp xếp thì sắp xếp hợp nhất hoặc sắp xếp nhanh là giải pháp hiệu quả nhất. Nếu danh sách lớn và chưa được sắp xếp thì sắp xếp nhanh là giải pháp hiệu quả nhất, nhưng điều quan trọng là phải nhận thức được độ phức tạp về thời gian trong trường hợp xấu nhất. Ngoài ra còn có một số giải thuật sắp xếp khác chuyên biệt hơn cho một số loại dữ liệu hoặc ứng dụng nhất định. Ví dụ: sắp xếp cơ số là giải thuật sắp xếp rất hiệu quả cho số nguyên và sắp xếp đếm là giải thuật sắp xếp rất hiệu quả cho các danh sách nhỏ có phạm vi giá trị giới hạn. Nói chung, tốt nhất nên chọn giải thuật sắp xếp đã được nghiên cứu và triển khai kỹ lưỡng trong một thư viện tốt. Điều này sẽ giúp đảm bảo rằng giải thuật hiệu quả và chính xác.

1. **Ứng dụng giải thuật trong khoa học máy tính**

Lĩnh vực Công Nghệ Thông Tin nói chung và Khoa Học Máy Tính nói riêng đối với sự phát triển đa ngành nghề hiện nay thì không thể thiếu ứng dụng thực tiễn của hai lĩnh vực này. Lấy một ví dụ “Một doanh nghiệp cần đến một hệ thống phân tích hành vi tiêu dùng của các khách hàng của họ thông qua việc những dữ liệu đã được thu thập được, dựa trên sự phân tích của hệ thống mà doanh nghiệp sẽ đưa ra các chương trình khuyên mãi, các chiến lược dài hạn trong tương lai.”, tùy theo tính hiệu quả của các giải thuật đã áp dụng trong hệ thống phân tích đó, mà có thể ảnh hưởng đến hiệu năng phân tích. Một giải thuật đủ tối ưu thì sẽ cho ra kết quả gần đúng nhất với thực tế từ đó tạo một gốc nhìn khách quan hơn cho các tổ chức doanh nghiệp, để đưa ra các lựa chọn đúng đắn. Giải thuật được sử dụng trong mọi lĩnh vực khoa học máy tính, từ hệ điều hành, mạng cho đến trí tuệ nhân tạo. Dưới đây là một vài ví dụ:

* + **Operating systems**: Giải thuật được sử dụng để quản lý CPU, bộ nhớ và các tài nguyên khác của hệ thống máy tính. Ví dụ: giải thuật lập lịch xác định tiến trình nào sẽ chạy tiếp theo trên CPU.
  + **Networking**: Giải thuật được sử dụng để định tuyến các gói dữ liệu giữa các máy tính trên internet. Ví dụ: giải thuật Dijkstra có thể được sử dụng để tìm đường đi ngắn nhất giữa hai nút trong mạng.
  + **Databases**: Giải thuật được sử dụng để lưu trữ và truy xuất dữ liệu hiệu quả từ cơ sở dữ liệu. Ví dụ: giải thuật cây B là cấu trúc dữ liệu có thể được sử dụng để tìm kiếm nhanh dữ liệu trong cơ sở dữ liệu đã được sắp xếp.
  + **Compilers**: Giải thuật được sử dụng để biên dịch các chương trình máy tính từ ngôn ngữ cấp cao (chẳng hạn như C hoặc Java) sang ngôn ngữ cấp thấp (chẳng hạn như mã máy) mà máy tính có thể hiểu được.
  + **Graphics**: Giải thuật được sử dụng để tạo và hiển thị hình ảnh và hoạt ảnh. Ví dụ: giải thuật dò tia có thể được sử dụng để tạo ra hình ảnh chân thực bằng cách mô phỏng cách ánh sáng truyền qua một cảnh.
  + **Machine Learning**: Các giải thuật được sử dụng để huấn luyện các mô hình học máy học từ dữ liệu và đưa ra dự đoán. Ví dụ: giải thuật máy vectơ hỗ trợ có thể được sử dụng để phân loại các điểm dữ liệu thành các danh mục khác nhau.

Các giải thuật rất cần thiết để giải quyết các vấn đề và nhiệm vụ có độ phức tạp , nâng cao hiệu quả và hiệu suất cũng như hỗ trợ các công nghệ và ứng dụng mới. Một số ví dụ cụ thể về các giải thuật được sử dụng ngày nay: Công cụ tìm kiếm Google sử dụng giải thuật phức tạp để xếp hạng các trang web và trả về kết quả phù hợp nhất cho người dùng. Hệ thống đề xuất Netflix sử dụng giải thuật để đề xuất phim và chương trình truyền hình cho người dùng dựa trên lịch sử xem của họ. Giỏ hàng Amazon sử dụng giải thuật để giới thiệu sản phẩm cho người dùng dựa trên những lần mua hàng trước đây của họ. Xe tự lái sử dụng nhiều giải thuật khác nhau để điều hướng đường đi và tránh chướng ngại vật. Bộ lọc thư rác trong ứng dụng email của bạn sử dụng giải thuật để xác định và chặn email spam.

Các giải thuật hiện nay đã có ở khắp mọi nơi trong thế giới hiện đại và rất cần thiết cho nhiều công nghệ mới, nâng cao chất lượng công việc, đời sống, sức khỏe, giáo dục,...cho con người.

1. **Di truyền**
2. **Khái niệm di truyền**

Di truyền là hiện tượng truyền đạt các đặc tính, tính cách, ngoại hình,... của bố mẹ, ông bà tổ tiên cho các thế hệ con, cháu. Lấy ví dụ người bố và đứa con có đôi tai rất giống nhau, có đôi mắt xanh dương giống nhau, có chiếc mũi cũng rất giống nhau,... thì được nhận định nôm na rằng "Bố đã di truyền những đặc điểm này cho con mình", hoặc "Đứa con đã được di truyền những đặc điểm của bố".

Trong sinh học và di truyền học, di truyền là quá trình di chuyển những đặc trưng sinh học từ một cá thể cha mẹ đến cá thể con cái và đồng nghĩa với việc di chuyển các gen, gen thừa nhận mang thông tin sinh học (hay thông tin di truyền). Ngoài ra, các đặc điểm về tính cách, nhận thức và tư duy của con cái có thể được tiếp nhận từ cha mẹ thông qua môi trường sinh hoạt gia đình (các thói quen, quy định của gia đình gọi là gia phong, nề nếp). Ở con người, xác định đặc trưng nào phụ thuộc vào di truyền và đặc trưng nào phụ thuộc vào môi trường thường gây tranh cãi; đặc biệt là đối với những đặc tính phức tạp như trí thông minh và màu da; giữa tự nhiên và nuôi dưỡng.

Di truyền, là tổng hợp của tất cả các quá trình sinh học mà qua đó các đặc điểm cụ thể được truyền từ cá thể cha mẹ sang cá thể con cái của họ. Khái niệm di truyền bao gồm hai quan sát dường như nghịch lý về các sinh vật: đầu tiên là về sự bất biến của một loài từ thế hệ này sang thế hệ khác và kế tiếp về sự khác biệt giữa các cá thể trong một loài. Như đã trở nên rõ ràng trong nghiên cứu về di truyền học. Cả hai khía cạnh của di truyền có thể được giải thích bằng gen, các đơn vị chức năng của vật liệu di truyền được tìm thấy trong tất cả các tế bào sống. Mỗi thành viên của một loài có một bộ gen đặc trưng cho loài đó. Chính bộ gen này đã cung cấp cấu trúc của loài. Tuy nhiên, trong một số các cá thể trong một loài, các biến thể có thể xảy ra ở dạng mỗi gen, tạo cơ sở di truyền cho thực tế là không có hai cá thể nào có những đặc điểm giống hệt nhau.

Việc truyền các đặc tính di truyền từ bố mẹ sang con cái phụ thuộc vào sự phân ly và tái tổ hợp của các gen trong quá trình phân bào và thụ tinh và việc này dẫn đến sự hình thành một số cá thể mới tương tự các loài khác, nhưng biểu hiện một số biến thể nhất định do sự kết hợp của các gen cụ thể và sự tương tác của chúng với môi trường. Và các sinh vật, động vật cứ thế tiếp tục quá trình di truyền qua nhiều thế hệ sau.

1. **Lịch sử di truyền**

Từ thời xa xưa, người ta đã nhận thức được rằng mọi sinh vật trên thế giới từ thực vật đến động vật đều thể hiện ít nhiều những đặc tính từ cha và mẹ; và khác biệt giữa con và cha mẹ cũng thường được truyền lại cho thế hệ cháu. Trên cơ sở đó, con người dò đoán và tìm được phương pháp tuyển chọn tự nhiên và phát triển giống tốt cho gia súc cũng như rau cải, lúa gạo, v.v.... Hiện tượng di truyền được con người chấp nhận như chuyện hiển nhiên cho đến cuối thế kỷ XIX mới có giải thích khoa học hơn. Nhóm nghiên cứu di truyền theo chủ thuyết của *Jean-Baptiste Lamarck* cho rằng ngoài di truyền của những biểu hiện bẩm sinh còn có di truyền của những biểu hiện mắc phải bởi môi trường.

*Charles Darwin* đưa lên giả thuyết về tiến hoá vào năm 1859 nhưng gặp một số khúc mắc, khó khăn nhất là giải thích phương thức của di truyền. *Charles Darwin* cho rằng có pha trộn giữa di truyền bẩm sinh và di truyền của những biểu hiện gây nên bởi môi trường. Nhưng nếu thật sự có sự pha trộn này, chỉ sau một vài thế hệ sẽ nảy sinh ra hiện tượng đồng dạng của toàn chủng và sẽ không có đủ biến dị để sự tuyển chọn tự nhiên có thể xảy ra. Do đó, *Charles* *Darwin* phải tiếp thu phần nào giả thuyết của *Jean-Baptiste Lamarck* vào công trình nghiên cứu của mình. Cách trình bày của *Charles Darwin* về di truyền là cho thấy xảy ra như thế nào và người ta có thể dự đoán hướng di truyền trong thế hệ tới (ví dụ những đặc tính được di truyền nhưng không biểu hiện ở cha hay mẹ vào lúc thụ thai, nhiều đặc tính được di truyền phân biệt theo giống đực hay cái). Với phương thức của di truyền này thì *Charles Darwin* không giải thích được.

Khái niệm di truyền của *Charles Darwin* được người anh em bà con của ông *Francis Galton* cải tiến sau đó, từ đó tạo cơ sở cho nền tảng khoa học di truyền. Tuy nhiên *Francis Galton* không chấp nhận thuyết tiến hóa toàn diện của *Charles Darwin* đó là về thuyết *Pangenesis*: là một cơ chế giả thuyết về di truyền được đề xuất bởi *Charles Darwin* trong cuốn sách *The Variation of Animals and Plants under* *Domestication* (Sự biến đổi của động vật và thực vật được thuần hóa) năm 1868. *Charles Darwin* cho rằng mỗi bộ phận của cơ thể phát ra các hạt nhỏ gọi là *Gemmules*, chúng lưu thông khắp cơ thể và cuối cùng tập trung lại ở các cơ quan sinh sản. Những hạt *Gemmules* này sau đó được truyền lại cho con cái và định hướng sự phát triển của các bộ phận tương tự trên cơ thể ở thế hệ tiếp theo.

Năm 1880, *August Friedrich Leopold Weismann* cắt đuôi của nhiều thế hệ chuột trong phòng thử nghiệm và cho thấy các con chuột trong thế hệ sau vẫn có đuôi. Từ đó, ông chứng minh rằng không có sự di truyền của các tính trạng mắc phải trong cuộc sống.

1. **Các thành phần di truyền**
2. **Population**

Quần thể trong tự nhiên vô cùng đa dạng, khác về số lượng, về loài. Nhưng thường giữa các quần thể với nhau luôn luôn có điểm chung đó là tính di truyền, để một quần thể có thể tồn tại trong tự nhiên thì di truyền là yếu tố không thể thiếu, một quần thể tồn tại theo năm tháng dĩ nhiên sẽ có sự khác biệt với các quần thể trước đó để có được sự khác biệt đó quá trình di truyền luôn được vận hành liên tục trong môi trường tự nhiên để thúc đẩy sự phát triển của các loài, các quần thể.

A group of lions walking on a path in tall grass

Description automatically generated

*Hình 2.1. Quần thể sư tử hoang dã trong tự nhiên*

Nguyên nhân dẫn đến sự biến mất vĩnh viễn của một số quần thể trong tự nhiên là do khí hậu quá khắc nghiệt, môi trường sống không đủ đáp ứng hoặc quần thể đó không thể thích nghi, các đối thủ săn mồi trong tự nhiên và cả việc thức ăn của quần thể đó khan hiếm, nhưng những điều này cũng là một phần của thế giới tự nhiên.

1. **Natural Selection**

Quá trình chọn lọc tự nhiên là một bước rất quan trọng đối với sự phát triển của các quần thể trong tự nhiên, quá trình này có thể là môi trường xung quanh, khí hậu, thời tiết, các biến cố trong tự nhiên. Chọn lọc tự nhiên sẽ chọn ra các cá thể tốt nhất, có tính thích nghi cao đối với môi trường xung quanh, như vậy đồi hỏi các cá thể trong một quần thể phải có các thế mạnh riêng để có thể tiếp tục tồn tại trong môi trường tự nhiên, quá trình chọn lọc tự nhiên sẽ luôn luôn được tiếp diễn trong toàn thế giới tự nhiên, việc này dẫn đến một số quần thể không thể thích nghi tốt dần dần theo thời gian biến mất khỏi tự nhiên có thể nói là tuyệt chủng, nhưng ngược lại nhờ quá trình này mà có các quần thể được cơ hội tiến hóa, mạnh mẽ hơn, có tính thích nghi cao hơn từ đó sinh ra con cái có các bộ gen tốt và tiếp tục được phát triển bởi các thế hệ sau.

A silhouette of a person in a line

Description automatically generated

*Hình 2.2 Mô phỏng loài người trải qua các giai đoạn tiến hóa*

Hình bên trên cho chúng ta thấy được quá trình tiến hóa của loài người được mô phỏng tóm tắt, nhờ quá trình chọn lọc của tự nhiên mà loài người được cơ hội để phát triển các đặc điểm như thể chất, cảm xúc, tư duy, tin thần..., có thể khẳng định vai trò đóng góp, thúc đẩy sự phát triển của quá trình chọn lọc tự nhiên cho mọi quần thể trong tự nhiên là không thể thiếu cho muôn loài trong tự nhiên.

1. **Mutation**

Đột biến là quá trình thay đổi trình tự của DNA của một cá thể tronng tự nhiên, xảy ra việc đột biến có thể do lỗi trong quá trình di truyền từ cá thể truyền đi sang cá thể thừa hưởng, đột biến xảy ra ở các bộ gen, kể cả một gen đơn lẻ. Đột biến đôi khi không mang đến sự thay đổi cụ thể nào, khó có thể nhìn thấy được các đột biến như dạng này ngược lại có một số loại đột biến lại ảnh hưởng rất lớn đến một cá thể, sự ảnh hưởng này có thể là tích cực hoặc tiêu cực. Một số dạng đột biến thường được biết đến:

* **Point mutation**

Đột biến điểm là một loại đột biến gen xảy ra khi một bazơ nucleotide đơn lẻ bị thay đổi, chèn hoặc xóa khỏi chuỗi DNA. Đột biến điểm có thể xảy ra ở bất kỳ gen nào và có thể gây ra nhiều tác động khác nhau, tùy thuộc vào gen bị ảnh hưởng và mức độ nghiêm trọng của đột biến. Một số đột biến điểm là vô hại, trong khi những đột biến khác có thể gây ra các vấn đề sức khỏe nghiêm trọng. Ví dụ, đột biến điểm ở gen liên quan đến sự phát triển và phân chia tế bào có thể làm tăng nguy cơ ung thư. Đột biến điểm cũng có thể gây ra các rối loạn di truyền như bệnh hồng cầu hình liềm và bệnh xơ nang.

A diagram of a sequence of dna

Description automatically generated

*Hình 2.3 minh họa về Point Mutations*

* **Frame – shift mutations**

Đột biến dịch khung là một loại đột biến gen xảy ra khi một hoặc nhiều nucleotide được chèn vào hoặc xóa khỏi chuỗi DNA. Điều này có thể làm thay đổi khung đọc của DNA, đó là cách trình tự được đọc theo nhóm gồm ba nucleotide được gọi là codon. Mỗi codon mã hóa cho một axit amin cụ thể, do đó đột biến dịch khung có thể phá vỡ trình tự axit amin trong protein, dẫn đến protein không có chức năng hoặc bị biến đổi. Đột biến dịch khung có thể xảy ra một cách tự nhiên hoặc do các yếu tố môi trường gây ra như tiếp xúc với tia cực tím hoặc hóa chất. Chúng cũng là đặc điểm chung của một số loại ung thư, chẳng hạn như ung thư đại trực tràng và ung thư phổi.

Đột biến Frameshift có thể gây ra nhiều tác động khác nhau, tùy thuộc vào gen bị ảnh hưởng và mức độ nghiêm trọng của đột biến. Một số đột biến dịch chuyển khung hình có thể không gây ra triệu chứng nào đáng chú ý, trong khi những đột biến khác có thể dẫn đến các vấn đề sức khỏe nghiêm trọng. Ví dụ, đột biến dịch khung ở các gen liên quan đến sự phát triển và phân chia tế bào có thể làm tăng nguy cơ ung thư. Không có cách chữa trị cho đột biến dịch khung, nhưng có các phương pháp điều trị để kiểm soát các triệu chứng rối loạn di truyền do đột biến dịch khung gây ra. Ví dụ, một số rối loạn di truyền có thể được điều trị bằng liệu pháp thay thế enzyme, liệu pháp gen hoặc các liệu pháp nhắm mục tiêu nhắm vào protein cụ thể bị ảnh hưởng bởi đột biến.

* **Chromosome mutations**

Đột biến nhiễm sắc thể là sự thay đổi về số lượng hoặc cấu trúc của nhiễm sắc thể. Nhiễm sắc thể là những cấu trúc dạng sợi có chứa DNA, vật liệu di truyền của mọi sinh vật sống. Đột biến nhiễm sắc thể có thể xảy ra một cách tự nhiên hoặc do các yếu tố môi trường gây ra như tiếp xúc với bức xạ hoặc hóa chất.

Đột biến nhiễm sắc thể có thể gây ra nhiều tác động khác nhau, tùy thuộc vào loại đột biến và gen bị ảnh hưởng. Một số đột biến nhiễm sắc thể có thể không gây ra triệu chứng đáng chú ý, trong khi những đột biến khác có thể dẫn đến các vấn đề sức khỏe nghiêm trọng, chẳng hạn như chậm phát triển, dị tật bẩm sinh và ung thư.

1. **Evolution**

Tiến hóa là quá trình di truyền các đặc tính của tổ tiên sang các thế hệ con cháu trong quần thể tự nhiên, các đặc tính được di truyền sang các thế hệ sau sẽ có sự khác biệt, thích ứng với môi trường xung quanh, khí hậu,... Những cá thể nhận được các đặc tính di truyền sẽ vẫn giữ được các đặc tính giống với tổ tiên nhưng theo thời gian thông qua quá trình di truyền các đặc tính này sẽ được cá thể thể hiện một cách khác. Nguyên nhân dẫn đến quá trình tiến hóa một phần là do sự đột biến ở một cá thể hoặc cả một quần thể, nhưng đột biến thường khó đoán trước bởi vì không đảm bảo được cá thể đó sẽ nhận được bộ gen tốt hay bộ gen xấu, còn tiến hóa thì hướng đến một cá thể hoặc một quần thể tốt hơn, có thể hiểu quá trình tiến hóa là quá trình đột biến theo hướng tốt.

A bird sitting on a branch

Description automatically generated

*Hình 2.4 Kingfishers loài chim tiến hóa mỏ dài để bắt cá*

1. **Giải thuật di truyền**
2. **Khái niệm giải thuật di truyền**

Giải thuật di truyền là một kỹ thuật trong chuyên ngành khoa học máy tính nhằm tìm kiếm giải pháp thích hợp nhất trong tập hợp các giải pháp cho bài toán tối ưu tổ hợp (*Combinatorial Optimization*). Giải thuật di truyền cũng như các giải thuật tiến hóa nói chung, hình thành dựa trên quan niệm cho rằng, quá trình tiến hóa tự nhiên là quá trình hoàn hảo nhất, hợp lý nhất và chính bản thân đã mang tính tối ưu nhất. Quan niệm này có thể được xem như một tiên đề đúng, không thể chứng minh được, nhưng rất phù hợp với thực tế khách quan. Quá trình tiến hóa thể hiện tính tối ưu ở chỗ, thế hệ sau luôn luôn bao giờ cũng tốt hơn, phát triển hơn, hoàn thiện hơn thế hệ trước. Tiến hóa tự nhiên được duy trì nhờ hai quá trình cơ bản sau: sinh sản và chọn lọc tự nhiên. Xuyên suốt quá trình tiến hóa tự nhiên, các thế hệ mới luôn được sinh ra để bổ sung thay thế các thế hệ cũ. Cá thể nào phát tiển hơn, thích ứng hơn với môi trường sẽ được tồn tại còn cá thể nào không thích ứng được với môi trường sẽ bị đào thải. Sự thay đổi môi trường là động lực thúc đẩy quá trình tiến hóa. Ngược lại, tiến hóa cũng tác động trở lại góp phần làm thay đổi môi trường.

Các cá thể mới sinh ra trong quá trình tiến hóa nhờ sự lai ghép ở các thế hệ trước. Một cá thể mới có thể mang những đặc tính của ông bà, cha mẹ (gọi là di truyền), cũng có một số cá thể mang những tính trạng hoàn toàn mới (gọi là đột biến). Di truyền và đột biến là hai cơ chế có vai trò quan trọng như nhau trong quá trình tiến hóa, dù rằng khả năng đột biến ở một cá thể xảy ra với xác suất nhỏ hơn rất nhiều so với khả năng di truyền. Các giải thuật tiến hóa tuy có những đặc điểm khác nhau, nhưng đều mô phỏng bốn quá trình cơ bản bao gồm: Fitness – Selection – Crossover – Mutation.

1. **Lịch sử giải thuật di truyền**

Lịch sử của giải thuật di truyền có thể bắt nguồn từ công trình của *John Holland* vào những năm 1960. *John Holland* quan tâm đến việc phát triển một mô hình về sự thích ứng xảy ra trong các hệ thống tự nhiên. Ông đề xuất rằng sự thích ứng có thể được giải thích bằng sự tương tác của ba yếu tố chính: biến thể, chọn lọc và kế thừa. Sự biến đổi đề cập đến những thay đổi ngẫu nhiên xảy ra trong gen. Chọn lọc đề cập đến quá trình các cá thể có gen thích nghi tốt hơn có nhiều khả năng sống sót và sinh sản hơn. Di truyền đề cập đến việc truyền gen từ cha mẹ sang con cái. *John Holland* đã chỉ ra rằng ba yếu tố này có thể dẫn đến sự xuất hiện của các quần thể ngày càng thích nghi tốt theo thời gian. Ông đã phát triển một khuôn khổ toán học để mô tả quá trình này mà ông gọi là định lý lược đồ.

Năm 1975, *John Holland* xuất bản cuốn sách mang tên *Adaptation in Natural and Artificial Systems* (*Thích ứng trong các hệ thống tự nhiên và nhân tạo*), trong đó mô tả công trình của ông về giải thuật di truyền. Cuốn sách này được coi là tác phẩm tiêu biểu về lĩnh vực này. Các nhà nghiên cứu khác bắt đầu khám phá việc sử dụng giải thuật di truyền để giải quyết các vấn đề trong thế giới thực vào cuối những năm 1970 và đầu những năm 1980. Một trong những ứng dụng ban đầu đáng chú ý nhất là thiết kế ăng-ten cho tàu vũ trụ *Voyager 1*. Giải thuật di truyền đã trở nên phổ biến rộng rãi hơn vào cuối những năm 1980 và đầu những năm 1990. Điều này một phần là do sự phát triển của máy tính nhanh hơn và sự sẵn có ngày càng tăng của phần mềm để thực hiện các giải thuật di truyền.

Ngày nay, giải thuật di truyền được sử dụng để giải quyết rất nhiều vấn đề trong nhiều lĩnh vực khác nhau, bao gồm kỹ thuật, tài chính và học máy. Dưới đây là dòng thời gian của một số sự kiện quan trọng trong lịch sử giải thuật di truyền:

* **1962**: *John Holland* xuất bản một bài báo về việc sử dụng giải thuật di truyền để giải các bài toán tối ưu hóa.
* **1975**: *John Holland* xuất bản cuốn sách Adaptation in Natural and Artificial Systems, trong đó mô tả chi tiết công trình của ông về giải thuật di truyền.
* **1975**: Giải thuật di truyền được sử dụng để thiết kế ăng-ten cho tàu vũ trụ *Voyager 1*.
* **1985**: *David Goldberg* xuất bản một cuốn sách có tên Giải thuật di truyền trong tìm kiếm, tối ưu hóa và học máy, giúp phổ biến lĩnh vực này.
* **1992**: *John Koza* xuất bản cuốn sách có tên Lập trình di truyền, trong đó mô tả một loại giải thuật di truyền mới có thể được sử dụng để phát triển các chương trình máy tính.
* **Thập niên 1990**: Giải thuật di truyền được sử dụng để giải quyết nhiều vấn đề trong nhiều lĩnh vực khác nhau, bao gồm kỹ thuật, tài chính và học máy.
* **Thập niên 2000 đến nay**: Các giải thuật di truyền tiếp tục được phát triển và áp dụng cho các bài toán mới.

1. **Initialization**

Initialization là quá trình thiết lập giá trị ban đầu cho các giải pháp trong quần thể. Các giải pháp này thường được biểu diễn dưới dạng gen. Việc khởi tạo quần thể giải pháp một cách chính xác có thể ảnh hưởng đến hiệu suất và sự đa dạng của quần thể giải pháp trong quá trình thực hiện giải thuật di truyền.

Việc chọn phương pháp khởi tạo thường phụ thuộc vào bản chất của vấn đề cụ thể và kiến thức trước đó của người ứng dụng giải thuật này. Sự cân nhắc cẩn thận về quá trình khởi tạo có thể ảnh hưởng lớn đến hiệu suất và tốc độ của giải thuật di truyền.

1. **Population**
2. **Gen**

Trong giải thuật di truyền, Gen là một đơn vị cơ bản của thông tin di truyền, tương đương với các đặc trưng hoặc biến trong vấn đề tối ưu hóa được giải pháp. Gen thường được biểu diễn dưới dạng một phần của cá thể. Mỗi gen mang theo thông tin liên quan đến một khía cạnh cụ thể của giải pháp.

Trong quá trình thực hiện giải pháp, gen có thể trải qua các phép toán di truyền như lai ghép (*Crossover*) và đột biến (*Mutation*), tạo ra các gen mới ở thế hệ tiếp theo. Các phép toán này giúp tạo ra đa dạng trong quần thể và tìm kiếm không gian giải pháp một cách hiệu quả hơn. Ví dụ, trong demo tìm kiếm chuỗi mục tiêu, đại diện của mỗi gen sẽ là các chữ cái, con số, ký tự đặt biệt, chuỗi các gen liên kết lại với nhau sẽ tạo ra một chuỗi hoàn chỉnh có thể gọi đó là một nhiễm sắc thể.

1. **Chromosome**

Chromosome là một cấu trúc dữ liệu biểu diễn một cá thể, và thường được tổ chức dưới dạng một chuỗi các Gen. Mỗi Chromosome đại diện cho một cá thể hoặc một nhiễm sắc thể trong quần thể, trong nghiên cứu này chúng ta sẽ xem một cá thể như một nhiễm sắc thể (giải pháp).

A close-up of a screen

Description automatically generated

*Hình 2.5 Mô phỏng về Gen và Chromosome*

Mỗi Gen trong Chromosome biểu diễn một đặc trưng cụ thể hoặc một giá trị của nhiễm sắc thể. Qua các bước thực hiện giải pháp, các phương pháp di truyền như lai ghép (Crossover) và đột biến (Mutation) được thực hiện trên Chromosome để tạo ra các Chromosome mới. Quá trình này nhằm mục đích tạo ra sự đa dạng giải pháp trong quần thể. Với demo tìm chuỗi mục tiêu, mỗi Chromosome biểu diễn một chuỗi có các đặc trưng khác nhau với các chữ cái, con số, ký tự đặc biệt được khởi tạo ngẫu nhiên.

1. **Population**

Population là tập hợp của các cá thể đơn lẻ, có thể hiểu cách khác population đại diện cho một không gian giải pháp, giải pháp ở đây sẽ là các nhiễm sắc thể được tạo thành từ các Gen, Population là một thành phần không thể thiếu cho các thuật toán mang hơi hướng di truyền vì Population sẽ là nơi chứa các giải pháp tối ưu cho bài toán.

A rectangular image with a red border

Description automatically generated with medium confidence

*Hình 2.6 Mô phỏng về Population*

Population cũng sẽ trải qua các phương pháp như Fitness, Selection, Crossover, Mutation để tìm ra các giải pháp tốt nhất trong quần thể hiện hành. Population trong giải thuật di truyền là nền tảng quan trọng, đóng vai trò quyết định đến khả năng tiến hóa và tối ưu hóa cho giải pháp.

1. **Fitness**

Fitness sẽ là bước đánh giá độ tương tích của các giải pháp có trong quần thể, fitness chia ra khá nhiều phương pháp khác nhau để đánh giá các giải pháp như: Absolute Fitness, Relative Fitness, Pareto Optimality,... Nhưng trong demo tìm kiếm chuỗi mục tiêu này sẽ thực hiện ứng dụng phương pháp Abolute Fitness.

1. **Absolute fitness**

Phương pháp này khá trực diện, công việc của phương pháp này là đem các nhiễm sắc thể có trong Population so sánh cá đặc trưng của các nhiễm sắc thể này với nhiễm sắc thể mục tiêu, từ đó dựa trên sự khác nhau đó thì phương pháp này sẽ gán điểm Fitness cho nhiễm sắc thể đó. Tạo thuận lời cho bước tiếp theo là selection (chọn lọc).

A group of squares with letters and numbers

Description automatically generated

*Hình 2.7 So sánh chuỗi bằng Absolute Fitness*

Lấy ví dụ so sánh chuỗi ở hình trên, chuỗi mục tiêu đã được xác định là “1 # C B” và chuỗi được lấy từ quần thể hiện hành là “H A C H”, giải thuật sẽ duyệt từ đầu chuỗi cho đến hết, ở vị trí số 0 so sánh hai ký tự với nhau, với ‘H’ và ‘1’ khác nhau thì điểm Fitness sẽ được cộng 1.

A group of squares with text

Description automatically generated

*Hình 2.8 So sánh chuỗi bằng Absolute Fitness*

Lược đến vị trí số 2, hai ký tự được so sánh là ‘C’ và ‘C’, hai ký tự giống nhau, nghĩa là chuỗi ở quần thể đã có Gen trùng với chuỗi mục tiêu vì vậy điểm Fitness vẫn được ngẫu nhiên

1. Fitness = 0

2. if Chromosome\_in\_population != Chromosome\_Target:

3. Fitness += 1

1. **Selection**

Selection là một trong những bước quan trọng trong quá trình thực hiện giải thuật đi truyền, selection sẽ thức hiện công việc chọn lọc ra các giải pháp tốt nhất trong quần thể giải pháp hiện hành. Việc chọn một phương pháp Selection cho giải thuật cũng không dễ dàng, vì độ phức tạm của các bài toán cũng khác nhau. Cũng giống như các bước trong giải thuật di truyền, bước Selection cũng chia ra khá nhiều phương pháp khác nhau, có thể kể đến như: Roulette Wheel Selection, Tournament Selection, Rank – Based Selection, Stochatic Universal Sampling, Boltzmann Selection, Elitist Selection, Sigma Scaling, Truncation Selection, demo tìm kiếm chuỗi mục tiêu sẽ kết hợp hai phương pháp Rank – Based Selection và Elitist Selection.

1. **Rank – Based selection**

Rank – Based Selection là một trong những phương pháp khá phổ biến trong giải thuật di truyền, phương pháp này sẽ chọn lọc các cá thể có điểm Fitness tương đối tốt, có thể hiểu phương pháp này chọn ra các cá thể nào ở đầu danh sách các cá thể có điểm Fitness tốt nhất, thế mạnh của phương pháp này là chọn nhiều cá thể tốt tránh việc bỏ soát. Trong demo chúng ta sẽ ứng dụng phương pháp Rank – Based Selection để chọn các chuỗi có độ tương thích cao so với chuỗi mục tiêu.

A grey rectangular object with red text

Description automatically generated

*Hình 2.9 Population A được gán điểm Fitness*

Ở ví dụ trên Population A có bốn nhiễm sắc thể với các điểm Fitness đã được gán lần lượt 4, 7, 11 và 5, nhiệm vụ của phương pháp Rank – Based Selection là sắp xếp các nhiễm sắc thể này dựa trên điểm Fitness theo thứ tự giá trị từ nhỏ đến lớn.

**A grey rectangular object with red numbers

Description automatically generated**

*Hình 2.10 Population A sau khi được sắp xếp*

1. **Elitist selection**

Mục tiêu của phương pháp này là luôn giữ được các cá thể tốt nhất cho các thế hệ sau, để thực hiện được việc đó Elitist Selection cần đến một số lượng hoặc phần trăm nhất định được xác định từ trước.

**A diagram of numbers and a row of squares

Description automatically generated**

*Hình 2.11 Minh họa bước Elitist Selection*

Hình minh họa trên minh họa Population A có 7 nhiễm sắc thể được gán điểm Fitness và được sắp xếp theo điểm Fitness từ nhỏ đến lớn, nhiệm vụ của Elitist Selection là dựa trên số phần trăm đã được xác định và chọn ra các nhiễm sắc thể số phần trăm đó, số phần trăm được xác định là 20%, số cá thể được chọn sẽ được thêm vào quần thể sau để thực hiện các bước kế tiếp trong giải thuật.

1. **Crossover**

Crossover (*lai tạo*) là một quá trình quan trọng trong giải thuật di truyền, mục tiêu của quá trình này là tạo ra cá thể dựa trên những thông tin hoặc gen của hai hoặc nhiều cá thể đã có sẵn (*được gọi là bố mẹ*). Trong Crossover lại chia ra nhiều phương pháp khác nhau như: Single – Point Crossover, Tow – Point Crossover, Uniform Crossover, Multi – Point Crossover, Arithmetic Crossover, Order Crossover, Cycle Crossover, Edge Recombination Crossover. Demo “*Tìm kiếm chuỗi mục tiêu được nhập từ bàn phím*” sẽ được áp dụng phương Uniform Crossover.

1. **Uniform crossover**

Uniform Crossover là phương pháp lai tạo dựa trên xác suất ngẫu nhiên, đầu tiên chọn ra hai cá thể tốt từ quần thể, dựa trên xác suất ngẫu nhiên mà giải thuật sẽ quyết định gen của bố hoặc mẹ sẽ được chèn vào gen của cá thể con, quá trình này sẽ được lập lại cho đến khi tạo ra cá thể con hoàn chỉnh có gen của bố hoặc mẹ.

**A screenshot of a computer

Description automatically generated**

*2.11 Minh họa cách thực hiện của Uniform Crossover*

1. **Mutation**

Cũng như Selection hay Crossover thì Mutation là quá trình quan trọng không kém, đột biến trong giải thuật di truyền là một quá trình thiết yếu nhầm mang lại sự đa dạng các cá thể trong quần thể, công việc của quá trình này là thay đổi gen hoặc một số đặc tính của cá thể đó, mục tiêu của quá trình này là duy trì sự đa dạng cho quần thể. Có khá nhiều phương pháp để thực hiện quá trình Mutation như: Bit Flip Mutation, Swap Mutation, Scramble Mutation, Inversion Mutation, Insertion Mutation, Shuffle Mutation, Creep Mutation, Boundary Mutation. Trong demo này sẽ ứng dụng phương pháp Shuffle Mutation.

1. **Shuffle Mutation**

Shuffle Mutation là một dạng đột biến ngẫu nhiên, bước đầu tiên của phương pháp này là chọn một đoạn hoặc một Gen đơn lẻ ngẫu nhiên, sau đó chèn các Gen đó vào cá thể. Phụ thuộc vào mỗi bài toán Gen sẽ được xác định là gì và cách thức chọn ngẫu nhiên cũng khác. Trong demo tìm kiếm chuỗi mục tiêu các Gen được đột biến sẽ được lấy từ bộ Gen được khai báo trong bước Initialization.

1. Genes = “A B C D E F G H I J K L M N O P Q R S T U V”

2. Gen = random.string(Genes)

3. Return Gen

# **Chương 3**

**PHƯƠNG PHÁP THỰC HIỆN**

1. **Giới thiệu Demo**

Trong Chương 2 đã trình bày các cơ sở lý thuyết về các bước thực hiện của giải thuật di truyền và các phương pháp, trong Chương 3 này sẽ ứng dụng các lý thuyết, phương pháp đó vào trong demo được xác định là “*Tìm chuỗi mục tiêu được nhập ngẫu nhiên*”.

Mặc dù độ phức tạp của demo chưa cao khi đem so sánh với các bài toán khác, nhưng thông qua demo này sẽ cho thấy một phần nào đó về cách thức mà giải thuật di truyền hoạt. Các phương pháp sẽ được trình bày thông qua các dòng lệnh sẽ được hộ trợ bởi các ngô ngữ, công cụ sau:

* **Ngôn ngữ lập trình**: Python 3.8.2
* **Môi trường phát triển**: Visual Studio Code

1. **Khởi tạo kích thước quần thể chuỗi (Initialization)**

Trong demo này việc khởi tạo một quần thể chuỗi khá đơn giản, bước đầu tiên khởi tạo một biến Population\_Size và kế tiếp gán giá trị cho biến này, trong demo này sẽ gán giá trị 100 với kiểu dữ liệu là int. Nghĩa là quần thể chuỗi đầu tiên được khởi tạo có 100 chuỗi.

3. # The variable 'Population\_Size' creates a population of the specified number

4. Population\_Size : int = 100

**(1)**

1. **Quần thể chuỗi (Population)**
2. **Tạo bộ ký tự (Gen)**

Đối với giải thuật di truyền gen được xem như một phân tử nhỏ của một giải pháp, phân tử đó được xác định là gì thì tùy thuộc vào mỗi bài toán, điều quan trọng chúng ta sẽ làm là thực hiện công việc xác định được đâu sẽ là phân tử tạo thành một giải pháp từ đó việc tìm được giải pháp cho bài toán sẽ dễ dàng hơn. Với demo “*Tìm chuỗi mục tiêu được nhập ngẫu nhiên từ bàn phím*” chúng ta sẽ xác định mục tiêu cần đạt được thông qua giải thuật di truyền sẽ là một chuỗi ngẫu nhiên được nhập, chuỗi này sẽ là tập hợp các chữ cái có dấu hoặc không có dấu, con số, ký tự đặc biệt.

5. # Declare variable 'Genes'

6. Genes = '''aáàảãạâấầẩẫậăắằẳẵặ bcd đ eéèẻẽẹ êếềểễệ

7.            fgh iíìỉĩị jklmn oóòỏõọơớờởỡợôốồổỗộ pq

8.            rst uúùủũụưứừửữự vwx yýỳỷỹỵ z AÁÀẢÃẠÂẤ

9.            ẦẨẪẬĂẮẰẲẴẶ BCD Đ EÉÈẺẼẸ ÊẾỀỂỄỆ FGH IÍÌ

10.            ỈĨỊ JKLMN OÓÒỎÕỌƠỚỜỞỠỢÔỐỒỔỖỘ PQRST UÚÙ

11.            ỦŨỤƯỨỪỬỮỰ VWX YÝỲỶỸỴ Z0123456789,.-;:\_

12.            !"#%&/()=?@${[]}'''

**(1)**

Demo tìm chuỗi mục tiêu việc khai báo một bộ gen gồm các chữ cái có dấu hoặc không có dấu, con số, ký tự đặc biệt là rất quan trọng, một bộ gen đầy đủ sẽ hỗ trợ rất nhiều trong việc tìm kiếm chuỗi mục tiêu, phải luôn đảm bảo rằng bộ gen đầy đủ, vì một số trường hợp nếu thiếu đi một gen mà gen đó được nhập từ bàn phím thì việc tìm kiếm sẽ diễn ra vô tận, vì gen khai báo đã thiếu không tìm thấy được.

1. **Tạo chuỗi từ ký tự (Chromosome)**

Các chữ cái, con số, ký tự đặc biệt đóng vai trò là các gen riêng lẻ, khi quá trình tạo ra các nhiễm sắc thể cho quần thể đầu tiên thì các gen sẽ được gọi ngẫu nhiên để liên kết tạo thành một chuỗi (*nhiễm sắc thể*). Khởi tạo hàm Create\_Chromosome() để tạo ra các chuỗi ngẫu nhiên, biến Target sẽ được khởi tạo là toàn cục, biến Chromosome\_Len sẽ nhận giá trị là độ dài của chuỗi mục tiêu được nhập vào, sử dụng vòng lập For với tham số là Chromosome\_Len (*độ dài của chuỗi mục tiêu*).

27. # Create chromosomes

28. def Create\_Chromosome(self):

29.      global Target

30.      Chromosome\_Len = len(Target)

31.      return [self.Gene\_Mutation() for \_ in range(Chromosome\_Len)]

**(1)**

Đối với hàm Genetic\_Mutation() gọi các ký tự ngẫu nhiên, từ đó tạo ra các chuỗi với độ dài thì như nhau nhưng sẽ khác về các ký tự.

1. **Xây dựng quần thể chuỗi từ chuỗi**

Demo này chúng ta xem quần thể là tập hợp các chuỗi đã được tạo ra ngẫu nhiên với số lượng là 100 chuỗi, điểm chung của các cá thể này đều được tạo ra từ cùng một bộ gen là các chữ cái, con số, ký tự đặc biệt. Trong quá trình giải thuật đi tìm chuỗi mục tiêu quần thể ban đầu sẽ thay đổi qua từng giai đoạn của giải thuật di truyền, vì tìm ra chuỗi mục tiêu quần thể sẽ được đánh giá độ thích nghi của từng cá thể, lai tạo sinh ra cá thể mới, đột biến gen ký tự.

27. # Create chromosomes

28. def Create\_Chromosome(self):

29.      global Target

30.      Chromosome\_Len = len(Target)

31.      return [self.Gene\_Mutation() for \_ in range(Chromosome\_Len)]

**(1)**

Để khởi tạo một quần thể chuỗi trong demo này, chúng ta sử dụng lại hàm Create\_Chromosome() để tạo ra quần thể chuỗi với các chuỗi được tạo ra mang tính ngẫu nhiên, để các chuỗi được tạo ra mang tính ngẫu nhiên chúng ta cần đến hàm Genetic\_Mutation(), hàm này sẽ được trình bày trong phần sau.

9. # Initiialization

10. for \_ in range(Population\_Size):

11.     # Call the function "Create\_Chromosome" to create chromosomes

12.     Chromosome = Individual.Create\_Chromosome()

13.     # Add chromosomes to the "Population" array

14.     Population.append(Individual(Chromosome))

**(2)**

1. **Đánh giá độ tương thích của chuỗi (Fitness)**
2. **Ứng dụng phương pháp Absolute Fitness**

Các chuỗi này sẽ được đánh giá dựa trên chuỗi mục tiêu bằng cách so sánh các gen (*chữ cái, con số, ký tự đặc biệt*) của hai chuỗi với nhau. Cần đến một hàm Fitness dựa trên phương pháp Absolute fitness để thực hiện nhiệm vụ này.

50. # Evaluate the fitness of the chromosome, compared to the target sequence

51. def Fitness(self):

52.     global Target

53.     fitness = 0

54.     for fitness\_Chromosome, fitness\_Target in zip(self.chromosome, Target):

55.         if fitness\_Chromosome != fitness\_Target:

56.             fitness += 1

57.       return fitness

**(1)**

Trong hàm Fitness() sẽ khởi tạo biến fitness được gán giá trị mặc định là 0, vòng lập for sử dụng hai biến lần lượt là fitness\_Chromosome và fitness\_Target đại diện cho gen của chuỗi được so sánh và gen của chuỗi mục tiêu là self.chromosome và Target. Sử dụng hàm zip() để so sánh từng từ ký tự với nhau của hai chuỗi, với mỗi ký tự khác nhau giữa hai chuỗi thì biến fitness sẽ được cộng 1, nghĩa là chuỗi có điểm fitness càng cao thì độ tương thích với chuỗi mục tiêu càng thấp và ngược lại chuỗi có điểm fitness thấp thì lại tương thích với chuỗi mục tiêu. Quá trình so sánh sẽ kết thúc khi vòng lặp for so sánh hết các ký tự của hai chuỗi và trả về chuỗi đã được so sánh cùng với điểm fitness, dựa trên điểm fitness này chúng ta đánh giá được độ tương thích của từng chuỗi qua những lần so sánh với chuỗi mục tiêu.

1. **Chọn lọc chuỗi (Selection)**
2. **Ứng dụng phương pháp Rank – Based Selection**

Đầu tiên sẽ ứng dụng phương pháp Rank – Based Selection để chọn các chuỗi có độ tương thích cao so với chuỗi mục tiêu.

17. # Sort chromosomes in ascending order based on fitness score

18. Population = sorted(Population, key = lambda x:x.fitness)

**(2)**

Population được khởi tạo trước đó và đã có các cá thể được tạo ra, nhiệm vụ là sắp xếp các chuỗi có điểm fitness thấp nhất cho đến cao nhất, vì trong demo này chuỗi có điểm fitness thấp nhất sẽ có độ tương thích với chuỗi mục tiêu nhất và ngược lại chuỗi có điểm fitness càng cao thì độ tương thích với chuỗi mục tiêu càng thấp. Mặc định của hàm sorted() là sắp xếp từ giá trị nhỏ nhất cho đến giá trị lớn nhất, tham số của hàm sẽ là Population, hàm lambda sẽ giúp cho hàm sorted() sắp xếp các chuỗi dựa trên điểm fitness.

1. **Ứng dụng phương pháp Elitist Selection**

Để thực hiện được việc đó Elitist Selection cần đến một số lượng hoặc phần trăm nhất định được xác định từ trước.

23. # Create a new generation and select good individuals into this new position

24. New\_Generation = []

25. # Perform a transfer of 10% of individuals from the current population to the next generation

26. Size = int((10 \* Population\_Size) / 100)

27. # Add 10% of the old population's strings to the new population using the extend() function

28. New\_Generation.extend(Population[:Size])

**(2)**

Đầu tiên khởi tạo mảng New\_Generation (thế hệ mới), biến Size sẽ mang giá trị là 10% chuỗi đầu tiên trong mảng Population đã được sắp xếp trong thứ tự của chuỗi bằng phương pháp Rank – Based Selection trước đó, sử dụng hàm extend() thêm 10% chuỗi đó vào mảng New\_Generation.

1. **Lai tạo chuỗi (Crossover)**
2. **Ứng dụng phương pháp Uniform Crossover**

Với phương pháp Uniform Crossover được ứng dụng vào demo thì đầu tiên khởi tạo hàm Crossover() có một tham số là Parents, mảng Child\_Chromosome sẽ chứa các cá thể con đã được lai tạo, dùng vòng lập for để duyệt qua hai chuỗi bố mẹ, hai biến Parent\_1 và Parent\_2 đại diện cho các ký tự của self.chromosome , Parents.chromosome, xác suất được tạo ra ngẫu nhiên bởi biến Probability với giá trị gọi từ hàm random().

32. # Create a new chromosome by the 'Crossover' function

33. def Crossover(self, Parents):

34.     # Create an array containing the child chromosomes

35.     Child\_Chromosome = []

36.     for Parent\_1, Parent\_2 in zip(self.chromosome, Parents.chromosome):

37.         # Generate random probabilities

38.         Probability = random.random()

39.         # If probability is less than 0.45, insert Parent\_1's gene

40.         if Probability < 0.45:

41.              Child\_Chromosome.append(Parent\_1)

42.         # If the probability is between 0.45 - 0.90, insert Parent\_2's gene

43.         elif Probability < 0.90:

44.              Child\_Chromosome.append(Parent\_2)

45.         # The remaining case is a genetic mutation using the "Gene\_Mutation" function

46.         else:

47.              Child\_Chromosome.append(self.Gene\_Mutation())

48.    # Returns the child chromosome

49. return Individual(Child\_Chromosome)

**(1)**

Nếu như Probability nhỏ hơn 0.45 thì giải thuật sẽ dùng hàm append() để chèn Parent\_1 vào biến Child\_Chromosome hoặc Probability nhỏ hơn 0.90 thì giải thuật sẽ chèn Parent\_2 vào biến Child\_Chromosome và ngược lại hai trường hợp trên thì giải thuật sẽ gọi hàm Genetic\_Mutation() để đột biến một ký tự ngẫu nhiên và chèn ký tự đó vào biến Child\_Chromosome. Quá trình lai tạo sẽ kết thức khi vòng lập for duyệt hết hai chuỗi.

1. **Đột biến chuỗi (Mutation)**
2. **Ứng dụng phương pháp Shuffle Mutation**

Bước đầu tiên của phương pháp này là chọn một đoạn hoặc một gen đơn lẻ ngẫu nhiên, sau đó sắp xếp các đoạn đó vào cá thể. Với demo tìm chuỗi mục tiêu phương pháp Shuffle Mutation sẽ thực hiện chọn ngẫu nhiên một chữ cái, con số, ký tự đặc biệt trong bộ gen đã có sẵn, sau đó các ký tự này sẽ được sắp xếp thành các chuỗi bằng cách đưa vào quá trình Create\_Chromosome().

21. # The 'Gene Mutation' function is used to create mutated genes

22. def Gene\_Mutation(self):

23.     global Genes

24.     Gen = random.choice(Genes)

25.     return Gen

**(1)**

Biến Genes sẽ được khởi tạo toàn cục trong suốt quá trình thực hiện giải thuật di truyền, khởi tạo biến Gen biến này được gán giá trị là các ký tự được chọn ngẫu nhiên bằng hàm choice() với tham số sẽ là biến Genes toàn cụ, kết quả được trả về là một ký tự ngẫu nhiên đã được chọn. Các ký tự đã được chọn sẽ được tạo thành chuỗi bằng hàm Create\_Chromosome().

27.     # Create chromosomes

28.     def Create\_Chromosome(self):

29.         global Target

30.         Chromosome\_Len = len(Target)

31.         return [self.Gene\_Mutation() for \_ in range(Chromosome\_Len)]

**(1)**

# **Chương 4**

**KẾT QUẢ THỰC HIỆN VÀ ĐÁNH GIÁ PHƯƠNG PHÁP**

1. **Kết quả thực hiện**

Trong phần kết quả thực hiện này sẽ trình bày kết quả của chương trình demo, để thực hiện điều đó chúng ta sẽ chuẩn bị một chuỗi mục tiêu trước, chuỗi mục tiêu được xác định là : “**Can Tho**”, với chuỗi này sẽ được thực hiện tìm kiếm hai lần để có thể thấy sự khác nhau của chương trình demo.

1. **Kết quả thực hiện lần 1**

**A screenshot of a computer

Description automatically generated**

*Hình 4.1 Kết quả thực hiện lần 1*

1. **Kết quả thực hiện lần 2**

**A screenshot of a computer

Description automatically generated**

*Hình 4.2 Kết quả thực hiện lần 2*

1. **Đánh giá phương pháp**

Với hai lần thực hiện chương trình demo có thể thấy sự khác biệt ở số lượng Generation (thế hệ), String (chuỗi được nhập), Fitness (độ tương thích), nhưng đều tìm kiếm đúng chính xác chuỗi mục tiêu. Để có sự khác nhau đó nhờ vào việc ứng dụng các phương pháp vào trong quá trình tìm kiếm chuỗi mục tiêu.

1. **Đánh giá phương pháp Absolute Fitness**

Cũng như đã được trình bày ở **Chương 2** phương pháp Absolute Fitness là một trong những phương pháp đánh giá nhiễm sắc thể phổ biến trong bước Fitness. Trong chương trình tìm kiếm chuỗi mục tiêu phương pháp này đã thực hiện tốt việc so sánh giữa chuỗi trong quần thể hiện hành với chuỗi mục tiêu, có thể nói phương pháp này đã góp phần tìm kiếm chuỗi mục tiêu một cách nhanh hơn rất nhiều, dù biết rằng sẽ đến giai đoạn nào đó các ký tự không được đột biến theo hướng tốt thì việc tìm kiếm chuỗi mục tiêu sẽ ra chậm hơn.

Absolute sẽ là một trong những phương pháp đánh giá độ tương tích chuỗi tốt cho những bài toán có liên quan đến chuỗi.

1. **Đánh giá phương pháp Rank – Based Selection và Elitist Selection**

Ở phương pháp này đã nhận được sự hỗ trợ không ít từ phương pháp Absolute Fitness trước đó, công việc của phương pháp này là tìm ra các chuỗi có điểm Fitness thấp nhất đưa về đầu danh sách chuỗi và ngược lại các chuỗi có điểm Finess cao sẽ được đưa về sau danh sách. Có thể đánh giá phương pháp này ở mức tốt, bởi vì công việc của phương pháp chỉ là sắp xếp các chuỗi đã có sẵn điểm Fitness, nhưng với phương pháp này sẽ đảm bảo rằng chuỗi tốt nhất luôn nằm đầu danh sách.

Tiếp theo, phần công việc còn lại của bước Selection sẽ được thực hiện bởi phương pháp Elitist Selection, cũng như phương pháp Rank – Based Selection được nhận sự hỗ trợ, việc của Elitist Selection là dựa trên số phần trăm đã được xác định

1. **Đánh giá phương pháp Uniform Crossover**
2. **Đánh giá phương pháp Shuffle Mutation**

# **Chương 5**

**KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN**

1. **Kết luận**
2. **Hướng phát triển**

**TÀI LIỆU THAM KHẢO**

**PHỤ LỤC**

1. **Module, hàm và biến của demo:**

1. # Import the “random” module, the “random” is module supports return random numbers, characters

2. import random

3. # The variable 'Population\_Size' creates a population of the specified number

4. Population\_Size : int = 100

5. # Declare variable 'Genes'

6. Genes = '''aáàảãạâấầẩẫậăắằẳẵặ bcd đ eéèẻẽẹ êếềểễệ

7.            fgh iíìỉĩị jklmn oóòỏõọơớờởỡợôốồổỗộ pq

8.            rst uúùủũụưứừửữự vwx yýỳỷỹỵ z AÁÀẢÃẠÂẤ

9.            ẦẨẪẬĂẮẰẲẴẶ BCD Đ EÉÈẺẼẸ ÊẾỀỂỄỆ FGH IÍÌ

10.            ỈĨỊ JKLMN OÓÒỎÕỌƠỚỜỞỠỢÔỐỒỔỖỘ PQRST UÚÙ

11.            ỦŨỤƯỨỪỬỮỰ VWX YÝỲỶỸỴ Z0123456789,.-;:\_

12.            !"#%&/()=?@${[]}'''

13. # The variable 'Target' contains the random string to enter

14. Target = str(input("Enter target: "))

15. class Individual(object):

16.     # The \_\_init\_\_ function represents an individual in a population

17.     def \_\_init\_\_(self, Value\_Chromosome):

18.         self.chromosome = Value\_Chromosome

19.         self.fitness = self.Fitness()

20.     @classmethod

21.     # The 'Gene Mutation' function is used to create mutated genes

22.     def Gene\_Mutation(self):

23.         global Genes

24.         Gen = random.choice(Genes)

25.         return Gen

26.     @classmethod

27.     # Create chromosomes

28.     def Create\_Chromosome(self):

29.         global Target

30.         Chromosome\_Len = len(Target)

31.         return [self.Gene\_Mutation() for \_ in range(Chromosome\_Len)]

32.     # Create a new chromosome by the 'Crossover' function

33.     def Crossover(self, Parents):

34.         # Create an array containing the child chromosomes

35.         Child\_Chromosome = []

36.         for Parent\_1, Parent\_2 in zip(self.chromosome, Parents.chromosome):

37.             # Generate random probabilities

38.             Probability = random.random()

39.             # If probability is less than 0.45, insert Parent\_1's gene

40.             if Probability < 0.45:

41.                 Child\_Chromosome.append(Parent\_1)

42.             # If the probability is between 0.45 - 0.90, insert Parent\_2's gene

43.             elif Probability < 0.90:

44.                 Child\_Chromosome.append(Parent\_2)

45.             # The remaining case is a genetic mutation using the "Gene\_Mutation" function

46.             else:

47.                 Child\_Chromosome.append(self.Gene\_Mutation())

48.         # Returns the child chromosome

49.         return Individual(Child\_Chromosome)

50.     # Evaluate the fitness of the chromosome, compared to the target sequence

51.     def Fitness(self):

52.         global Target

53.         fitness = 0

54.         for fitness\_Chromosome, fitness\_Target in zip(self.chromosome, Target):

55.             if fitness\_Chromosome != fitness\_Target:

56.                 fitness += 1

57.         return fitness

1. **Main khởi tạo các giá trị của demo:**

1. from Genetic\_Algorithm import \*

2. # Main Function

3. def Main():

4.     global Population\_Size

5.     # Initialize the first generation

6.     Generation = 1

7.     Found = False

8.     Population = []

9.     # Initiialization

10.     for \_ in range(Population\_Size):

11.         # Call the function "Create\_Chromosome" to create chromosomes

12.         Chromosome = Individual.Create\_Chromosome()

13.         # Add chromosomes to the "Population" array

14.         Population.append(Individual(Chromosome))

15.     # Selection

16.     while not Found:

17.         # Sort chromosomes in ascending order based on fitness score

18.         Population = sorted(Population, key = lambda x:x.fitness)

19.         # If the chromosome has a fitness score of 0 then the target chromosome has been found

20.         if Population[0].fitness <= 0:

21.             Found = True

22.             break

23.         # Create a new generation and select good individuals into this new position

24.         New\_Generation = []

25.         # Perform a transfer of 10% of individuals from the current population to the next generation

26.         Size = int((10 \* Population\_Size) / 100)

27.         # Add 10% of the old population's strings to the new population using the extend() function

28.         New\_Generation.extend(Population[:Size])

29.         # 50% of the 90% of individuals in the current population will undergo 'Crossover'

30.         Size = int((90 \* Population\_Size) / 100)

31.         # Crossover

32.         for \_ in range(Size):

33.             Parent\_1 = random.choice(Population[:50])

34.             Parent\_2 = random.choice(Population[:50])

35.             Child = Parent\_1.Crossover(Parent\_2)

36.             New\_Generation.append(Child)

37.         Population = New\_Generation

38.         print("Generation: {}\tString: {}\tFitness: {}".\

39.             format(Generation,"".join(Population[0].chromosome),Population[0].fitness))

40.         Generation += 1

41.     print("Generation: {}\tString: {}\tFitness: {}".\

42.             format(Generation,"".join(Population[0].chromosome),Population[0].fitness))

43. if \_\_name\_\_ == '\_\_main\_\_':

44.     Main()