Mục lục

[Mục lục 1](#_Toc358798522)

[1.1. Danh mục các bảng 2](#_Toc358798523)

[1.2. Danh mục các hình 2](#_Toc358798524)

[Chương 1 2](#_Toc358798525)

[MỞ ĐẦU 2](#_Toc358798526)

[1.1. Phạm vi và mục tiêu 2](#_Toc358798527)

[1.2. Những đóng góp chính của khóa luận 2](#_Toc358798528)

[1.3. Cấu trúc khóa luận 2](#_Toc358798529)

[1.4. Qui ước ký hiệu và viết tắt 2](#_Toc358798530)

[Chương 2 2](#_Toc358798531)

[TỔNG QUAN VỀ GIẢI THUẬT VÀ ĐỘ PHỨC TẠP CỦA GIẢI THUẬT 2](#_Toc358798532)

[2.1. Giới thiệu 2](#_Toc358798537)

[2.2. Giải thuật và các tính chất của giải thuật 2](#_Toc358798538)

[2.3. Một số kỹ thuật cơ bản để thiết kế giải thuật 2](#_Toc358798539)

[2.4. Độ phức tạp của giải thuật 2](#_Toc358798542)

[2.5. Phân tích giải thuật 2](#_Toc358798543)

[2.6. Một số công cụ toán học hỗ trợ phân tích độ phức tạp của giải thuật 2](#_Toc358798544)

[2.7. Kết luận 2](#_Toc358798545)

[Chương 3 Error! Bookmark not defined.](#_Toc358798546)

[GIẢI THUẬT NGẪU NHIÊN Error! Bookmark not defined.](#_Toc358798547)

[3.1. Giới thiệu **Error! Bookmark not defined.**](#_Toc358798549)

[3.2. Định nghĩa và ví dụ giải thuật ngẫu nhiên **Error! Bookmark not defined.**](#_Toc358798550)

[3.3. Phân loại các giải thuật ngẫu nhiên **Error! Bookmark not defined.**](#_Toc358798551)

[3.4. Giải thuật Monte Carlo **Error! Bookmark not defined.**](#_Toc358798552)

[3.5. Giải thuật Las Vegas **Error! Bookmark not defined.**](#_Toc358798553)

[3.6. Các lĩnh vực áp dụng của giải thuật ngẫu nhiên **Error! Bookmark not defined.**](#_Toc358798554)

[3.7. Kết luận **Error! Bookmark not defined.**](#_Toc358798555)

[Chương 4 2](#_Toc358798556)

[GIẢI THUẬT NGẪU NHIÊN CHO MỘT SỐ BÀI TOÁN 2](#_Toc358798557)

[4.1. Giới thiệu 2](#_Toc358798559)

[4.2. Bài toán sắp xếp 2](#_Toc358798560)

[4.3. Bài toán xác định số nguyên tố 2](#_Toc358798561)

[4.4. Bài toán tìm cặp điểm gần nhất 2](#_Toc358798562)

[4.5. Bài toán so trùng mẩu (có thể chọn bài khác) **Error! Bookmark not defined.**](#_Toc358798563)

[4.6. Bài toán tìm cây bao trùm nhỏ nhất của đồ thị 2](#_Toc358798564)

[4.7. Kết luận 2](#_Toc358798565)

[Chương 5 2](#_Toc358798566)

[HIỆN THỰC GIẢI THUẬT NGẪU NHIÊN CHO MỘT SỐ BÀI TOÁN 2](#_Toc358798567)

[5.1. Giới thiệu 2](#_Toc358798569)

[5.2. Bài toán sắp xếp 2](#_Toc358798570)

[5.3. Bài toán xác định số nguyên tố 2](#_Toc358798571)

[5.4. Bài toán tìm cặp điểm gần nhất 2](#_Toc358798572)

[5.5. Bài toán so trùng mẩu 2](#_Toc358798573)

[5.6. Giải thuật ngẫu nhiên tìm cây bao trùm nhỏ nhất của đồ thị 2](#_Toc358798574)

[5.7. Kết luận 2](#_Toc358798575)

[Chương 6 2](#_Toc358798576)

[TỔNG KẾT VÀ ĐỀ NGHỊ 2](#_Toc358798577)

[6.1. Tổng kết 2](#_Toc358798579)

[6.2. Đề nghị 2](#_Toc358798580)

[TÀI LIỆU THAM KHẢO 2](#_Toc358798581)

## Danh mục các bảng

[Bảng 1 Bảng so sánh thời gian chạy (đơn vị tính: microsecond) của giải thuật ngẫu nhiên và giải thuật Prim 2](#_Toc358798582)

## Danh mục các hình

[Hình 4.6‑1 Đồ thị 2](#_Toc358798583)

[Hình 4.6‑2 Các cạnh được chọn trong Boruka step 2](#_Toc358798584)

[Hình 4.6‑3 Đồ thị sau khi xây dựng 2](#_Toc358798585)

[Hình 4.6‑4 Kết quả sau khi áp dụng Boruvka step lần thứ nhất 2](#_Toc358798586)

[Hình 4.6‑5 Các cạnh được chọn trong Boruvka step lần 2 2](#_Toc358798587)

[Hình 4.6‑6 Đồ thị được xây dựng lại từ Boruvka step lần 2 2](#_Toc358798588)

[Hình 4.6‑7 Đồ thị kết quả thu được từ Boruvka step lần 2 2](#_Toc358798589)

[Hình 4.6‑8 Cây bao trùm nhỏ nhất thu được 2](#_Toc358798590)

[Hình 4.6‑9 2](#_Toc358798591)

[Hình 4.6‑10 2](#_Toc358798592)

[Hình 4.6‑11 2](#_Toc358798593)

[Hình 4.6‑12 2](#_Toc358798594)

[Hình 5.6‑1 Giao diện chính của giải thuật tìm cây bao trùm nhỏ nhất 2](#_Toc358798595)

[Hình 5.6‑2 Nhập trọng số 2](#_Toc358798596)

[Hình 5.6‑3 Chọn vị trí lưu 2](#_Toc358798597)

[Hình 5.6‑4 Chọn một tập tin để mở. 2](#_Toc358798598)

[Hình 5.6‑5 Kết quả thu được khi thực hiện giải thuật tìm cây bao trùm nhỏ nhất 2](#_Toc358798599)

[Hình 5.6‑6 Đồ thị thời gian chạy. 2](#_Toc358798600)

[Hình 5.6‑7 2](#_Toc358798601)

[Hình 5.6‑8 2](#_Toc358798602)

[Hình 5.6‑9 2](#_Toc358798603)

[Hình 5.6‑10 2](#_Toc358798604)

[Hình 5.6‑11 2](#_Toc358798605)

[Hình 5.6‑12 2](#_Toc358798606)

[Hình 5.6‑13 2](#_Toc358798607)

[Hình 5.6‑14 2](#_Toc358798608)

[Hình 5.6‑15 2](#_Toc358798609)

[Hình 5.6‑16 Biểu đồ thời gian chạy của giải thuật Prim 2](#_Toc358798610)

[Hình 5.6‑17 Biểu đồ thời gian chạy của giải thuật ngẫu nhiên 2](#_Toc358798611)

# Chương 1

# MỞ ĐẦU

## Phạm vi và mục tiêu

Các nhà khoa học vào thế kỉ trước đã thấy rằng tính chất ngẫu nhiên là thành phần thiết yếu trong việc mô hình hóa và phân tích tự nhiên. Ngày nay, sự ngẫu nhiên đóng vai trò quan trọng trong hầu hết mọi lĩnh vực của khoa học từ di truyền học và tiến hóa trong sinh học đến mô hình biến động giá của nền kinh tế thị trường tự do. Khoa học máy tính cũng không ngoại lệ, phương pháp ngẫu nhiên và xác xuất đóng vai trò cốt lỗi trong khoa học máy tính hiện đại. Trong hai thập kỉ qua chúng ta đã chứng kiến sự phát triển to lớn của việc sử dụng lý thuyết xác xuất trong khoa học máy tính. Thêm vào đó nhiều kỹ thuật xác suất hiện đại và cải tiến tạo điều kiện cho việc phát triển các giải thuật ngẫu nhiên.

Khái niệm giải thuật ngẫu nhiên là một khái niệm tương đối mới. Trong mọi giải thuật được giới thiệu từ xưa đến nay, mỗi bước trong giải thuật đều được xác định. Với giải thuật ngẫu nhiên thì chúng ta không xác định được, trong quá trình thực hiện một giải thuật ngẫu nhiên nó sẽ đưa ra một chọn lựa tùy ý. Một số hành động được thực hiện một cách ngẫu nhiên. Ngoài đầu vào, giải thuật sữ dụng một nguồn các con số ngẫu nhiên. Trong quá trình tính toán, nó sử dụng những lựa chọn ngẫu nhiên phụ thuộc vào những số ngẫu nhiên. Output có thể khác nhau nếu giải thuật chạy nhiều lần trên cùng một input.

Hơn nữa, giải thuật ngẫu nhiên thường đơn giản và dễ thực hiện. Các giải thuật này chạy nhanh hơn giải thuật cổ điển giải cùng một bài toán. Yêu cầu về không gian của một giải thuật ngẫu nhiên có thể nhỏ hơn của giải thuật nhất định mà chúng ta biết để giải cùng một bài toán . Trong vài trường hợp, giải thuật ngẫu nhiên là cách duy nhất hoặc tốt nhất để giải quyết vấn đề.([3],[4],[5])

Giải thuật ngẫu nhiên và xác suất đóng vai trò quan trọng trong khoa học máy tính, với những ứng dụng từ tối ưu tổ hợp và máy tính học cách giao tiếp với mạng và giao thức bảo mật.

Hiện nay, các giải thuật được áp dụng trong lĩnh vực khoa học máy tính đã được tối ưu rất nhiều, các giải thuật này đã chạy rất nhanh. Tuy nhiên, thời gian rất quý giá, một giây cũng có thể thay đổi rất nhiều thứ. Với giải thuật ngẫu nhiên chúng ta sẽ tiết kiệm được rất nhiều thời gian so với giải thuật truyền thống cùng giải một bài toán. Thêm vào đó, giải thuật ngẫu nhiên có thể sử dụng ít tài nguyên hệ thống hơn. Với những ưu điểm mà giải thuật ngẫu nhiên mang lại, thì việc nghiên cứu các giải thuật ngẫu nhiên là hết sức có ý nghĩa. Vì vậy, chúng em đã chọn đề tài “*Nghiên cứu và ứng dụng các giải thuật ngẫu nhiên*” để làm khóa luận. Trong khóa luận này chúng em đã đầu tư nghiên cứu các giải thuật ngẫu nhiên, so sánh giải thuật ngẫu nhiên với các giải thuật cổ điển đã biết.

## Những đóng góp chính của khóa luận

Dưới đây là những đóng góp chính của khóa luận đối với lĩnh vực khoa học máy tính:

1. Trình bày khái niệm giải thuật và cách tính toán độ phực tạp của giải thuật
2. Trình bày khái niệm giải thuật ngẫu nhiên
3. Trình bày và áp dụng lý thuyết xác suất để tính toán độ phức tạp, cũng như khả năng cho ra kết quả đúng của giải thuật ngẫn nhiên.
4. Hiện thực một số giải thuật ngẫu nhiên cho các bài toán thuộc các lĩnh vực toán học, tìm kiếm, lý thuyết đồ thị.

## Cấu trúc khóa luận

Khóa luận bao gồm 6 chương.

Chương 1, giới thiệu phạm vi, mục tiêu, những đóng góp chính của khóa luận, cấu trúc khóa luận. Ngoài ra trong chương này chứa toàn bộ những qui ước kí hiệu và viết tắt được sữ dụng trong toàn bộ quyển sách này.

Chương 2, giới thiệu tổng quan về giải thuật và độ phức tạp của giải thuật. Đây là một số kiến thức cần thiết sữ dụng cho việc thiết kế giải thuật. Chương này còn giới thiệu các công cụ toán học hỗ trợ phân tích, đánh giá độ phức tạp của giải thuật

Chương 3, trình bày tổng quát về giải thuật ngẫu nhiên và phân loại giải thuật ngẫu nhiên

Chương 4, trình bày một số giải thuật ngẫu nhiên cho các bày toán thuộc nhiều lĩnh vực khác nhau như: sắp xếp, tìm kiếm, toán học, lý thuyết đồ thị….

Chương 5, là phần hiện thực các giải thuật ngẫu nhiên được nêu ra ở chương 4.

Chương 6, là phần tổng kết và đề nghị các hướng nghiên cứu trong tương lai liên quan đến giải thuật ngẫu nhiên.

## Qui ước ký hiệu và viết tắt

Trong sách này có một số qui ước và kí hiệu viết tắt như sau:

: Với mọi

: Thuộc

: Tồn tại

: -lớn

: Omega

: Theta

: Phép toán giao tập hợp

: Phép toán hợp tập hợp

BT: Bài toán.

# Chương 2

# TỔNG QUAN VỀ GIẢI THUẬT VÀ ĐỘ PHỨC TẠP CỦA GIẢI THUẬT



## Giới thiệu

Nếu muốn trở thành một chuyên gia máy tính thì có nhiều lý do để chúng ta học giải thuật.

Ở quan điểm thực tiễn, chúng ta phải biết được một số giải thuật cơ bản từ các lĩnh vực khác nhau của máy tính; ngoài ra chúng ta có thể thiết kế giải thuật mới và phân tích độ phức tạp của chúng. Ở quan điểm lý thuyết, nghiên cứu các giải thuật là nền tảng của khoa học máy tính.

Kể cả khi không phải là một sinh viên công nghệ thông tin, cũng có nhiều lý do thuyết phục để nghiên cứu giải thuật. Nói thẳng ra, chương trình máy tính sẽ không tồn tại nếu không có các giải thuật. Ngày nay, các ứng dụng máy tính trở thành không thể thiếu trong hầu hết các khía cạnh trong cuộc sống của chúng ta, nghiên cứu giải thuật trở thành một điều cần thiết cho nhiều người.

Một lý do khác để nghiên cứu các giải thuật là chúng hữu dụng trong việc phát triển kỹ năng phân tích. Các giải thuật cũng có thể được xem như là dạng đặt biệt của giải quyết vấn đề. Do đó, kỹ thuật thiết kế giải thuật cụ thể có thể được giải thích như chiến lược giải quyết vấn đề.

Dĩ nhiên không phải công việc nào cũng có giải thuật để giải quyết. Ví dụ, chúng ta sẽ không tìm thấy một giải thuật để tìm một cuộc sống hạnh phúc hoặc giải thuật để trở thành người giàu có, người nổi tiếng.

Chương này chúng tôi sẽ giới thiệu về các tính chất cơ bản của giải thuật, một số kỹ thuật cơ bản để thiết kế giải thuật cũng như các phương pháp để tính toán độ phức tập của giải thuật.

## Giải thuật và các tính chất của giải thuật

Định nghĩa giải thuật

*Giải thuật* hay *thuật toán* (algorithm) là một dãy hữu hạn các thao tác được sắp xếp theo một trình tự xác định sao cho sau khi thực hiện dãy thao tác ấy, từ Input (dữ liệu đầu vào) của bài toán, ta nhận được Output (dữ liệu đầu ra hay kết quả) cần tìm. ([15])

Nói cách khác, giải thuật là một bộ các qui tắc hay qui trình cụ thể nhằm giải quyết một vấn đề trong một số bước hữu hạn, hoặc nhằm cung cấp một kết quả từ một tập hợp của các dữ kiện đưa vào ([18]).

Ví dụ ‑

Giải thuật để giải phương trình bậc nhất , (a, b, c là các số thực), trong tập hợp các số thực có thể là một bộ các bước sau đây:

* Nếu

thì có nghiệm bất kì

thì vô nghiệm

* Nếu

có duy nhất một nghiệm

Khi một giải thuật đã hình thành thì ta không xét đến việc chứng minh giải thuật đó mà chỉ chú trọng đến việc áp dụng các bước theo sự hướng dẫn sẽ có kết quả đúng. Việc chứng minh tính đầy đủ và tính đúng của các giải thuật phải được tiến hành xong trước khi có giải thuật. Nói rõ hơn, giải thuật có thể chỉ là việc áp dụng các công thức hay qui tắc, qui trình đã được công nhận là đúng hay đã được chứng minh về mặt toán học.

Giải thuật hiện nay thường được dùng để chỉ giải thuật giải quyết các vấn đề tin học. Hầu hết các giải thuật tin học đều có thể viết thành các chương trình máy tính mặc dù chúng thường có một vài hạn chế (vì khả năng của máy tính và khả năng của người lập trình). Trong nhiều trường hợp, một chương trình khi thiết kế bị thất bại là do lỗi ở các giải thuật mà người lập trình đưa vào là không chính xác, không đầy đủ, hay không ước định được trọn vẹn lời giải của vấn đề. Tuy nhiên cũng có một số bài toán mà hiện nay người ta chưa tìm được lời giải triệt để, những bài toán ấy gọi là những bài toán NP-không đầy đủ ([18]).

Trong ngành khoa học máy tính, thì giải thuật là được thể hiện thông qua một chương trình máy tính (hay một tập hợp các chương trình máy tính) được thiết kế để giải quyết một số loại vấn đề một cách có hệ thống. Một thí dụ kinh điển trong ngành khoa học máy tính là giải thuật đệ quy dùng để giải bài toán tháp Hà Nội.

Tính chất của giải thuật

Một giải thuật có các tính chất sau ([1],[2],[15]):

* Tính chính xác: để đảm bảo kết quả tính toán hay các thao tác mà máy tính thực hiện được là chính xác.
* Tính rõ ràng: Giải thuật phải được thể hiện bằng các câu lệnh minh bạch; các câu lệnh được sắp xếp theo thứ tự nhất định.
* Tính khách quan: Một giải thuật dù được viết bởi nhiều người trên nhiều máy tính vẫn phải cho kết quả như nhau.
* Tính phổ dụng: Giải thuật không chỉ áp dụng cho một bài toán nhất định mà có thể áp dụng cho một lớp các bài toán có đầu vào tương tự nhau.
* Tính kết thúc: Giải thuật phải gồm một số hữu hạn các bước tính toán.

## Một số kỹ thuật cơ bản để thiết kế giải thuật

2. 3. 1. Chiến lược trực tiếp (Brute force)

Các đặc trưng cơ bản của chiến lược trực tiếp:

Chiến lược trực tiếp một phương pháp đơn giản để giải quyết vấn đề, dựa trên định nghĩa và các khái niệm liên quan của bài toán ([1]).  
Là chiến lược dễ dàng áp dụng và được lựa chọn đầu tiên trong quá trình phân tích thiêt kế giải thuật

Chiến lược trực tiếp được áp dụng cho một lớp rất rộng các bài toán

Chi phí thiết kế rẻ, nên chiến lược trực tiếp thích hợp cho các bài toán kích thước nhỏ

Chiến lược trực tiếp có thể sinh ra một số giải thuật có độ phức tạp khá lớn (hoặc rất lớn)

Chiến lược trực tiếp là cơ sở để đề xuất các giải thuật mới

*Chiến lược vét cạn* (Exhaustive) là trường hợp đặc biệt của *chiến lược trực tiếp* (Brute force)

Ví dụ ‑

Bài toán tính tổng

Chiến lược thiết kế giải thuật trực tiếp sử dụng kỹ thuật cộng dồn các bình phương của các số liên tiếp

Giải thuật sử dụng một vòng lặp:

|  |
| --- |
| ALGORITHM\_SUM(n)  1 S ←0  2 for i ←1 to n do  3 S ←S+i\*i  4 returnS |

* + 1. Chiến lược chia để trị

Các đặc trưng cơ bản của chiến lược chia để trị:

Là chiến lược thiết kế được áp dụng rất rộng rãi trong giải toán nói chung và thiết kế giải thuật nói riêng ([1])

Giải thuật giải một bài toán dựa trên chiến lược chia để trị được phát triển theo các bước:

* Chia bài toán thành các bài toán con, thường là cùng kiểu
* Giải các bài toán con (thường dùng kỹ thuật đệ qui)
* Kết hợp lời giải các bài toán con để có lời giải bài toán ban đầu

Bài toán kích thước n

BT con 1

kích thước n/2

BT con 2

kích thước n/2

Sơ đồ 2 BT con

Giải BT con 2

Giải BT con 1

Kết hợp lời giải

các BT con

Hình 2.3.1 Sơ đồ 2 BT con

Ví dụ ‑

Cho một mảng các số được sắp theo thứ tự tăng và một số , tìm trong mảng số có giá trị bằng

Giải thuật tìm kiếm dựa trên chiến lược chia để trị như sau

* Chia mảng thành 2 phần (thường chọn điểm chia m ở giữa mảng)
* Nếu thì kết thúc tìm kiếm
* Nếu tìm trong mảng con bên trái (giải bài toán con)
* Nếu tìm trong mảng con bên phải
  + 1. Chiến lược tham ăn

Các đặc trưng cơ bản của chiến lược tham ăn:

Chỉ sử dụng để giải các bài toán tối ưu

Giải thuật được thiết kế bằng chiến lược tham ăn (greedy) giải các bài toán con trước khi giải bài toán gốc

Quá trình tìm lời giải được thực hiện thông qua một dãy các bước để tìm lời giải (nghiệm) tối ưu cục bộ (lời giải các bài toán con) cho đến khi tìm thấy lời giải bài toán gốc

Mỗi bước tìm nghiệm cục bộ thỏa mãn ba tính chất sau

* Phải thỏa mãn ràng buộc bài toán (feasible)
* Tối ưu cục bộ (locally optimal)
* Không thay đổi nghiệm cục bộ trong các bước kế tiếp

Chiến lược greedy luôn thực hiện một “lựa chọn” tốt nhất hiện tại khi tìm kiếm lời giải bài toán con mà không quan tâm đến bước tiếp theo

Chiến lược gready không giải tất cả các bài toán con (tối ưu) như qui hoạch động mà mỗi bước chỉ tìm lời giải tối ưu trong một tập các bài toán con ([1])

* + 1. Chiến lược biến đổi để trị

Các đặc trưng cơ bản của chiến lược biến đổi để trị ([1]):

Chiến lược biến đổi để trị (transform-and-conquer) gồm hai tầng (giai đoạn):

* Biến đổi thể hiện (instance-input) của bài toán để có thể dễ giải (hoặc giải hiệu quả) hơn
* Giải bài toán trên thể hiện (input) đã được biến đổi

Có ba kiểu biến đổi

* Biến đổi về thể hiện đơn giản hoặc thuận tiện hơn của cùng bài toán, gọi là (instance simplification)
* Biến đổi về một dạng biểu diễn khác của cùng thể hiện bài toán, gọi là (representation change)
* Biến đổi về một thể hiện của một bài toán khác đã có giải thuật, gọi là (problem reduction)

Ví dụ ‑

Bài toán kiểm tra tính duy nhất của một phần tử

Nếu dùng chiến lược trực tiếp sẽ có giải thuật

Biến đổi để trị bằng cách sắp xếp lại danh sách tăng dần (biến đổi đầu vào của bài toán) sau đó thực hiện việc kiểm tra tính duy nhất trên danh sách được sắp

Giải thuật thực hiện chỉ một vòng lặp

|  |
| --- |
| ALGORITHM PRESORTELEMENTUNIQUENESS(A[0..n − 1])  1 Sort the array A  2 for i ←0 to n − 2 do  3 if A[i]= A[i + 1] return false  4 return true |

* + 1. Chiến lược quy hoạch động

Các đặc trưng cơ bản của chiến lược quy hoạch động ([1]):

Giải một bài toán bằng cách giải các bài toán con kích thước nhỏ hơn

Nghiệm của bài toán có được bằng cách kết hợp nghiệm của các bài toán con

Nghiệm (lời giải) bài toán ban đầu và các bài toán con được biểu diễn bởi một hệ thức truy hồi

Sử dụng chiến lược qui hoạch động để giải một bài toán được chia làm hai bước:

* Mô hình hóa lời giải bài toán bằng một hệ thức truy hồi
* Giải hệ thức truy hồi bằng một giải thuật hiệu quả (từ dưới lên)

Chiến lược qui hoạch động phát triển giải thuật giải bài toán bằng cách giải các bài toán con từ dưới lên (bottom up)

Kết quả các bài toán con được lưu trữ trong một bảng và được sử dụng để giải bài toán đã cho

Lưu ý:

* Nếu giải hệ thức truy hồi bằng đệ qui (top down) thì độ phức tạp có thể rất lớn
* Chiến lược qui hoạch động giải bài toán từ dưới lên (bottom up) với giải thuật hiệu quả
* Qui hoạch động thường được ứng dụng để giải các bài toán tối ưu

Ví dụ ‑

Bài toán tính số fibonacci thứ n

Cho dãy số Fibonacci được biểu diễn bởi hệ thức truy hồi

và Hãy tính số hạng thứ n trong dãy số fibonacci

Các bài toán tính … là các bài toán con của bài toán tính

Thay vì dùng chiến lược trực tiếp (giải thuật đệ qui), giải thuật qui hoạch động tính

Số hạng được tính qua và đã cho, số được tính qua và ,…

Giải thuật cần một vòng lặp để tính

|  |
| --- |
| FIBONACCI(n)  1 f[0] ←0  2 f[1] ←1  3 for i ←2 to n  4 do f[i] ←f[i-1] +f[i-2]  5 return f[n] |

* + 1. Chiến lược quay lui

Các đặc trưng cơ bản của chiến lược quay lui :

Giải thuật được thiết kế bằng chiến lược quay lui biểu diễn nghiệm bài toán như một vector với được chọn trong tập nào đó

Quá trình tìm được thực hiện bằng cách tính toán mỗi thành phần tại một thời điểm

Tại bước thứ i, vector con tìm được gọi là một nghiệm bộ phận của bài toán

Để tìm thành phần thứ k của nghiệm (thỏa một số ràng buộcnào đó), khi đã chọn được thành phần của , giải thuật chọn (tính) trong số các khả năng có thể có của nó trong

Với mỗi khả năng j, giải thuật kiểm tra xem có chấp nhận đươc không:

Nếu chấp nhận thì xác định theo j, khi thì có một lời giải, ngược thì tiếp tục xác định (thường bằng đệ qui)

Nếu thử tất cả các khả năng mà không có khả năng nào được chấp nhận thì quay lui lại bước trước để xác định lại ([1]).

* + 1. Chiến lược nhánh cận

Các đặc trưng cơ bản của chiến lược nhánh cận:

Chiến lược quay lui cho phép tìm được nghiệm đúng của bài toán, nhưng do phải thử mọi khả năng, khi kích thước bài toán lớn sẽ rất kém hiệu quả

Chiến lược nhánh cận khắc phục được hạn chế này bằng cách, xác định nhánh cận với mục tiêu tại mỗi bước tìm kiếm, vì vậy loại bỏ được hầu hết các hướng tìm kiếm

không cần thiết (trong cây không gian tìm kiếm).

Chiến lược nhánh cận thường được áp dụng để giải các bài toán tối ưu

Chiến lược nhánh cận dựa trên chiến lược quay lui và một hàm lượng giá mục tiêu hướng đến các nhánh cận mục tiêu để tìm lời giải nhanh nhất có thể. ([1])

## Độ phức tạp của giải thuật

Độ phức tạp của giải thuật là chi phí về tài nguyên của hệ thống (chủ yếu là thời gian, bộ nhớ, bộ xử lí, đường truyền) cần thiết để thực hiện giải thuật. ([1] [18])

Thời gian mà máy tính khi thực hiện một giải thuật không chỉ phụ thuộc vào bản thân giải thuật đó, ngoài ra còn tùy thuộc từng máy tính. Để đánh giá hiệu quả của một giải thuật, có thể xét số các phép tính phải thực hiện khi thực hiện giải thuật này. Thông thường số các phép tính được thực hiện phụ thuộc vào cỡ của bài toán, tức là độ lớn của đầu vào. Vì thế độ phức tạp giải thuật là một hàm phụ thuộc đầu vào. Tuy nhiên trong những ứng dụng thực tiễn, chúng ta không cần biết chính xác hàm này mà chỉ cần biết một ước lượng đủ tốt của chúng.

Để ước lượng độ phức tạp của một giải thuật ta thường dùng khái niệm bậc -lớn và bậc (bậc Theta).

## Phân tích giải thuật

Một số thuật ngữ và khái niệm ([1],[17][17])

* *Phân tích giải thuật* (Analyzing of Algorithm) là quá trình tìm ra những đánh giá về tài nguyên cần thiết để thực hiện giải thuật.
* Thời gian tối thiểu để thực hiện giải thuật với kích thước đầu vào n gọi là *thời gian chạy tốt nhất* (best-case) của giải thuật
* Thời gian nhiều nhất để thực hiện giải thuật với kích thước đầu vào n được gọi là *thời gian chạy xấu nhất* (worst-case) của giải thuật
* Thời gian trung bình để thực hiện giải thuật với kích thước đầu vào n được gọi là *thời gian chạy trung bình* (average case) của giải thuật.

Giả sử là hàm không âm đối số nguyên dương n

* có tốc độ tăng (bậc) không quá và viết nếu có hằng số và số nguyên dương N sao cho
* có tốc độ tăng ít nhất là và viết nếu có hằng số và số nguyên dương N sao cho
* có tốc độ tăng là và viết nếu và ([1])

Nếu và thì

Nếu giải thuật có thời gian chạy tốt nhất (trung bình, xấu nhất) là và thì ta nói thời gian chạy tốt nhất (trung bình, xấu nhất) của giải thuật có bậc (tốc độ tăng) không quá hay thời gian chạy tốt nhất (trung bình, xấu nhất) của giải thuật là .

Bậc (tốc độ tăng) của thời gian chạy càng lớn thì giải thuật càng chậm(chẳng hạn, giải thuật có thời gian chạy hiệu quả hơn giải thuật có thời gian chạy ([1])

Khi đánh giá độ phức tạp theo *O-*lớn, ta bỏ qua lệnh gán chỉ số vòng lặp.

Ví dụ ‑

Phân tích giải thuật tính gần đúng

|  |
| --- |
| Exp(x, n)  1 s 🡨1  2 for i 🡨1 to n  3 do p 🡨1  4 for j 🡨1 to i  5 do p 🡨p\*x/j  6 s 🡨s + p  7 return s |

Gọi lần lượt là thời gian thực hiện lệnh gán, trả về, thì

Độ phức tạp (tốc độ tăng) biểu diễn theo O, , phụ thuộc vào số lần thực hiện của lệnh (phép toán) được thực thi nhiều lần nhất và có có chiphí lớn nhất trong giải thuật. Các lệnh (phép toán) được thực thi nhiều lần nhất và có có chi phí

lớn nhất trong giải thuật gọi là lệnh (phép toán) cơ bản (basic operation). Vì vậy, có thể tính độ phức tạp đơn giản bằng cách chỉ xác định và đếm số lệnh cơ bản trong giải thuật ([1],[17])

Ví dụ 2.5.2:

|  |
| --- |
| Exp(, n)  1 s 🡨1  2 p 🡨1  3 for i 🡨1 ton  4 do p 🡨 // phép toán cơ bản là phép nhân  5 s 🡨s + p  6 return s |

Vậy: , trong đó là thời gian thực hiện .

## Một số công cụ toán học hỗ trợ phân tích độ phức tạp của giải thuật

Hệ thức truy hồi

Hệ thức truy hồi đối với dãy là một phương trình có dạng

Ví dụ ‑

Dãy được gọi là nghiệm của hệ thức truy hồi nếu các số hạng của nó thỏa mãn hệ thức truy hồi này

Các điều kiện đầu là các giá trị của các số hạng của dãy đi trước số hạng đầu tiên kể từ đó hệ thức truy hồi có hiệu lực.

Hệ thức truy hồi với điều kiện đầu xác định một dãy (nghiệm) duy nhất của nó. Điều kiện đầu và hệ thức truy hồi cung cấp một định nghĩa đệ qui cho một dãy.

Hệ thức truy hồi tuyến tính thuần nhất

Một hệ thức truy hồi tuyến tính thuần nhất bậc ([1],[17],[18]) là một hệ thức dạng

Với là các hằng số thực,

Ví dụ ‑

Hệ thức Fibonaci là hệ thức truy hồi tuyến tính thuần nhất bậc 2.

Giải hệ thức truy hồi tuyến tính thuần nhất

Dãy là nghiệm của

Phương trình (2) được gọi là phương trình đặc trưng của (1), và nghiệm của nó được gọi là nghiệm đặc trưng

Định lý ‑

Cho là các hằng số. Giả sử có hai nghiệm phân biệt . Khi đó dãy {} là nghiệm của hệ thức truy hồi nếu và chỉ nếu với n =0,1,2,… Trong đó là các hằng số.

Định lý ‑

Cho là các hằng số. Giả sử chí có một nghiệm kép . Khi đó dãy {} là nghiệm của hệ thức truy hồi nếu và chỉ nếu

với n =0,1,2,… Trong đó là các hằng số.

Định lý ‑

Cho là các hằng số. Giả sử phương trình đặc trưng chí có k nghiệm phân biệt. Khi đó dãy {} là nghiệm của hệ thức truy hồi nếu và chỉ nếu

với n =0,1,2,… Trong đó là các hằng số.

Hệ thức truy hồi tuyến tính không thuần nhất

Một hệ thức truy hồi tuyến tính không thuần nhất bậc là một hệ thức dạng

Với là các hằng số thực, , là một hàm chỉ phụ thuộc vào n

Ví dụ ‑

Hệ thức là hệ thức truy hồi tuyến tính không thuần nhất bậc 1

Định lý ‑

Nếu là một nghiệm của hệ thức truy hồi tuyến tính không thuần nhất , thì mọi nghiệm của nó có dạng , trong đó là nghiệm của hệ thức truy hồi tuyến tính kết hợp .

Định lý ‑

Giả sử thỏa mãn hệ thức truy hồi tuyến tính không thuần nhất , với

Nếu s không phải là nghiệm của PT đặc trưng của hệ thức truy hồi kết hợp thì (1) có một nghiệm dạng

Nếu s là một nghiệm bội m của PT đặc trưng của hệ thức truy hồi kết hợp thì (1) có một nghiệm dạng

Hệ thức truy hồi chia để trị

Hệ thức truy hồi dạng , trong đó a, b là các nguyên dương, được gọi là hệ thức truy hồi chia để trị

Ví dụ ‑

là một hệ thức truy hồi chia để trị

Định lý ‑

Giả sử là một hàm tăng thỏa mãn hệ thức truy hồi khi n là số nguyên lớn hơn 1, c là một số thực dương , thì

Định lý ‑

Giả sử là một hàm tang thỏa mãn hệ thức truy hồi

khi n= là số nguyên lớn hơn 1, c và d là một số thực dương, thì

Một số công cụ toán khác

* Phần nguyên sàn (floor), phần nguyên trần (ceiling) của số thực , ký hiệu lần lượt và ,là các số nguyên thỏa mãn
* n là số tự nhiên và a, b, c là các số thực dương thì
* Với mọi thì
* Với mọi thì

( gọi là số điều hòa thứ n - Harmonic Number)

* 3. 1. Định nghĩa xác suất



Định nghĩa xác suất cổ điển

Giả sử là một phép thử có n biến cố đồng khả năng xuất hiện.

A là một biến cố trong phép thử, có m biến cố đồng khả năng thuận lợi cho biến cố A. Thì tỷ số được gọi là xác suất của biến cố A .

Định nghĩa này dựa trên hai điều kiện:

* Các biến cố trong phép thử phải đồng khả năng.
* Số các biến cố đồng khả năng khi thực hiện phép thử phải hữu hạn ([4],[18]).

Ví dụ ‑

Một số điện thoại gồm 8 chữ số, bắt đầu bằng các chữ số 38393…..., ba chữ số cuối bị xóa nhòa.

Tính xác suất để ba chữ số bị xóa

a) Khác nhau và khác các chữ số đầu

b) Trùng nhau và khác các chữ số đầu

Định nghĩa xác suất theo tần suất

Tiến hành phép thử T nhiều lần trong cùng điều kiện .

Nếu trong n lần thực hiện phép thử T có k lần xuất hiện biến cố A, thì tỷ số: được gọi là tần suất xuất hiện A ([4],[18]).

Khi số phép thử tăng lên vô hạn thì tần suất dao động chung quanh một giá trị ổn

định, giá trị đó được gọi là xác suất của biến cố A.

Trong thực tế khi số phép thử lớn, thì được tính xấp xỉ bởi tần suất .

Ví dụ ‑

Tại một địa phương khảo sát ngẫu nhiên 1000 em bé chào đời trong năm 2008, thì có 560 bé trai.

Gọi A là biến cố sinh trai tại địa phương

Thì tần suất sinh trai là

* + 1. Công thức cộng xác suất ([4],[18])

Trường hợp các biến cố xung khắc

* A và B là hai biến cố xung khắc
* xung khắc từng đôi

Trường hợp tổng quát

* A và B là hai biến cố bất kỳ
* A, B, C là ba biến cố bất kỳ
* là các biến cố bất kỳ

Ví dụ ‑

Khảo sát 100 thí sinh nộp đơn dự thi vào đại học có 70 thí sinh nộp đơn dự thi vào khối A, 50 thí sinh nộp đơn dự thi vào khối B, 35 thí sinh nộp đơn dự thi cả hai khối A và B. Chọn ngẫu nhiên một thí sinh trong số 100 thí sinh trên. Tính xác suất thí sinh này nộp đơn dự thi

a) Vào ít nhất một khối trên

b) Không phải hai khối trên

GIẢI:

Đặt A: thí sinh dự thi vào khối A

B: thí sinh dự thi vào khối B

Xác suất có điều kiện-công thức nhân

Xác suất có điều kiện

Giả sử có phép thử T, A và B là hai biến cố trong cùng phép thử. Xác suất của biến cố A

được tính trong trường hợp biết rằng biến cố B đã xảy ra được gọi là xác suất của A với điều kiện B ([4],[18]).

Ký hiệu:

Công thức nhân xác xuất

* A và B là hai biến cố bất kỳ, ta có:
* là n biến cố bất kỳ, ta có:

Ví dụ ‑

Khoa T được phân phối 3 lô đất, ưu tiên cho 3 nhân viên chưa có nhà riêng, nhưng có 5 nhân viên chưa có nhà riêng, điều kiện công tác tốt đều như nhau. Tổ chức bắt thăm, trong 5 lá thăm có 3 lá thăm có dấu X, 2 lá thăm có dấu O. Năm người lần lượt bắt thăm, ai bắt được thăm có dấu X thì được nhận một lô đất.Theo anh chị bắt thăm trước hay sau có lợi thế hơn.

GIẢI:

Đặt : người bắt thăm ở lần thứ i được lô đất,

Tương tự ta tính được

Kết luận: sự bắt thăm là công bằng, bắt thăm trước hay sau đều như nhau.

* + 1. Các biến cố độc lập

Hai biến cố A và B được gọi là độc lập với nhau nếu biến cố A có xảy ra hay không xảy ra không ảnh hưởng đến sự xảy ra hay không xảy ra của biến cố B và ngược lại ([4]).

Nếu A và B độc lập thì:

độc lập

độc lập

độc lập

Độc lập toàn thể

Các biến cố được gọi là độc lập với nhau (độc lập toàn thể) nếu việc xảy ra hay không xảy ra của một nhóm bất kỳ k biến cố ( ) không có ảnh hưởng đến việc xảy ra hay không xảy ra các biến cố còn lại ([4]).

Nếu độc lập thì:

CHÚ Ý:

* độc lập thì độc lập từng đôi
* độc lập từng đôi thì chưa chắc độc lập
  + 1. Công thức xác suất đầy đủ

Giả sử có một phép thử T

Các biến là một hệ đầy đủ các biến cố. F là một biến cố của phép thử.

Ta có:

([18])

Ví dụ ‑

Sinh viên K34. ĐHKT dự thi môn Xác suất thống kê, trong đó có 30% là nữ. Xác suất đậu của sinh viên nữ là 80%, xác suất đậu của sinh viên nam là 75%. Chọn ngẫu nhiên một sinh viên K34. Tính xác suất sinh viên này đậu môn Xác suất thống kê.

GIẢI:

Đặt F: sinh viên được chọn đậu môn XSTK .

M: sinh viên được chọn là nữ

N: sinh viên được chọn là nam

Nhận xét:

ca là một hệ đầy đủ các biến cố

Áp dụng công thức xác suất đầy đủ, ta có:

* + 1. Sự phân bố xác xuất

Một phân bố xác suất ([18][18]) là một hàm số nhằm gán các giá trị (gọi là xác suất) cho các sự kiện. Các giá trị số này đặc trưng cho khả năng xảy ra của các sự kiện. Với một tập bất kì các sự kiện, có rất nhiều cách để gán các xác suất, và thường dựa vào sự lựa chọn loại phân bố của các sự kiện đang xem xét.

Có nhiều cách để chỉ định một phân bố xác suất. Thông thường nhất có lẽ là chỉ định một *hàm mật độ xác suất* (probability density function). Từ đó, xác suất của một sự kiện sẽ được bằng cách lấy tích phân hàm mật độ. Tuy nhiên, hàm phân bố cũng có thể được chỉ định rõ trực tiếp. Trong trường hợp chỉ có một biến (hay một chiều), thì hàm phân bố được gọi là *hàm phân bố tích lũy* (cumulative distribution function). Phân bố xác suất cũng có thể được chỉ định thông qua các giá trị mômen hay *hàm đặc trưng* (characteristic function), hay các cách khác nữa.

Một phân bố được gọi là phân bố rời rạc nếu nó được định ra trên một tập rời rạc, đếm được.

Một phân bố được gọi là phân bố liên tục nếu nó được định ra trên một tập vô hạn, không đếm được.

Hầu hết các phân bố trong các ứng dụng thực tế đều hoặc là một trong hai, nhưng có một số ví dụ về phân bố bao gồm của cả 2, gọi là phân bố hỗn hợp.

Các phân bố rời rạc quan trọng bao gồm phân bố đồng nhất, phân bố Poisson, phân bố nhị thức, phân bố nhị thức âm và phân bố Maxwell-Boltzmann.

Các phân bố liên tục quan trọng bao gồm phân bố chuẩn (hay còn gọi là phân bố Gauss), phân bố gamma, *phân bố-t của Student* (Student's t-distribution), và *phân bố hàm mũ* (exponential distribution).

* + 1. Kỳ vọng

Trong Lý thuyết xác suất, giá trị kỳ vọng, giá trị mong đợi, hoặc trung bình của một biến ngẫu nhiên là trung bình có trọng số của tất cả các giá trị của thể của biến đó, hay là được tính bằng tổng các tích giữa xác suất xảy ra của mỗi giá trị có thể của biến với giá trị đó. Như vậy, nó biểu diễn giá trị trung bình mà người ta "mong đợi" thắng cược nếu đặt cược liên tục nhiều lần với khả năng thắng cược là như nhau. Lưu ý rằng bản thân giá trị đó có thể không được mong đợi theo nghĩa thông thường; nó có thể ít có khả năng xảy ra hoặc không thể xảy ra. ([18])

## Kết luận

Với phần tổng quan về giải thuật được chúng tôi trình bày trên đây, hy vọng quý độc giả có thể hiểu một cách khái quát về giải thuật và phân tích giải thuật, để có thể hiểu được phần trình bày ở các chương sau. Ở chương kế tiếp chúng tôi sẽ trình bày tổng quan về giải thuật ngẫu nhiên.

# Chương 3

# GIẢI THUẬT NGẪU NHIÊN



## Giới thiệu

Khái niệm các giải thuật ngẫu nhiên tương đối mới. Trong mọi giải thuật được giới thiệu từ trước đến nay, mỗi bước trong giải thuật đều được xác định. Với giải thuật ngẫu nhiên thì chúng ta không xác định được, trong quá trình thực hiện một giải thuật ngẫu nhiên nó sẽ đưa ra một chọn lựa tùy ý. Một số hành động được thực hiện một cách ngẫu nhiên.

**Giải thuật xác định:**

**INPUT**

**ALGORITHM**

**OUTPUT**

Mục tiêu: chứng minh giải thuật luôn giải quyết vấn đề một cách chính xác và nhanh (thông thường, số lượng bước là đa thức dựa trên kích thước đầu vào).

**Giải thuật ngẫu nhiên:**

**INPUT**

**ALGORITHM**

**OUTPUT**

**RANDOM NUMBERS**

Ngoài đầu vào, giải thuật còn dùng một nguồn số ngẫu nhiên và thực hiện các chọn lựa ngẫu nhiên trong quá trình thực hiện. Kết quả có thể khác nhau ngay cả khi thực hiện trên một đầu vào nhất định.

## Định nghĩa và ví dụ giải thuật ngẫu nhiên

Định nghĩa

Giải thuật ngẫu nhiên là một giải thuật mà nó không chỉ nhận dữ liệu đầu vào*input* mà còn một dãy *bit* được dùng nhằm mục đích đưa ra các chọn lựa ngẫu nhiên. Ngay cả đối với đầu vào là cố định, những lần chạy khác nhau của giải thuật ngẫu nhiên thì đưa ra các kết quả khác nhau; do đó một mô tả cho các thuộc tính của giải thuật ngẫu nhiên là sẽ liên quan đến các phát biểu về xác suất. Ví dụ, ngay khi đầu vào là cố định, thời gian thực hiện một giải thuật ngẫu nhiên là một giá trị ngẫu nhiên.([3])

Ví dụ ‑

**Input:** Một dãy phần tử, trong đó một nửa là *a* và nửa còn lại là *b*.

**Ouput:** Tìm *a* trong mảng.

Chúng tôi đưa ra hai phiên bản của giải thuật, một là **giải thuật Las Vegas** và một là **giải thuật Monte Carlo**.

Giải thuật Las Vegas:

|  |
| --- |
| FINDINGA\_LV(ARRAY A,N)  begin  repeat  Randomly select one element out of n elements.  until ‘a’ is found  end |

Giải thuật này thành công với xác suất là 1. Thời gian chạy là ngẫu nhiên (và lớn tùy ý) nhưng sự mong chờ của nó thì cận trên là . ([17])

Giải thuật Monte Carlo:

|  |
| --- |
| FINDINGA\_MC(ARRAY A, N, K)  begin  i=1  repeat  Randomly select one element out of n elements.  i=i+1  until i=k or ‘a’ is found  end |

Nếu *a* được tìm thấy, giải thuật thành công, không thì giải thuật thất bại. Sau khi lặp lại *k* lần, xác suất tìm thấy *a* là:

Giải thuật này không bảo đảm thành công nhưng thời gian chạy ổn định. Sự chọn lựa thì được thực hiện chính xác *k* lần, do dó thời gian chạy là . ([17])

## Phân loại các giải thuật ngẫu nhiên

Người ta phải phân biệt giữa các giải thuật mà chúng sử dụng đầu vào ngẫu nhiên để giảm thời gian chạy được mong đợi hoặc bộ nhớ sử dụng (thời gian hay giá thành), nhưng luôn chấm dứt với một kết quả đúng trong một giới hạn thời gian, và các giải thuật xác suất, mà phụ thuộc vào đầu vào ngẫu nhiên, có khả năng phát sinh ra kết quả sai *(giải thuật Monte Carlo)* hoặc thất bại trong việc tìm ra kết quả *(giải thuật Las Vegas)*hoặc bằng việc báo hiệu sự thất bại hoặc không thể chấm dứt.

## Giải thuật Monte Carlo

Giải thuật Monte Carlo là một giải thuật ngẫu nhiên luôn chấm dứt trong thời gian xác định bằng đa thức, nhưng có thể tạo ra kết quả sai.

*(Không được nhầm lẫn với phương pháp Monte Carlo)*

Trong lĩnh vực tính toán, một giải thuật Monte Carlo là một giải thuật ngẫu nhiên mà có số lần chạy xác định, nhưng đầu ra của nó có thể không chính xác với xác suất nhất định (thường là nhỏ).

Một lớp có liên quan của giải thuật Las Vegas cũng là ngẫu nhiên, nhưng bằng cách khác chúng có lượng thời gian ngẫu nhiên, nhưng đưa ra câu trả lời chính xác. Một giải thuật Monte Carlo có thể được chuyển thành giải thuật Las Vegas khi mà tồn tại một thủ tục để xác định output được sinh ra bởi giải thuật chắc chắn là chính xác.

## Giải thuật Las Vegas

Giải thuậy Las Vegas là giải thuật ngẫu nhiên luôn đưa ra kết quả đúng. Nhưng thời gian chạy ở mỗi lần thực hiện có thể khác nhau.

Trong lĩnh vực tính toán, một giải thuật Las Vegas là một giải thuật ngẫu nhiên, mà nó luôn đưa ra các kết quả chính xác; nghĩa là, luôn sinh ra kết quả chính xác hoặc nó thông báo cho biết thất bại. Nói cách khác giải thuật Las Vegas không đánh cược với tính chân thực của kết quả; nó đánh cược với các nguồn được dùng để tính toán. Một ví dụ đơn giản là giải thuật *randomized quicksort*, có chốt được chọn ngẫu nhiên, nhưng kết quả thì luôn là được sắp xếp. Định nghĩa thông thường của giải thuật Las Vegas gồm là sự hạn chế tức thời gian chạy dự kiến là hữu hạn, khi sự kì vọng được thực hiện trên không thông tin ngẫu nhiên, hoặc entrôpi, được dùng trong giải thuật.

Giải thuật Las Vegas được giới thiệu bởi Lászlo Babai năm 1979, trong phạm vi vấn đề đẳng cấu đồ thị, như một phiên bản vững chắc hơn giải thuật Monte Carlo. Giải thuật Las Vegas có thể được dùng trong các tình huống mà số lượng các giải pháp khả thi thì tương đối hạn chế, và xác định sự chính xác của các giải pháp đề cử thì tương đối dễ trong khi việc tính toán các giải pháp thì thực sự phức tạp

Tên của giải thuật dựa theo tên thành phố Las Vegas, Nevada – nơi đây được biết đến là một biểu tượng bài bạc của Mỹ.

## Các lĩnh vực áp dụng của giải thuật ngẫu nhiên ([5])

Giải thuật ngẫu nhiên và xác suất đóng vai trò quan trọng trong khoa học máy tính, với những ứng dụng từ tối ưu tổ hợp và máy tính học cách giao tiếp với mạng và giao thức bảo mật.

Ngày nay có thể nhật ra rằng, trong phạm vi ứng dụng rộng lớn, sự ngẫu nhiên hóa là một công cụ vô cùng quan trọng cho các giải thuật xây dựng.

* Lý thuyết số: kiểm tra số nguyên tố (Monte Carlo).
* Cấu trúc dữ liệu: sắp xếp, thống kê theo yêu cầu, tìm kiếm, tính toán trong hình học.
* Nhận dạng đại số: xác định tính đồng nhất của ma trận và đa thức, hệ thống chứng minh tương tác.
* Lập trình toán: các giải thuật nhanh hơn cho lập trình tuyến tính, làm tròn số.
* Các giải thuật đồ thị: các giải thuật tìm cây bao trùm nhỏ nhất, đường đi ngắn nhất, lát cắt nhỏ nhất.
* Đếm và bảng đếm: ma trận vĩnh cửu, giải tích tổ hợp.
* Tính toán song song và phân phối
* Chứng minh sự tồn tại có xác suất: đưa ra một tổ hợp đối tượng phát sinh với xác suất khác 0 giữa các đối tượng được vẽ ra từ một không gian xác suất phù hợp.
* Loại bỏ sự ngẫu nhiên: đầu tiên đưa ra một giải thuật ngẫu nhiên sau đó cho rằng nó có thể được loại bỏ sự ngẫu nhiên và tạo ra một giải thuật xác định.

## Kết luận

Qua phần giới thiệu về giải thuật ngẫu nhiên, chúng tôi xin nhấn mạnh rằng: đối với nhiều bài toán, giải thuật ngẫu nhiên là đơn giản nhất, hoặc nhanh nhất, hoặc vừa đơn giản nhất vừa nhanh nhất. Tuy nhiên, nó có thể là một nhiệm vụ phức tạp để chứng minh rằng giải thuật ngẫu nhiên xác định này có thuộc tính được mong muốn.

Ở chương kế tiếp, chúng tôi xin trình bày giải thuật ngẫu nhiên cho một số bài toán tiêu biểu.

# Chương 4

# GIẢI THUẬT NGẪU NHIÊN CHO MỘT SỐ BÀI TOÁN



## Giới thiệu

## Bài toán sắp xếp

Giải thuật *quicksort* có độ phức tạp thuật toán theo thời gian trong trường hợp xấu nhất là , với mảng nhập có *n* số. Mặc dù độ phức tạp về thời gian trong trường hợp xấu nhất của giải thuật chậm, *quicksort* vẫn thường là sự chọn lựa thực tế nhất cho việc sắp xếp bởi hiệu quả đáng kể ở trường hợp trung bình: độ phức tạp về thời gian được mong chờ của giải thuật là

|  |
| --- |
| RANDOMIZED-PARTITION(A,p,r)  i=RANDOM(p,r)  exchange A[r] with A[i]  **return** PARTITION(A,p,r) |

Một phiên bản *quicksort* mớiđược gọi là RANDOMIZED-PARTITION thay cho hàm PARTITION:

|  |
| --- |
| RANDOM-QUICKSORT(A,p,r) |
| **if** p<r  q=RANDOMIZED-PARTITION(A,p,r)  RANDOMIZED-QUICKSORT(A,p,q-1)  RANDOMIZED-QUICKSORT(A,q+1,r) |

Phân tích độ phức tạp

## Bài toán xác định số nguyên tố

Bài toán này trình bày vấn đề xác định một số có phải là số nguyên tố hay không. Đây là một bài toán rất khó và mãi cho tới năm 2004 mới tìm được một giải thuật có thời gian chạy là đa thức. Ở đây chúng tôi sẽ giới thiệu một giải thuật ngẫu nhiên giải quyết vấn đề này.

Ý tưởng

Giải thuật này thực hiện *m* lần kiểm tra. Nếu có bất kì lần kiểm tra nào thành công, chúng ta kết luận rằng số này là hợp số, và chúng ta tin chắc rằng kết luận này là hoàn toàn đúng. Mặt khác, nếu toàn bộ *m* lần kiểm tra đều thất bại thì chúng ta kết luận số này là số nguyên tố. Tuy nhiên, với trường hợp này chúng ta không thể tin chắc đây là một kết luận đúng. Kết luận này đúng với xác suất . Vì vậy, nếu *m* lớn, thì sẽ tiến gần về 1 hơn, khi đó giải thuật cho ta một kết quả có độ tin cậy cao.

Mã giả

**Input**: Một số nguyên *N* >1, và một tham số *m*, với .

**Output**: *N* là số nguyên tố hay hợp số, với xác suất đúng là

|  |
| --- |
| KiemTraSoNguyenTo(N,m)  k=(n-1)/2  for i=0 to m do  b= randomly choose a numbers ,  uc=GCD(N,Pow(b,k)-1)  if uc>1 and uc<N then  return false  return true |

Hàm GCD là hàm tìm ước chung của *x, y*.

Hàm Pow là hàm tính *x* mũ y (

Khi thủ tục trên trả về false thì số *N* là hợp số, ngược lại nếu return true thì *N* là số nguyên tố.

Một số ví dụ

Ví dụ ‑

N=12, m=3.

## Bài toán tìm cặp điểm gần nhất

Bài toán tìm cặp điểm gần nhất có thể giải quyết bằng giải thuật chia để trị với độ phức tạp là . Trong phần này, chúng tôi sẽ chỉ ra một giải thuật ngẫu nhiên và độ phức tạp của giải thuật để giải quyết bài toán tìm cặp điểm gần nhất là .

Gọi *x1,x2,….xn* là *n* điểm trên mặt phẳng. Yêu cầu của bài toán là tìm cặp điểm *xi* và *xj* mà khoảng cách giữa *xi* và *xj*là nhỏ nhất trong toàn bộ các cặp điểm. Với phương pháp tiếp cận trực tiếp dùng để giải bài thì theo tính toán có tất cả *n(n-1)/2* khoảng cách sau đó tìm con số nhỏ nhất trong tất cả khoảng cách này.

Ý tưởng chính của giải thuật ngẫu nhiên là dựa trên sự quan sát sau: Nếu hai điểm *xi* và *x­j* có khoảng cách giữa chúng, và khoảng cách giữa chúng có vẻ như không là ngắn nhất và do đó tốt hơn hết là bỏ qua. Với ý tưởng này, giải thuật ngẫu nhiên trước hết là phân vùng các điểm thành một vài cụm theo một cách nào đó mà ở mỗi cụm có các điểm gần nhau. Sau đó chúng ta tính toán khoảng cách giữa các điểm trong cùng một cụm.

x1

x2

x3

x4

x5

x6

Nếu chúng ta chia vùng cho sáu điểm ở … thành ba cụm *S1={x1,x2}, S2={x3,x4}, S1={x5,x6}*, sau đó chúng ta chỉ tính toán ba khoảng cách đặt tên là *d(x1,x2)*, *d(x2,x4), d(x5,x6)*, sau cùng chúng ta tìm giá trị nhỏ nhất trong ba khoảng cách đó. Ngược lại, nếu chúng ta không chia các điểm thành các cụm, số khoảng cách phải tính là *(6\*5)/2=15* khoảng cách.

Dĩ nhiên, mô tả này thì hơi không thực tế bởi không có gì đảm bảo chiến lược này hoạt động. Trong thực tế, chiến lược này có vẻ giống một chiến lược chia để trị. Nhưng nó có một quá trình phân chia nhưng không có quá trình hợp nhất. Xem xét hình sau:

x1

x2

x3

x4

x5

x6

x7

Chúng ta có thể thấy cặp điểm gần nhất là *{x1,x3}*. Nhưng, chúng ta đã chia hai điểm này vào hai cụm khác nhau. Vì thế, nếu chúng ta chia không gian này thành các hình vuông với độ dài cạnh *δ*, không nhỏ hơn khoảng cách ngắn nhất, thì sau khi chúng ta tính hết các khoảng cách ở từng vùng, chúng ta phải nhân đôi độ dài cạnh và tạo ra các hình vuông lớn hơn. Các trường hợp mở rộng điển hình được minh họa ở hình

*T­1*

*T­4*

*T­3*

*T­2*

Giải thuật

**Input:** Một tập *S* gồm *n* điểm *x1, x2,…, xn* với .

**Output:** Cặp điểm gần nhất trong tập R

Bước 1: Chọn ngẫu nhiên một tập điểm *S1={xi1, xi2,…, xim}* với . Tìm cặp điểm gần nhất trong tập *S1* và gán giá trị khoảng cách giữa cặp điểm đó cho *δ*.

Bước 2: Xây dựng tập hình vuông *T* có độ dài cạnh là *δ*.

Bước 3: Xây dựng bốn tập hình vuông *T1, T2, T3 và T­­­4* bắt nguồn từ *T* bằng cách nhân đôi kích thước cạnh hình vuông thành 2*δ*.

Bước 4: Đối với mỗi *Ti*, xác định các tập điểm là giao giữa tập điểm S và các hình vuông của , sao cho

Bước 5: Đối với mỗi *x­p,xq Sj(i)*, tính *d(xp, xq)*. Đặt *xa* và *xb* là cặp điểm có khoảng cách ngắn nhất trong tất cả các cặp điểm. Trả về *xa* và *xb* là cặp điểm gần nhất.

Ví dụ



Phân tích độ phức tạp giải thuật

Trong bài toán tìm cặp điểm gần nhất ngẫu nhiên được trình bày ở trên, bước đầu tiên là tìm cặp điểm gần nhất trong điểm. Khoảng các ngắn nhất này có thể được tìm bằng cách đệ quy giải thuật ngẫu nhiên này một lần. Nghĩa là, chọn ngẫu nhiên từ điểm, sau đó dùng chiến lược trực tiếp để tìm cặp điểm gần nhất trong điểm, với số phép tính khoảng cách cần thực hiện là .

Có thể thấy rằng, bước 2 và bước 3 có thể hoàn thành với *O(n)*

Cuối cùng, câu hỏi quan trọng là tìm ra mạng lưới kích thước *δ* thích hợp. Nếu *δ* quá lớn, hình vuông ban đầu sẽ quá lớn và số lượng khoảng cách cần tính cũng lớn. Trong thực tế, nếu *δ* quá lớn, hầu như không có việc chia ra và bài toán của chúng ta trở về bài toán ban đầu. Mặt khác, *δ* cũng không thể quá nhỏ bởi vì nó không thể nhỏ hơn khoảng cách ngắn nhất. Trong giải thuật ngẫu nhiên, chúng ta chọn lựa ngẫu nhiên một tập con các điểm và tìm khoảng cách ngắn nhắn trong tập đó. Khoảng cách ngắn nhất này sẽ trở thành *δ* của chúng ta.

## Bài toán kiểm tra nhân ma trận

Giải thuật Freivalds (đặt tên theo Rusins Freivalds) là một giải thuật xác suất ngẫu nhiên được sử dụng kiểm tra phép nhân ma trận. Cho ba ma trận *A*, *B* và *C* kích thước , một vấn đề chung là kiểm tra liệu . Một giải thuật thông thường có thể tính sau đó so sánh kết quả với *C*. Tuy nhiên, ngay cả khi dùng một giải thuật nhân ma trận tốt nhất, biên tốt nhất có thể đạt được với cách tiếp cận này là theo thời gian. Giải thuật Freivalds sử dụng sự ngẫu nhiên để giảm biên thời gian này xuống với một xác suất cao. Trong độ phức tạp thời gian của thuật toán có thể xác minh ma trận kết quả với xác suất sai bé hơn .

Mô tả thuật toán

**Input:** ba ma trận *A*, *B* và *C* kích thước

**Output:** Đúng, nếu ; Sai, ngược lại.

Thủ tục

1. Tạo một ma trận ngẫu nhiên (chỉ gồm giá trị 0 và 1) kích thước , được gọi là vectơ .
2. Tính .
3. Trả về “Đúng” nếu ; Ngược lại, trả về “Sai”.

Lỗi

Nếu , thì giải thuật luôn trả lời “Đúng”. Nếu , thì xác suất giải thuật cũng trả lời “Đúng” thì bé hơn hoặc bằng . Điều này được gọi là lỗi một chiều (*one-sided error*). Bằng cách thực hiện giải thuật k lần và chỉ trả lời “Đúng” khi tất cả các lần lặp đều trả lời “Đúng”. Khi đó, ta đạt được độ phức tạp về thời gian là và xác suất lỗi .

Ví dụ

Giả sử có một yêu cầu xác minh:

.

Một vectơ ngẫu nhiên có bộ số gồm hai phần tử với giá trị nhập bằng *0* hoặc bằng *1*-được gọi là và dùng để tính

Ta được kết quả là vectơ *0*. Tuy nhiên, nếu trong lần thử thứ hai, được chọn, kết quả sẽ như sau:

Kết quả không bằng *0*, chứng minh thực tế .

Có tất cả bốn vectơ gồm 2 phần tử *0/1*, và hai trong số đó cho kết quả vectơ *0* (), vì vậy khả năng xuất hiện của sự chọn lựa ngẫu nhiên này trong hai lần thử (và giả sử kết luận rằng ) là hoặc . Trong trường hợp thông thường, tỷ lệ *r* đưa ra kết quả là vectơ *0* thì có thể nhỏ hơn , và với số lượng phép thử lớn (chẳng hạn 20 lần) được dùng, giúp cho xác suất lỗi là rất nhỏ.

Phân tích lỗi

Đặt *p* là xác suất lỗi. Cho rằng nếu , thì *p*=0, và nếu , thì *p*.

Nếu , thì , và vì vậy , chúng ta không cần quan tâm vectơ là cái gì.

Đặt , do đó . Do , chúng ta có , vì vậy một vài phần tử của *D* khác 0.

Giả sử rằng phần tử . Bằng định nghĩa của phép nhân ma trận, chúng ta có . Sử dụng định lý *Bayer* chúng ta lại có

.

Chúng ta cũng cần lưu ý rằng:

Khi thêm cái này vào hàm trên, chúng ta có:

Do đó,

## Bài toán tìm cây bao trùm nhỏ nhất của đồ thị

Trong phần này, chúng tôi sẽ giới thiệu một giải thuật ngẫu nhiên tìm cây bao trùm nhỏ nhất với độ phức tạp là Giải thuật này tốt hơn rất nhiều so với các giải thuật truyền thống tìm cây bao trùm nhỏ nhất (giải thuật Kruskal, độ phức tạp hay giải thuật Prim, độ phức tạp ).

Giải thuật này dựa trên khái niệm Boruvka step, được đề xuất bởi Boruvka vào năm 1926. Bổ đề sau minh họa ý tưởng bên trong Boruvka step:

Bổ đề ‑

Gọi và là tập hợp những đỉnh khác rỗng khi và  
 và gọi cạnh (v,u) là cạnh có trọng số nhỏ nhất với một điểm kết thúc trong và một điểm kết thúc trong . Thì, (u,v) là một trong các cạnh trong cây bao trùm nhỏ nhất của G.

Bổ đề 4.6‑1 có thể phát biểu dưới dạng khác như sau: trong một đồ thị G, với mỗi đỉnh u, trong số các cạnh liên thuộc u, nếu cạnh (u,v) có trọng số nhỏ nhất thì (u,v) phải là cạnh nằm trong cây bao trùm nhỏ nhất của G. Xem Hình 0‑1. Với đỉnh 2, trong số tất cả cạnh liên thuộc với 2, Cạnh (2,4) có trọng số nhỏ nhất. Vì vậy, cạnh (2,4) phải nằm trong cây bao trùm nhỏ nhất của G. Tương tự, dễ dàng chứng minh cạnh (5,6) cũng phải nằm trong cây bao trùm nhỏ nhất .

Hình ‑ Đồ thị

0

3

5

1

2

4

6

7

8

11

**4**

**2**

**5**

**3**

**7**

**6**

**2**

**1**

**4**

**3**

**5**

**1**

**2**

**4**

**6**

**1**

**4**

**9**

**2**

10

9

12

13

**3**

Bây giờ chúng ta hãy chọn tất cả cạnh phải nằm trong cây bao trùm nhỏ nhất, của đồ thị Hình 0‑1 dựa vào Bổ đề 4.6‑1. Kết quả những thành phần liên thông được thể hiện ở hình 4.6.2. Trong đồ thị Hình 0‑2, tất cả những đường đứt nét thì kết nối với các cạnh thành phần. bây giờ chúng ta hãy thu nhỏ tất cả các đỉnh liên thông với nhau thành một đỉnh. Bây giờ ta có 5 đỉnh, thể hiện trong Hình 0‑3. Sau khi chúng ta loại trừ những cạnh bội và khuyên được kết quả ở Hình 0‑4.

Khi đồ thị kết quả chứa nhiều hơn một đỉnh, Chúng ta áp dụng Bổ đề 4.6‑1 một lần nữa. Kết quả thể hiện ở Hình 0‑5 và cạnh được chọn là (0,3), (2,11) và (6,7).

Sau khi thu nhỏ các nút ở mỗi thành phần liên thông, chúng ta có 2 đỉnh. (Hình 0‑6) Sau khi chúng ta loại trừ những cạnh bội và khuyên được kết quả ở Hình 0‑7.

Hình ‑ Các cạnh được chọn trong Boruka step

2

0

1

3

4

5

6

7

10

9

8

11

12

13

**4**

**2**

**6**

**5**

**3**

**1**

**7**

**2**

**3**

**5**

**4**

**1**

**2**

**4**

**9**

**6**

**1**

**2**

**4**

**3**

Hình ‑ Đồ thị sau khi xây dựng

**4**

**5**

**6**

**7**

**3**

**5**

**4**

**3**

**4**

**6**

**9**

**0,1**

**2,3,4**

**5,6**

**7,9,10**

**8,11,12,13**

Hình ‑ Kết quả sau khi áp dụng Boruvka step lần thứ nhất

**7**

**4**

**3**

**3**

**5**

**2,3,4**

**0,1**

**5,6**

**7,9,10**

**8,11,12,13**

Hình ‑ Các cạnh được chọn trong Boruvka step lần 2

**7**

**4**

**3**

**3**

**5**

**0,1**

**5,6**

**2,3,4**

**7,9,10**

**8,11,12,13**

Hình ‑ Đồ thị được xây dựng lại từ Boruvka step lần 2

**5**

**7**

**0,1,2,3,4,8,11,12,13**

**5,6,7,9,10**

Hình ‑ Đồ thị kết quả thu được từ Boruvka step lần 2

**0,1,2,3,4,8,11,12,13**

**5**

**5,6,7,9,10**

0

3

5

1

2

4

6

7

8

11

**4**

**2**

**3**

**2**

**1**

**3**

**5**

**1**

**2**

**4**

**1**

**2**

10

9

12

13

**3**

Hình ‑ Cây bao trùm nhỏ nhất thu được

Một lần nữa, sau khi chúng ta loại bỏ cạnh bội và chọn (8,7), chúng ta có thể thu nhỏ tất cả đỉnh thành 1 đỉnh. Tiến trình hoàn thành. Những cạnh được chọn tạo thành một cây bao trùm nhỏ nhất (Hình 0‑8)

Giải thuật Boruvka’s để tìm cây bao trùm nhỏ nhất áp dụng Boruvka step một cách đệ quy cho đến khi đồ thị kết quả chỉ còn một cạnh. Gọi đầu vào là đồ thị G(V,E) và đầu ra là đồ thị G’(V’,E’). Boruvka step mô tả tiếp theo.

**Boruvka step:**

1. Với mỗi đỉnh u, Tìm cạnh (u,v) với trọng số nhỏ nhất kết nối với chúng. Tìm tất cả cách thành phần liên thông bằng cánh đánh dấu cạnh.
2. Thu nhỏ mỗi thành phần liên thông xác định bởi cạnh đánh dấu thành một đỉnh duy nhất. Gọi đồ thị kết quả là G’(V’,E’). Loại bỏ cạnh bội và khuyên.

Độ phức tạp của một Boruvka step là O(n+m) với n là số cạnh, m là số đỉnh. Để G liên thông thì m>n. Do đó, O(n+m)=O(m). Để mỗi thành phần liên thông xác định bởi một cạnh đánh dấu chứa ích nhất 2 đỉnh, sau khi thi hành mỗi Boruvka step , số cạnh của đồ thị thì nhỏ hơn một nữa so với ban đầu, do đó, tổng các lần Boruvka là O(log n). độ phức tạp của giải thuật Boruvka là O(m log n).

Để sử dụng Boruvka step hiệu quả, chúng ta phải sử dụng một khái niệm mới. Xem Hình 0‑9, Hình 0‑10, Hình 0‑11, Hình 0‑12. Trong Hình 0‑10, đồ thị G­s là một đồ thị con của G ở Hình 0‑9. Trong Hình 0‑12,rừng bao trùm nhỏ nhất F được gắn vào đồ thị G ban đầu . Tất cả những cạnh không thuộc F là những cạnh gạch đứt. chúng ta hãy xét cạnh (4,5). Trọng số của (4,5) là 7. Vâng, có một đường nối giữa đỉnh 4 và đỉnh 5 trong rừng F, đó là (4,3)-> (3,6)->(6,5). Trọng số của (4,5) lớn hơn trọng số lớn nhất của con đường này. Theo một bổ đề được phát biểu sao đây, cạnh (4,5) không thể là cạnh nằm trong cây bao trùm nhỏ nhất của G. Trước khi phát biểu bổ đề, chúng ta hãy mô tả một cụm từ gọi là F-heavy.

Gọi *w*(*x*, *y*) thể hiện trọng số của cạnh (*x*, *y*) trong *G*. Gọi *Gs* là một đồ thị con của *G*. *F* là rừng bao trùm nhỏ nhất của *Gs*. *wF*(*x*, *y*) thể hiện trọng số lớn nhất của một cạnh trong một đường liên thông *x* và *y* trong *F*. Nếu *x* và *y* là 2 đỉnh không liên thông trong *F*, đặt . Chúng ta nói rằng cạnh (*x*,*y*) là *F*-heavy(*F*-light) với *F* nếu .

Hình ‑

6

0

3

5

1

2

4

8

**5**

**8**

7

**7**

**5**

**3**

**12**

**1**

**10**

**2**

**10**

**9**

**4**

**6**

Hình ‑

0

3

5

1

2

4

6

8

**5**

7

**5**

**3**

**1**

**2**

**10**

**4**

**6**

Hình ‑

0

3

5

1

2

4

6

8

**5**

7

**5**

**3**

**1**

**2**

**10**

**4**

Hình ‑

0

5

3

8

4

**5**

7

**5**

**7**

**12**

**10**

**2**

**10**

**6**

1

**3**

**1**

**9**

**4**

2

**8**

6

7

Xem xét Hình 0‑12. Chúng ta có thể thấy rằng cạnh (4,5), (0,3) và (2, 6) là tất cả những F-heavy trong mối quan hệ với F. Với khái niệm mới này thì chúng ta có Bổ đề 4.6‑2 khá quan trọng.

Bổ đề ‑

Gọi Gs là một đồ thị con của G. F là một rừng bao trùm nhỏ nhất của Gs . cạnh F-heavy trong G với mối liên hệ với F không thể là cạnh nằm trong cây bao trùm nhỏ nhất.

Chúng tôi cũng không chứng minh Bổ đề 4.6‑2, với bổ đề này, chúng ta biết cạnh (4,5), (0, 3),(2, 6) không thể là cạnh cây bao trùm nhỏ nhất.

Chúng ta cần một bổ đề khác để dùng Boruvka step. Đó là Bổ đề 4.6‑3

Bổ đề ‑

Gọi H là đồ thị con của G bằng cánh bao gồm mỗi cạnh độc lập với xác xuất p, và gọi F là cây bao trùm nhỏ nhất của H. Con số mong đợi của cạnh F-light trong G lớn nhất là n/p, với n là số đỉnh của G.

**Giải thuật**

**Input:** Một đồ thị liên thông có trọng số G

**Output**: cây bao trùm nhỏ nhất của G

Bước 1: Thực hiện Boruvka step 3 lần. Gọi kết quả là đồ thị G1(V1, E1). Nếu G1 chứa một đỉnh, trả về bộ các cạnh đánh dấu ở bước 1 và thoát.

Bước 2: Phát sinh một đồ thị con H của G1 bằng cách chọn mỗi cạnh phụ thuộc với xác xuất ½. Sử dụng giải thuật đệ quy cho H để tạo một rừng bao trùm nhỏ nhất F của H. nhận được đồ thị G2(V2, E2) bằng cách xóa toàn bộ cạnh F-heavy trong G1 với mối quan hệ với F.

Bước 3: Sử dụng giải thuật đệ quy với G2

Độ phức tạp của giải thuật

Đặt T(|V|,|E|) là giá trị kỳ vọng của thời gian thời gian của giải thuật cho đồ thị

Mỗi lần chạy bước 1 tốn . Sau khi thực hiện bước 1, ta có

và . Với bước 2, thời gian để tính *H* là . Thời gian để tính *F* là T. Thời gian cần thiết để loại bỏ những cạnh *F*-heavy là . Sử dụng Bổ đề 4.6‑3, ta có giá trị kỳ vọng cho lớn nhất là . Từ đó, ta có kỳ vọng của thời gian cần thiết để thực hiện bước 3 là

T. Đặt và . Chúng ta có hệ thức truy hồi sau:

Với *c* là hằng số. Ta tìm được:

Như vậy, kỳ vọng thời gian chạy của giải thuật là

Mã giả

|  |
| --- |
| Randomized-MST (G)  G1 = BoruvkaStep(G); // Boruvka Step lần thứ I  if G1 has edges  if G1 has one edg //nếu G1 chỉ có một cạnh thì thêm vào đồ thị kết quả (solvedGraph)  Add this edg to solvedGraph  else  G2 = BoruvkaStep(G1); // Boruvka Step lần thứ II  if G2 has edges  if G2 has one edg //nếu G2 chỉ có một cạnh thì thêm vào đồ thị kết quả (solvedGraph)  Add this edg to solvedGraph  else  G3 = BoruvkaStep(G2); // Boruvka Step lần thứ III  if G3 has edges  if G3 has one edg //nếu G3 chỉ có một cạnh thì thêm vào đồ thị kết quả (solvedGraph)  Add this edg to solvedGraph // kết thúc bước 1  else  // phát sinh một đồ thị con H bằng cách chọn các cạnh từ đồ thị G3 với xác xuất 1/2  Obtain a subgraph H of G3 by selecting each edge with probability ½  F=Randomized-MST(H)  for each edg in G3 and not in F  if this edg is not a F-heavy edges  Add this edg to G4 // kết thúc bước 2  Randomized-MST(G4); // bước 3  return solvedGraph |

## Kết luận

# Chương 5

# HIỆN THỰC GIẢI THUẬT NGẪU NHIÊN CHO MỘT SỐ BÀI TOÁN



## Giới thiệu

## Bài toán sắp xếp

## Bài toán xác định số nguyên tố

## Bài toán tìm cặp điểm gần nhất

## Bài toán so trùng mẩu

## Giải thuật ngẫu nhiên tìm cây bao trùm nhỏ nhất của đồ thị

Cấu trúc dữ liệu

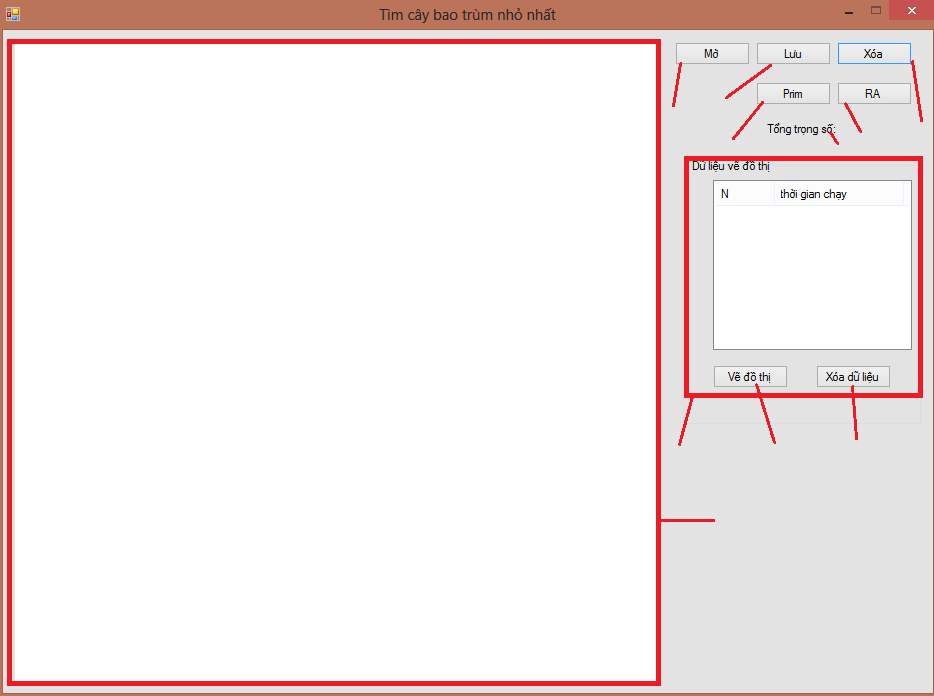
Chúng tôi mô tả một đồ thị là danh sách các cạnh, mỗi cạnh bao gồm 2 *đỉnh* (vertex), *trọng số của cạnh* (cost), vị trí hiển thị giá trị cost trên form (point).

Mỗi vertex bao gồm các thuộc tính *tên đỉnh* (name), *vùng* (rank), *nút gốc*( root), *vị trí hiển thị đỉnh trên form* (point).

Giao diện và sử dụng

Giao diện chính của phần hiện thực cho giải thuật tìm cây bao trùm nhỏ nhất:

Hình ‑ Giao diện chính của giải thuật tìm cây bao trùm nhỏ nhất



**1**

**8**

**9**

**10**

**7**

**6**

**4**

**5**

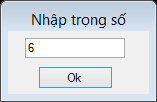
**3**

**2**

Vùng (1) đây là khu vực vẽ đồ thị mới, hiển thị đồ thị đã có lên đồng thời cũng là nơi hiển thị kết quả (cây bao trùm nhỏ nhất cần tìm)

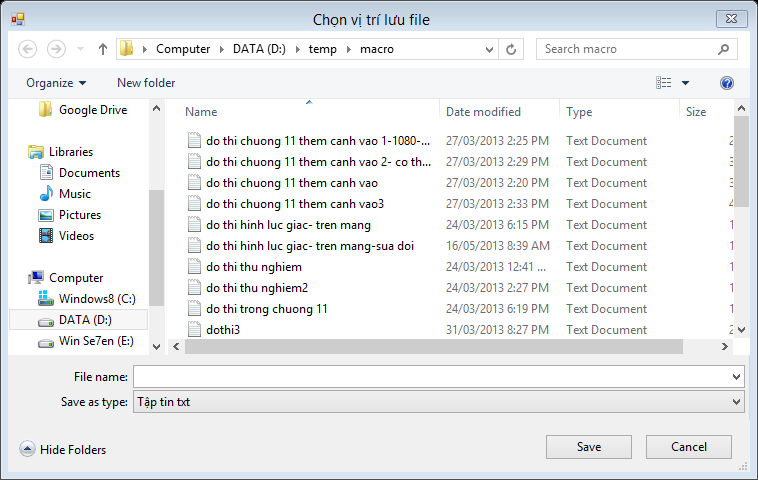
Khi muốn vẽ một đồ thị mới, ta tiến hành click chuột trên vùng 1, mỗi lần click ta sẽ nhận được một đỉnh của đồ thị tại vị trí click. Sau đó, giữ phím Ctrl click chọn lần lượt 2 đỉnh thuộc cạnh này. Một hộp thoại (Hình 0‑2) hiện lên ta nhập trọng số của cạnh này vào.

Hình ‑ Nhập trọng số



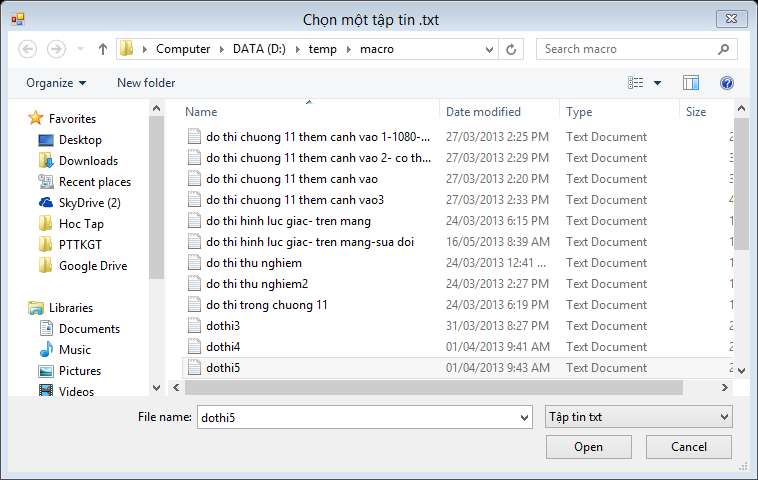
Sau khi vẽ xong đồ thị chúng ta có thể sử dụng nút *Lưu (3)* để lưu lại đồ thị đang có, để tiện sử dụng cho lần sau. Khi click *Lưu* thì một cửa số hiện ra (Hình 0‑3) ta chọn vị trí lưu, và đặt tên file, click *Save.*

Hình ‑ Chọn vị trí lưu

**

Chúng ta có thể mở đồ thị đã vẽ sẵn bằng cách click nút *Mở* (2)*.* Một của sổ hiện ra (Hình 0‑4) , chọn file đã lưu trước đó, rồi click *Open.* Đồ thị đã lưu sẽ hiện lại lên vùng (1), chúng ta có thể sử dụng luôn hay vẽ thêm cạnh.

Hình ‑ Chọn một tập tin để mở.

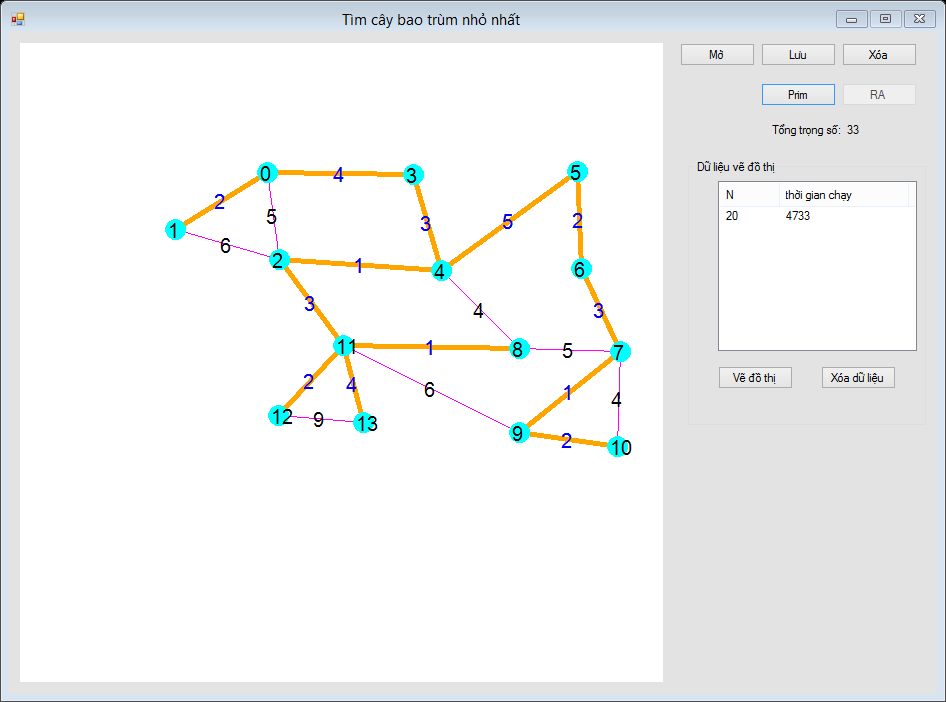


Nút *Xóa* (4) cho phép ta xóa đồ thị đang có trên vùng (1).

Nút *Prim* (5) và *RA* (6), đây là 2 nút quan trọng nhất , nút (5) cho phép ta tìm cây bao trùm nhỏ nhất theo giải thuật Prim, nút (6) cho phép ta tìm cây bao trùm nhỏ nhất theo giải thuật ngẫu nhiên đã trình bày ở chương 4.

*Tổng trọng số* (7), là tổng trọng số của cây bao trùm nhỏ nhất tìm được

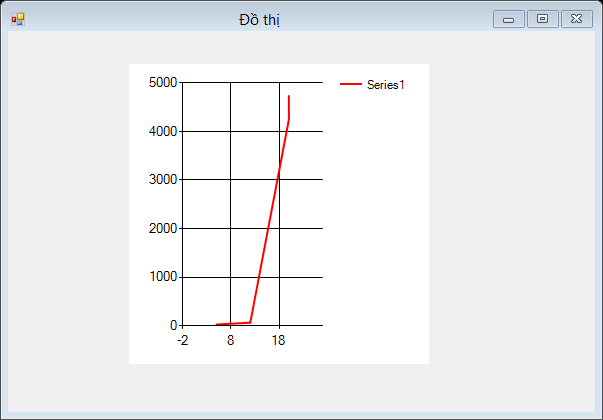
Hình ‑ Kết quả thu được khi thực hiện giải thuật tìm cây bao trùm nhỏ nhất



Vùng (8) là khu vực dùng vẽ đồ thị thời gian chạy cho giải thuật

Mỗi lần thực hiện giải thuật, chương trình sẽ đếm số cạnh của đồ thị và tính thời gian chạy (đơn vị microsecond) đưa vào list. Từ đó ta có thể vẽ đồ thị thời gian (như Hình 0‑6) chạy bằng nút *Vẽ đồ thị* (9), ta cũng có thể xóa hết dữ liệu trong list này bằng cách click nút *Xóa dữ liệu* (10).

Hình ‑ Đồ thị thời gian chạy.

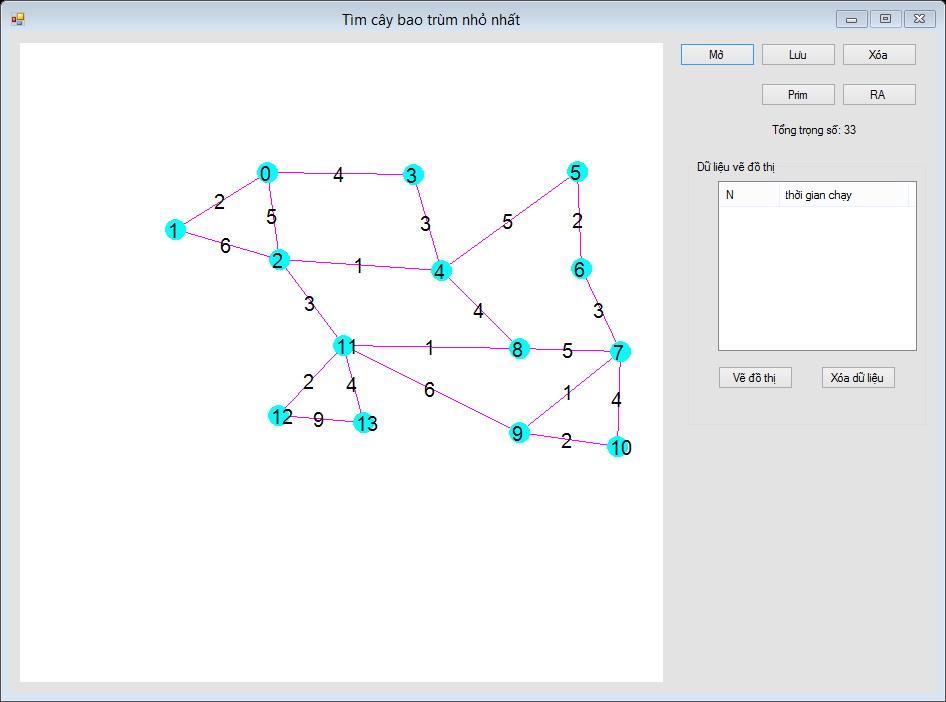


Dữ liệu thử

Đồ thị 1

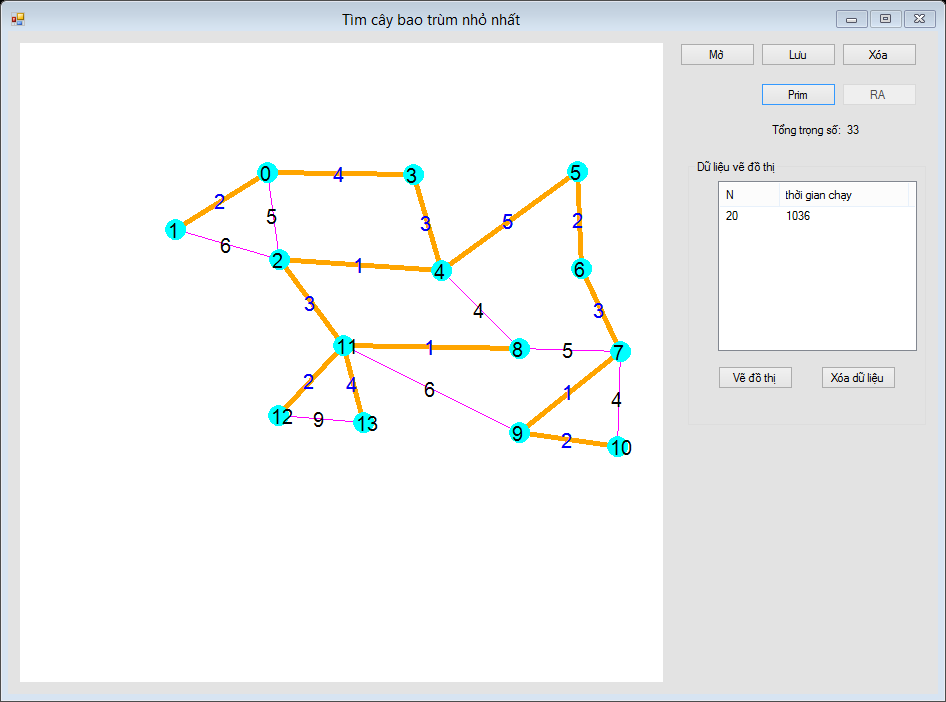
Đồ thị đưa vào (Hình 0‑7)

Hình ‑



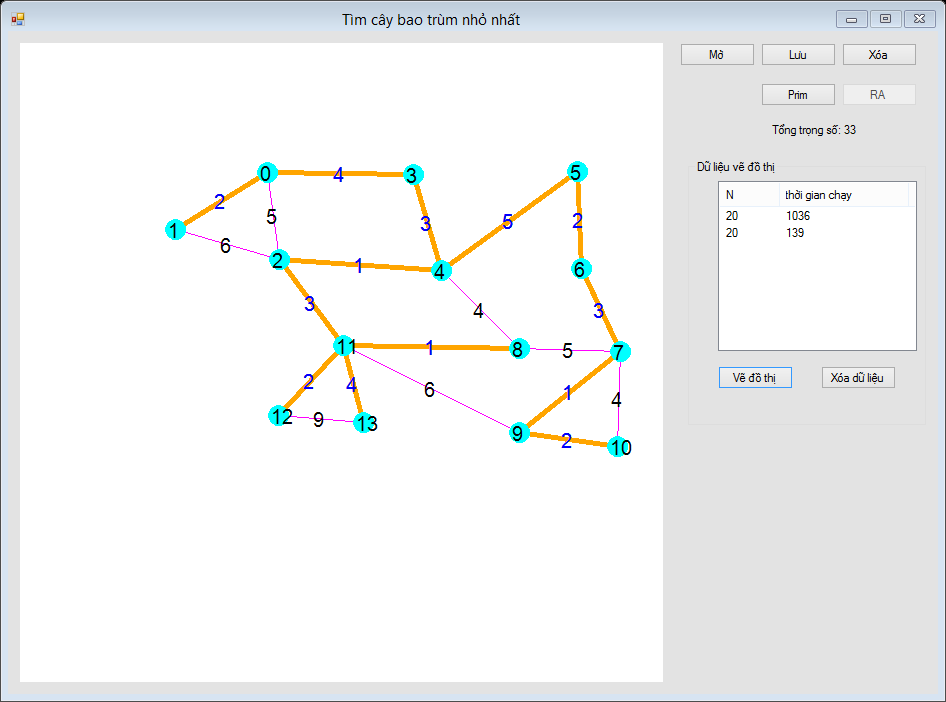
Kết quả khi chạy bằng giải thuật Prim (Hình 0‑8)

Hình ‑



Kết quả khi chạy bằng giải thuật ngẫu nhiên (Hình 0‑9)

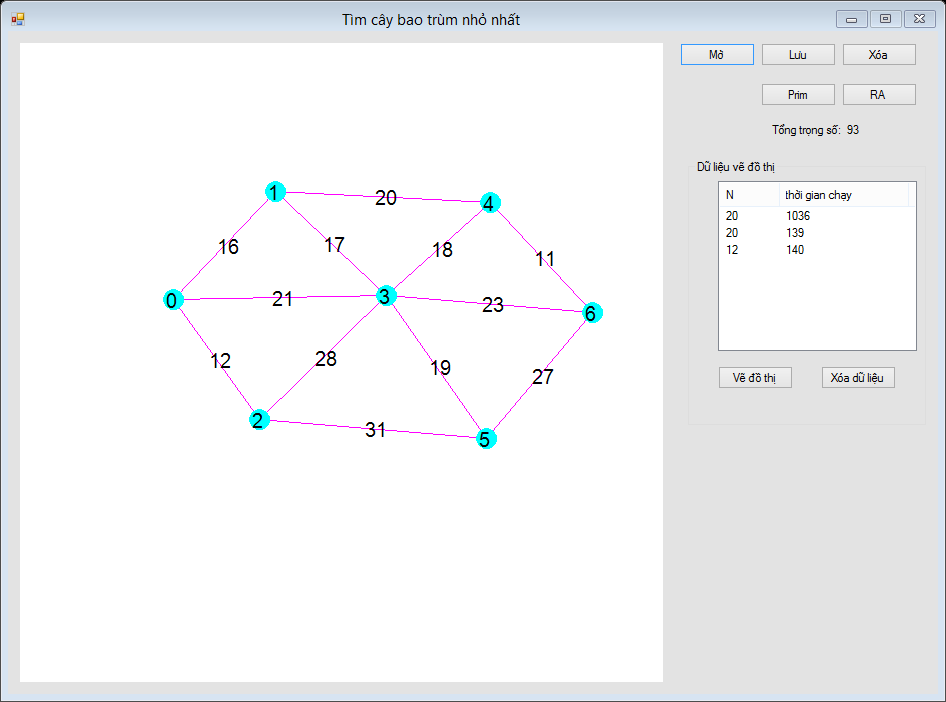
Hình ‑



Đồ thị 2

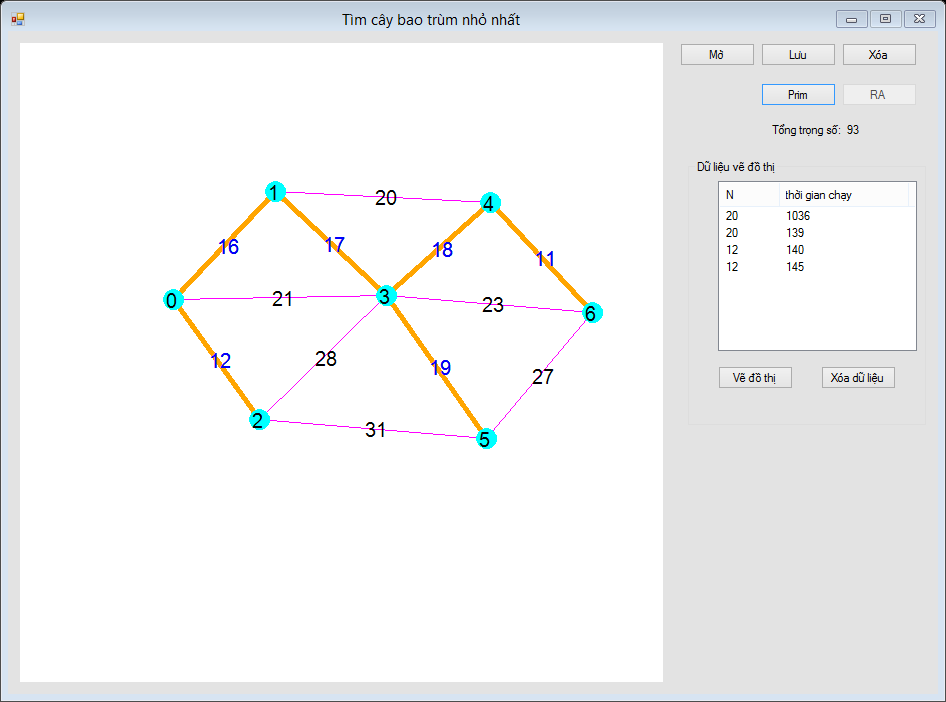
Đồ thị đưa vào (Hình 0‑10)

Hình ‑



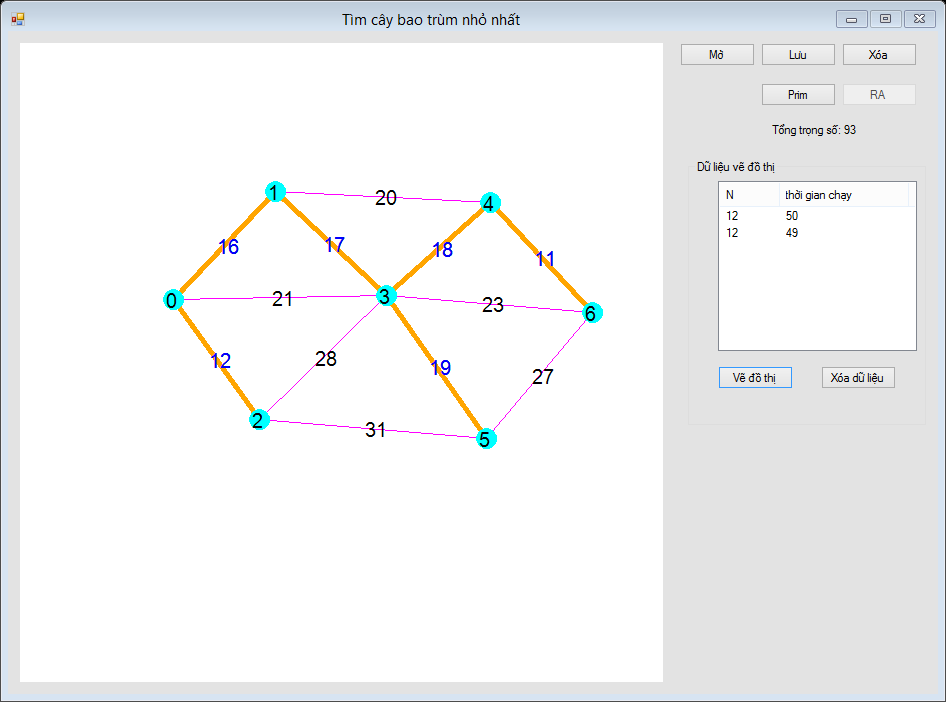
Kết quả khi chạy bằng giải thuật Prim (Hình 0‑11)

Hình ‑



Kết quả khi chạy bằng giải thuật ngẫu nhiên (Hình 0‑12)

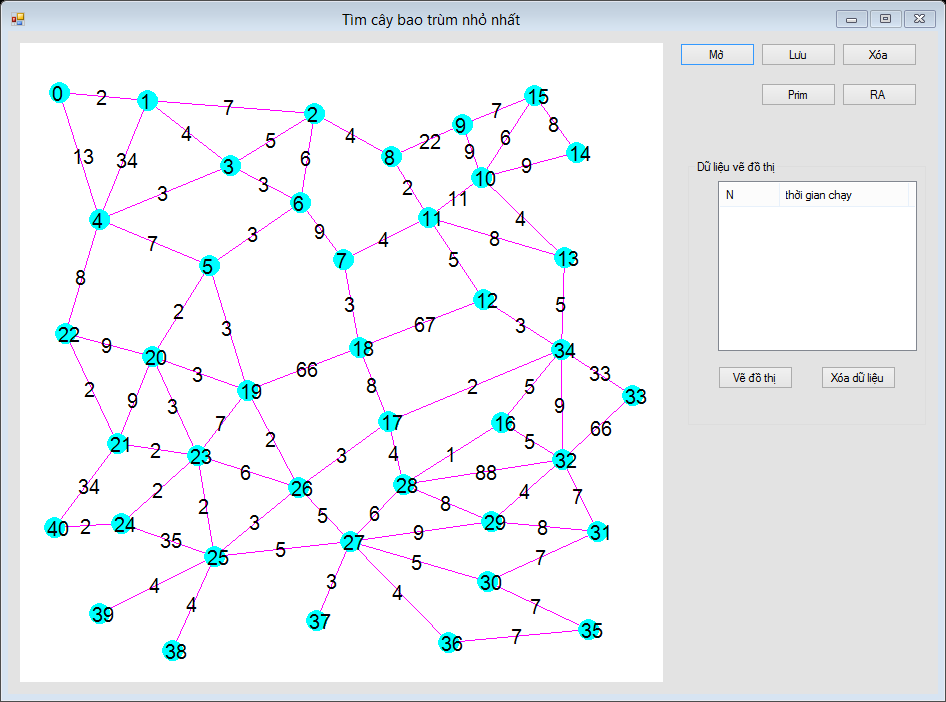
Hình ‑



Đồ thị 3

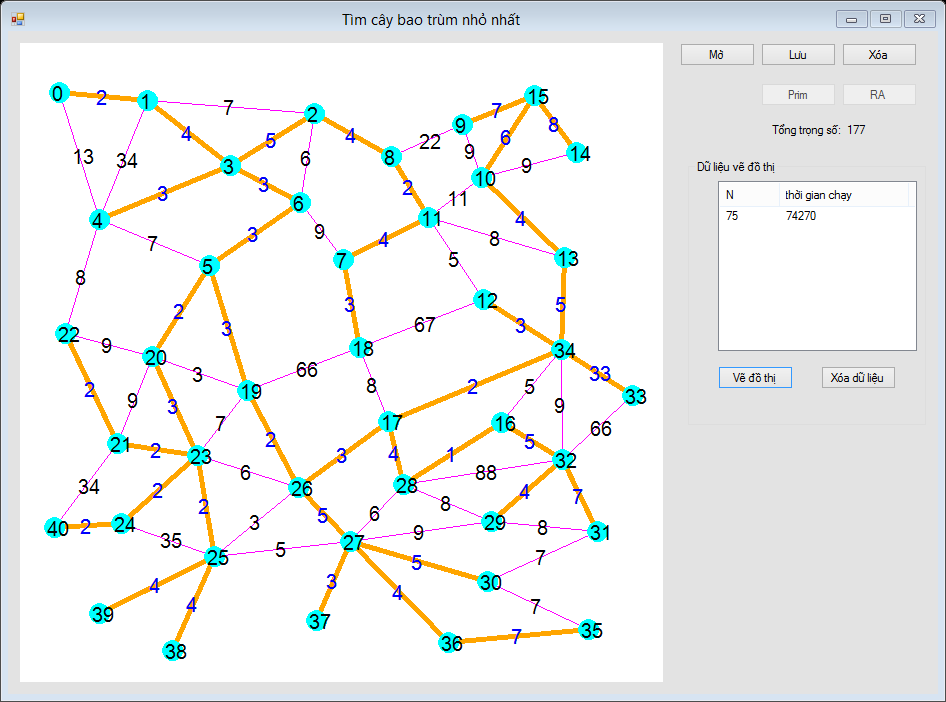
Đồ thị đưa vào (Hình 0‑13)

Hình ‑



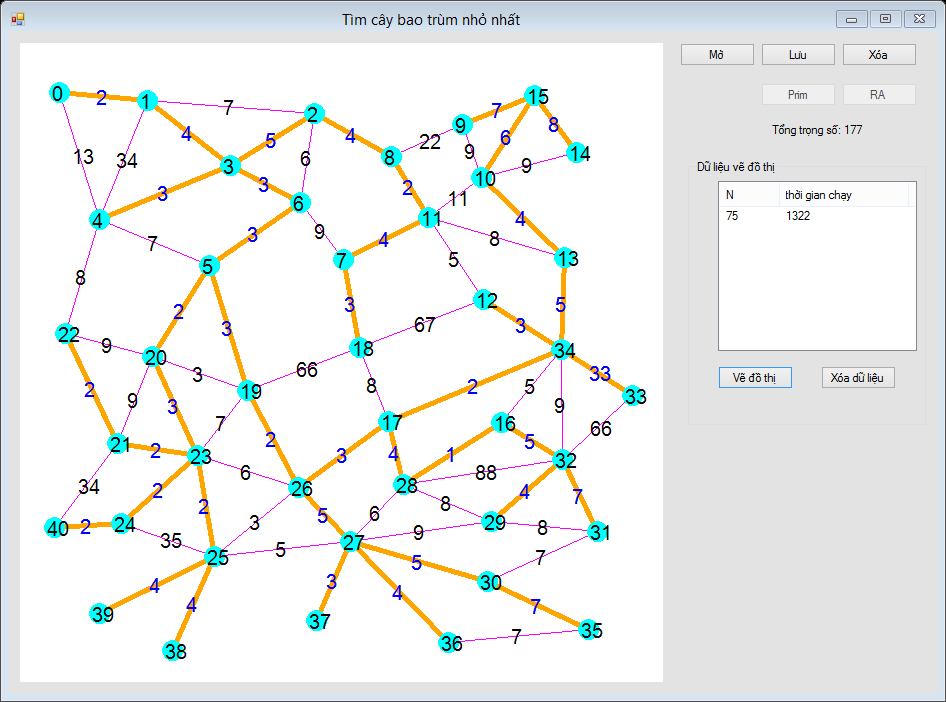
Kết quả khi chạy bằng giải thuật Prim (Hình 0‑14)

Hình ‑



Kết quả khi chạy bằng giải thuật ngẫu nhiên (Hình 0‑15)

Hình ‑



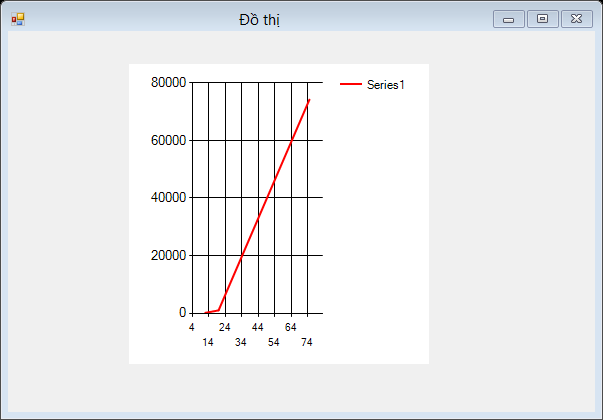
So sánh giữa giải thuật ngẫu nhiên và giải thuật Prim

Bảng Bảng so sánh thời gian chạy (đơn vị tính: microsecond) của giải thuật ngẫu nhiên và giải thuật Prim

|  |  |  |
| --- | --- | --- |
| Đồ thị | Thời gian chạy của giải thuật Prim | Thời gian chạy của giải thuật ngẫu nhiên |
| Đồ thị 1 | 988 | 139 |
| Đồ thị 2 | 146 | 68 |
| Đồ thị 3 | 75158 | 1323 |

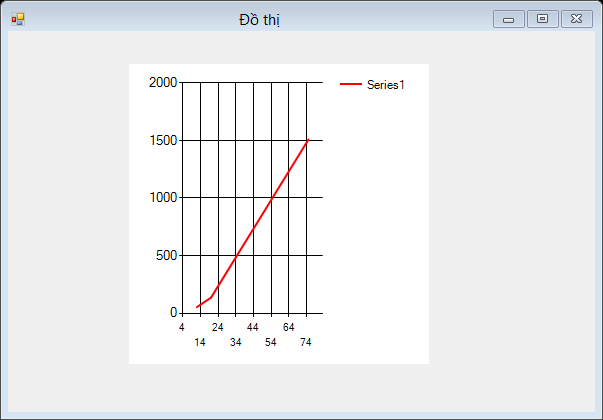
Biểu đồ thời gian chạy của giải thuật Prim

Hình ‑ Biểu đồ thời gian chạy của giải thuật Prim



Biểu đồ thời gian chạy của giải thuật ngẫu nhiên

Hình ‑ Biểu đồ thời gian chạy của giải thuật ngẫu nhiên



## Kết luận

# Chương 6

# TỔNG KẾT VÀ ĐỀ NGHỊ



## Tổng kết

## Đề nghị

# TÀI LIỆU THAM KHẢO

[1] Levitin A. *Introduction to The design and Analysis of Algorithms*. Addison-Wesley, 2012.

[2] Thomas H. Cormen, Charles E. Leiserson, Ronald D. Rivest, clifford Stein.

*Introduction to Algorithms*. McGraw-Hill Book Company, 2009.

[3] Karp Richard M.. *An introduction to randomized algorithms*. Discrete Applied

Mathematics 34 (1991) 165-201.

[4] Mitzenmacher M., Upfal E. *Probability and Computing Randomized Algorithms and Probabilistic Analysis*. Cambridge university press, 2005

[5] Motwani R., Prabhakar Raghavan P. *Randomized algorithms*. Cambridge university press, 1995.

[15] Tin Học 10, Nhà xuất bản giáo dục .

Website

[17] en.wikipedia.org/

[18] vi.wikipedia.org/