**MỤC LỤC**

[I. Tổng quan về thuật toán tham lam 1](#_Toc22253640)

[1. Sơ lược giải thuật tham lam 1](#_Toc22253641)

[**I.1. Khái niệm** 1](#_Toc22253642)

[**1.** **Đặc trưng chiến lược tham lam.** 1](#_Toc22253643)

[2. Lý thuyết Matroids 2](#_Toc22253644)

[**Các định nghĩa liên quan đến lý thuyết trên:** 2](#_Toc22253645)

[**1.** **Định nghĩa Matroids** 2](#_Toc22253646)

[**2.** **Định lý** 3](#_Toc22253647)

[**3.** **Thuật toán tham lam trên một Matroid trọng số** 4](#_Toc22253648)

[** Ở đây ta có thuật toán tham lam trên Matroids trọng số bất kì** 5](#_Toc22253649)

[*1. A ← Ø* 5](#_Toc22253650)

[*4.*  *do if A{x} ℓ [M]* 5](#_Toc22253651)

[*5.*   *then AA{x}* 5](#_Toc22253652)

[*6. return A* 5](#_Toc22253653)

[**Matroid M=(S,l). Nếu xS mà x không là mở rộng của  thì x không là mở rộng của bất kỳ tập con độc lập A nào của S.** 7](#_Toc22253654)

[**Nếu M=(S,l) là một matroid có trọng số với hàm trọng số là w thì hàm Greedy(S, l,w) trả về một tập con tối ưu.** 7](#_Toc22253655)

[I.3 Nguyên tắc - Đặc điểm của thuật toán tham lam 7](#_Toc22253656)

[I.4. Điều kiện để một bài toán áp dụng được giải thuật tham lam 8](#_Toc22253657)

[I.5. Những dạng bài toán thường được áp dụng để giải quyết. 9](#_Toc22253658)

[II. THUẬT TOÁN THAM LAM 9](#_Toc22253659)

[II.1. Mục tiêu nghiên cứu thuật toán tham lam 9](#_Toc22253660)

[II.2. Các thành phần của chiến lược tham lam 9](#_Toc22253661)

[-Tính lựa chọn tham lam 10](#_Toc22253662)

[- Cấu trúc con tối ưu 10](#_Toc22253663)

[II.3. Sơ đồ thuật toán 10](#_Toc22253664)

[II.4. Một số chiến lược - Tiến trình thực hiện – Các bước thực hiện thuật toán tham lam. 10](#_Toc22253665)

[II.5. Chứng minh tính đúng đắn 11](#_Toc22253666)

[**1.** **Lập luận biến đổi (Exchange Argument)** 11](#_Toc22253667)

[III- MỘT SỐ BÀI TOÁN ÁP DỤNG THUẬT TOÁN THAM LAM 12](#_Toc22253668)

[III.1. Bài toán người du lịch 12](#_Toc22253669)

[**1.** **Mô tả bài toán** 12](#_Toc22253670)

[**2.** **Ý tưởng giải quyết bài toán** 12](#_Toc22253671)

[**3.** **Mô tả bài toán bằng thuật toán tham lam như sau:** 12](#_Toc22253672)

[**4.** **Chương trình mã giả minh họa cho thuật toán** 12](#_Toc22253673)

[**5.** **Đánh giá độ phức tạp** 13](#_Toc22253674)

[III.2.Bài toán mã Huffman 13](#_Toc22253675)

[**1.** **Mô tả bài toán** 13](#_Toc22253676)

[**2.** **Nội dung - Ý tưởng giải quyết bài toán** 15](#_Toc22253677)

[**3.** **Chương trình mã giả minh họa cho thuật toán** 16](#_Toc22253678)

[**4.** **Đánh giá độ phức tạp** 17](#_Toc22253679)

[**1.** **Mô tả bài toán** 17](#_Toc22253680)

[2. **Yêu cầu:** Mỗi thời điểm chỉ có 1 hoạt động sử dụng tài nguyên chung. 18](#_Toc22253681)

[3. **Mục tiêu:** Chọn được 1 tập lớn nhất các hoạt động tương thích với nhau (khoảng thời gian thực hiện không giao nhau)=>Tận dụng tối đa tài nguyên. 18](#_Toc22253682)

[**4.** **Đánh giá độ phức tạp** 19](#_Toc22253683)

**GIỚI THIỆU GIẢI THUẬT THAM LAM**

# **I. Tổng quan về thuật toán tham lam**

**1. Sơ lược giải thuật tham lam**

**I.1. Khái niệm**

**Giải thuật tham lam** (tiếng Anh: *Greedy algorithm*) là một thuật toán giải quyết một bài toán theo kiểu metaheuristic để tìm kiếm lựa chọn tối ưu địa phương ở mỗi bước đi với hy vọng tìm được lựa chọn tối ưu toàn cục.

**Chiến lược tham lam** là một chiến lược xây dựng thuật toán tìm nghiệm tối ưu cục bộ cho các bài toán tối ưu nhằm đạt được nghiệm tối ưu toàn cục cho cả bài toán (trong trường hợp tổng quát). Trong trường hợp cho nghiệm đúng, lời giải của chiến lược tham lam thường rất dễ cài đặt và có hiệu năng cao (độ phức tạp thuật toán thấp).

Chẳng hạn áp dụng giải thuật tham lam với bài toán “hành trình của người bán hàng” ta có giải thuật sau: "Ở mỗi bước hãy đi đến thành phố gần thành phố hiện tại nhất".

***Giải thuật tham lam có năm thành phần:***

1. Một tập hợp các ứng viên (candidate), để từ đó tạo ra lời giải.
2. Một hàm lựa chọn, để theo đó lựa chọn ứng viên tốt nhất để bổ sung vào lời giải.
3. Một hàm khả thi (*feasibility*), dùng để quyết định nếu một ứng viên có thể được dùng để xây dựng lời giải.
4. Một hàm mục tiêu, ấn định giá trị của lời giải hoặc một lời giải chưa hoàn chỉnh.
5. Một hàm đánh giá, chỉ ra khi nào ta tìm ra một lời giải hoàn chỉnh
6. **Đặc trưng chiến lược tham lam.**
7. **Phương pháp tham lam** là kỹ thuật thiết kế thường được dùng để giải các bài toán tối ưu. Phương pháp được tiến hành trong nhiều bước. Tại mỗi bước, theo một chọn lựa nào đó (xác định bằng một hàm chọn), sẽ tìm một lời giải tối ưu cho bài toán nhỏ tương ứng. Lời giải của bài toán được bổ sung dần từng bước từ lời giải của các bài toán con. Lời giải được xây dựng như thế có chắc là lời giải tối ưu của bài toán?
8. Các lời giải theo phương pháp tham lam thường chỉ là chấp nhận được theo điều kiện nào đó, chưa chắc là tối ưu.
9. Cho trước một tập A gồm n đối tượng, ta cần phải chọn một tập con S của A. Với một tập con S được chọn ra thỏa mãn các yêu cầu của bài toán, ta gọi là một nghiệm chấp nhận được . Một hàm mục tiêu gắn mỗi nghiệm chấp nhận được với một giá trị. Nghiệm tối ưu là nghiệm chấp nhận được mà tại đó hàm mục tiêu đạt giá trị nhỏ nhất (lớn nhất).

* **Đặc trưng tham lam** của phương pháp thể hiệntrong mỗi bước việc xử lí sẽ tuân theo một sự chọn lựa trước, không kể đến tình trạng không tốt có thể xảy ra khi thực hiện lựa chọn lúc đầu.

## **2. Lý thuyết Matroids**

Lý thuyết Matroids, lý thuyết này cho phép xác định khi nào thuật toán tham lam đưa ra phương án tối ưu nhất. Nó liên quan đến cấu trúc tổ hợp đã biết như “Matroids”. Lý thuyết Matroids được ông Hasser Whitney đưa ra trong bài báo “ On the abstract properties of linear dependence” . Lý thuyết này không áp dụng cho tất cả các trường hợp (ví dụ, không áp dụng cho bài toán lựa chọn hoạt động hoặc bài toán mã Huffman), nó áp dụng một số trường hợp mang tính thực tế hơn ví dụ: bài toán tìm cây khung nhỏ nhất. Lý thuyết này đang được phát triển và mở rộng để áp dụng cho nhiều ứng dụng hơn nữa.

### **Các định nghĩa liên quan đến lý thuyết trên:**

1. **Định nghĩa Matroids**

Một matroid mà một bộ 2 có thứ tự M=(S,l), thỏa mãn:

* + S là một tập hữu hạn khác rỗng (*tập không rỗng hữu hạn*)
  + ℓ là một họ khác rỗng các tập con (độc lập) của S, thỏa mãn: nếu Bl và AB thì Al (ta nói l là di truyền) . Lưu ý : tập rỗng nhất thiết phải là một phần tử của l

M thỏa mãn tính chất trao đổi (exchange) nếu Al, Bl và |A|<|B| thì xB-A mà A{x}l

Ví dụ:

* Matroid đồ thị MG=(SG,lG). G=(V,E) là đồ thị vô hướng. SG là tập các cạnh của đồ thị G. Nếu A là một tập con của E .AlG A không chứa chu trình

### **Định lý**

1. ***Định lý 1*:** Nếu G=(V,E) là một đồ thị vô hướng thì MG=(SG,lG) là một matroid.

*Chứng minh:*

* SG=E là một tập hữu hạn khác rỗng.
* lG là di truyền (Bl và AB thì Al) vì một tập con của một rừng là một rừng . Nói cách khác,loại bỏ các cạnh từ chu trình của tập các cạnh không tạo chu trình thì không bao giờ có thể tạo ra chu trình.
* MG thoả mãn tính chất trao đổi . Tính chất được minh họa bởi hình dưới. A, B chứa các cạnh không taọ chu trình. GA,GB là rừng được tạo thành.

GA

GB

2

3

4

5

1

2

3

4

5

1

*Phần tử mở rộng:*

Cho một matroid M=(S, l), ta gọi một phần tử x∉A là phần tử mở rộng của A∈l nếu x có thể được thêm vào A trong khi vẫn đảm bảo tính độc lập. Nói cách khác x là phần tử mở rộng của A nếu A ∪{x}∈ l.

Ví dụ:

Cho một matroid đồ thị MG . Nếu A là tập hợp các cạnh độc lập thì cạnh e là phần tử mở rộng của A nếu và chỉ nếu e không thuộc A và khi thêm e vào A thì không tạo chu trình.

Nếu A là một tập con độc lập trong matroid M, ta nói rằng A là lớn nhất nếu không có phần tử mở rộng.Tức là, A là lớn nhất nếu nó không được chứa trong bất kỳ tập con độc lập lớn hơn của M.

b. Định lý 2 : **Tất cả các tập con độc lập lớn nhất trong matroid có cùng lực lượng.**

*Chứng minh:*

* Giả sử A là tập con độc lập lớn nhất của M và tồn tại B là tập con độc lập lớn nhất khác của M.
* Tính chất trao đổi chỉ ra rằng A được mở rộng thành một tập độc lập lớn hơn A{x} với xB-A.  điều này mâu thuẫn với giả thiết A là lớn nhất.  dpcm

Minh hoạ định lý : Cho một matroid đồ thị MG của đồ thị liên thông, vô hướng G. Mỗi tập con độc lập lớn nhất của MG phải thuộc một cây với |V| −1 cạnh nối đến tất cả các đỉnh của G. Cây như thế được gọi là cây khung của G.

1. **Thuật toán tham lam trên một Matroid trọng số**

Matroid có trọng số:

Matroid M=(S,l) là có trọng số nếu  ánh xạ: w: S  R+

x  w(x)

Với AS, ta có: w(A) = sumxA(w(x))

w(B)

w(A)

Ví dụ nếu ta đặt w(e) là độ dài của cạnh e trong Matroid đồ thị MG, thì w(A) là tổng độ dài của các cạnh trong tập cạnh A.

Thuật toán tham lam trên một matroid trọng số

Nhiều bài toán sử dụng Tham lam để tim lời giải tối ưu tương đương việc tìm tập con cực đại có trọng số lớn nhất trong một matroid có trọng số.

Tức là, cho một matroid trọng số M = (S,ℓ) và ta muốn tìm ra một tập Aℓ độc lập sao cho w(A) được lớn nhất. Ta gọi tập con độc lập và có trọng số lớn nhất là tập con tối ưu của matroid. Bởi vì trọng số w(x) của bất kỳ phần tử xS là số dương, một tập con tối ưu luôn là tập con độc lập lớn nhất.

Ví dụ:

Trong bài toán ***tìm cây khung nhỏ nhất:*** Ta có một đồ thị vô hướng, liên thông G=(V,E) và hàm tính độ dài w sao cho w(e) là độ dài của cạnh e. Ta cần tìm ra tập con của các cạnh mà kết nối tất cả các đỉnh với nhau và có tổng độ dài nhỏ nhất Xem ví dụ này như là một bài toán tìm một tập con tối ưu của matroid.

* Gọi matroid MG với hàm trọng số w’: w’(e) = w0-w(e). Trong đó w0 = max{w(e)}+1
* Mỗi tập con độc lập lớn nhất A tương đương với một cây khung. Từ đó: w’(A) = (|V|-1)w0 - w(A). w(A) là độ dài của cây khung
* Một tập con độc lập đạt cực đại về w’(A) thì phải cực tiểu w(A). Vì vậy, một thuật toán bất kỳ có thể tìm ra một tập con tối ưu A trong matroid tuỳ ý có thể giải quyết bài toán cây khung nhỏ nhất

### ** Ở đây ta có thuật toán tham lam trên Matroids trọng số bất kì**

Thuật toán là tham lam bởi vì nó xem mỗi phần tử xS lần lượt được sắp xếp theo trọng số giảm dần đều và trực tiếp cộng nó vào tập A đang được tích luỹ nếu A {x} là độc lập.

Giải thuật trên có:

* Đâù vào : một matroid trọng số M = (S,ℓ) với một hàm trọng số dương w
* Đầu ra: trả về một tập con tối ưu A

Mã giả: (*thành phần của M là S[M] và ℓ[M]và hàm trọng số là w*)

*GREEDY(M,w)*

### *1. A ← Ø*

*2. Sắp xếp S[M] theo thứ tự giảm dần bởi trọng số w*

*3. for mỗi xS[M] lấy ra từ tập được sắp xếp giảm dần theo trọng số w(x)*

### *4. do if A{x} ℓ [M]*

### *5. then AA{x}*

### *6. return A*

 Giải thích:

Những phần tử thuộc S được sắp xếp giảm dần theo khối lượng. Nếu phần tử x được xem có thể thêm vào A trong khi vẫn duy trì sự độc lập của A, thì đúng là nó. Bằng không, x bị loại bỏ. Bởi vì tập rỗng là độc lập bởi định nghĩa về matroid và x được thêm vào A với điều kiện A ∪{x} là độc lập, nên tập con A luôn luôn độc lập bằng phương pháp qui nạp.

Vì vậy, GREEDY luôn trả về một tập con độc lập A. Như sẽ thấy dưới dây A là tập con trọng số khả dĩ lớn nhất, vì vậy A một tập con tối ưu.

Thời gian thực hiện của thuật toán tham lam.

Cho n biểu diễn cho S . Giai đoạn sắp xếp mất thời gian là O(nlogn). Dòng 4 thực hiện chính xác là n lần, mỗi lần là một phần tử của S. Mỗi lần thực hiện dòng 4 yêu cầu là tập A∪ {x} có độc lập không. Nếu mỗi lần kiểm tra như thế chiếm O(f(n)), thì một giải thuật hoàn chỉnh chạy trong thời gian O(nlgn + nf(n)).

Ta chứng minh rằng GREEDY trả về một tập con tối ưu.

Bổ đề 1: **Matroid có tính lựa chọn tham lam**

* M=(S,l) là một matroid có trọng số, với hàm trọng số w và tập S có thứ tự không tăng dần theo trọng số.
* x là phần tử đầu tiên của S mà {x} độc lập.
* Nếu x thì  một tập con tối ưu A của S chứa x.

Chứng minh:

* Nếu không tồn tại x, khi đó chỉ có một tập con độc lập duy nhất là tập rỗng
* Mặt khác , cho B là một tập con tối ưu khác rỗng bất kỳ.
  + xB : Vì việc chọn x đảm bảo rằng w(x)>= w(y) với bất kì

y ∈B.

* + xB : xây dựng tập A .

Cấu trúc của tập A như sau:

Bắt đầu với A={x}. Do cách chọn x, nên A là tập độc lập. Sử dụng thuộc tính trao đổi , lặp lại việc tìm một phần tử mới của B sao cho có thể thêm vào A cho đến khi |A|=|B| trong khi vẫn giữ tính độc lập của A.

 Lúc đó A=B-{y} {x} với y∈ B, vì vậy:

**w(A)= w(B)-w(y)+w(x)>=w(B)**

Vì B là tối ưu, A cũng phải tối ưu, và vì x ∈ A

 Bổ đề đúng.

Bổ đề 2: **Với matroid M=(S,l). Nếu xS là một mở rộng của tập con độc lập A nào đó của S thì x cũng là một mở rộng của .**

Chứng minh:

* x là 1 mở rộng của A  A{x} độc lập.
* l di truyền  {x} độc lập.

*Hệ quả 1:*

### **Matroid M=(S,l). Nếu xS mà x không là mở rộng của  thì x không là mở rộng của bất kỳ tập con độc lập A nào của S.**

*Bổ đề 3:* **Matroid có tính cấu trúc con tối ưu**

Gọi x là phần tử đầu tiên của S được chọn bởi hàm Greedy. Vấn đề còn lại của việc tìm một tập con độc lập có trọng số cực đại (chứa x) là tìm một tập con độc lập có trọng số cực đại của matroid có trọng số M’=(S’,l’) với điều kiện :

* + S’ = {yS | {x,y}l,
  + l’ = {BS-{x} | B{x}S},
  + Hàm trọng số cho M’ là hàm trọng số cho M nhưng giới hạn bởi S’ (gọi M’ là rút gọn của M bởi x)

Tính đúng đắn của thuật toán tham lam trên Matroids:

### **Nếu M=(S,l) là một matroid có trọng số với hàm trọng số là w thì hàm Greedy(S, l,w) trả về một tập con tối ưu.**

Chứng minh:

* *Hệ quả 1*  những phần tử bị bỏ lúc đầu không hữu dụng  sau này không cần xét lại chúng nữa.
* *Bổ đề 1*  khi một phần tử đầu tiên x được chọn, Greedy đúng khi thêm x vào A vì luôn tồn tại một tập con tối ưu chứa x.
* *Bổ đề 3*  bái toán còn lại là tìm tập con tối ưu trong matroid M’ (M’ là rút gọn của M bởi x).

## **I.3 Nguyên tắc - Đặc điểm của thuật toán tham lam**

Mục đích của phương pháp tham lam (Greedy) là xây dựng bài toán giải nhiều lớp bài toán khác nhau, đưa ra quyết định dựa ngay vào thuật toán đang có, và trong tương lai sẽ không xem xét lại quyết định trong quá khứ.

Do vậy thuật toán tham lam có ưu điểm:

* Dễ đề xuất.
* Thời gian tính nhanh.

Thuật toán tham lam có những đặc điểm sau đây:

Lời giải của bài toán là một tập hữu hạn S các phần tử thoả mãn điều kiện nào đó, ta phải giải quyết bài toán một cách tối ưu. Nói cách khác, nghiệm S phải được xây dựng sao cho hàm mục tiêu f(S) có giá trị tốt nhất (lớn nhất hay nhỏ nhất) có thể được.

Các bước giải bài toán như sau:

* Có một tập các ứng cử viên C để chọn cho các thành phần của nghiệm tại mỗi bước.
* Xuất phát từ lời giải rỗng S, tại mỗi bước của thuật toán, ta sẽ lựa chọn một ứng cử viên trong C để bổ sung vào lời giải S hiện có.
* Xây dựng được hàm Select(C) tại mỗi bước chọn để lựa chọn một ứng cử viên có triển vọng nhất để đưa vào lời giải S.
* Xây dựng được hàm Feasible(S ∪ x) để kiểm tra tính chấp nhận được của ứng cử viên x khi đưa vào tập nghiệm S.
* Cuối cùng khi có được tập S ,xây dựng hàm Solution(S) để kiểm tra tính chấp nhận được của lời giải S.

## **I.4. Điều kiện để một bài toán áp dụng được giải thuật tham lam**

Các dạng bài tìm phương án tối ưu như bài toán người du lịch, bài toán cái túi…. Chúng thuộc lớp các bài toán tối ưu tổ hợp là một trường hợp riêng của bài toán tối ưu.

Các bài toán tối ưu tổ hợp có rất nhiều ứng dụng trong thực tiễn và việc ứng dụng trở nên tốt hơn rất nhiều khi người ta nghiên cứu các thuật toán tối ưu và cài đặt trên máy tính điện tử.

Một trong những thuật toán để giải quyết các bài toán trên là thuật toán tham lam.

Thuật toán tham ăn (Greedy algorithms) được dùng để giai quyết các bài toán mà chúng ta có thể quyết định đâu là lựa chọn tốt nhất.

Các bài toán áp dụng giải thuật tham lam có những đặc điểm sau đây:

* Tính lựa chọn tham lam (Greedy choice property): Một nghiêm tối ưu có thể nhận được bằng cách thực lựa chọn phương án tốt nhất tại mỗi thời điểm và không cần quan tâm tới các gợi ý của nó đối với các nghiệm của bài toán con. Tức là một nghiêm tối ưu của bài toán có thể được nhận bằng cách thực hiện lựa chọn tối ưu cục bộ.
* Tính chất cấu trúc con tối ưu: Một nghiệm tối ưu có thể nhận được bằng cách thêm các nghiệm thành phần đã được xây dựng với một nghiêm tối ưu của bài toán con vào. Tức là một nghiêm tối ưu sẽ chứa các nghiệm tối ưu đối với các bài toán con nhỏ hơn.

Nếu có thể chứng minh rằng một thuật toán tham lam cho ra kết quả tối ưu toàn cục cho một lớp bài toán nào đó, thì thuật toán thường sẽ trở thành phương pháp được chọn lựa, vì nó chạy nhanh hơn các phương pháp tối ưu hóa khác như quy hoạch động. Tuy nhiên trong một số trường hợp thuật toán tham lam chỉ cho nghiêm gần đúng với nghiêm tối ưu.

## **I.5. Những dạng bài toán thường được áp dụng để giải quyết.**

Các thuật toán tham lam chủ yếu để giải quyết các bài toán tối ưu.

Các bài toán tối ưu là các bài toán có dạng tổng quát như sau:

* Hàm f(X) được gọi là hàm mục tiêu , xác định trên một tập hữu hạn các phần tử D.
* Mỗi phần tử X thuộc D có dạng X=(x1,x2,….,xn) được gọi là một phương án.
* Tìm một phương án X0 thuộc D sao cho f(X) đạt max hoặc min trên D. Thì X0 được gọi là phương án tối ưu.
* Tập D được gọi là tập các phương án của bài toán.

**Ví dụ như các dạng bài toán sau:**

* Một tập các đối tượng.
* Một dãy các đối tượng đã lựa chọn.
* Một hàm để xem một tập các đối tượng có lập thành một giải pháp hay không (không nhất thiết tối ưu).
* Một hàm để xem một tập đối tượng có là tiềm năng hay không.
* Một hàm để lựa chọn ứng viên có triển vọng nhất.
* Một hàm đích cho một giá trị của một giải pháp (để tối ưu hóa).

# **II. THUẬT TOÁN THAM LAM**

## **II.1. Mục tiêu nghiên cứu thuật toán tham lam**

Mục tiêu nghiên cứu : Làm rõ về bản chất và tìm hiểu sự ứng dụng của thuật toán trong thực tế.Trên cơ sở lý thuyết nghiên cứu được áp dụng vào chương trình: bài toán cái túi.

## **II.2. Các thành phần của chiến lược tham lam**

Thuật toán tham lam có 5 thành phần:

1. Một tập hợp các ứng viên (candidate), để từ đó tạo ra lời giải.
2. Một hàm lựa chọn, để theo đố lựa chọn ứng viên tốt nhất để bổ sung vào lời giải.
3. Một hàm khả thi (feasibility), dùng để quyết định nếu ứng viên có thể được dùng để xây dựng lời giải.
4. Một hàm mục tiêu, ấn định giá trị của lời giải hoặc một lời giải chưa hoàn chỉnh.
5. Một hàm đánh giá, chỉ ra khi nào ta tìm ra một lời giải hoàn chỉnh.

Có 2 thành phần quyết định đến tham lam:

### -Tính lựa chọn tham lam

### - Cấu trúc con tối ưu

## **II.3. Sơ đồ thuật toán**

*procedure Greedy;*

*begin*

*C := Tập các ứng cử viên;*

*S := ∅ {S là lời giải cần xây dựng theo thuật toán}*

*while (C ≠ ∅) and not Solution(S) do*

*begin*

*x  Select(C);*

*C := C \ x;*

*if feasible(S ∪ x) then S := S ∪ x;*

*end;*

*if Solution(S) then Return S*

*end;*

## **II.4. Một số chiến lược - Tiến trình thực hiện – Các bước thực hiện thuật toán tham lam.**

Một số chiến lược tham lam:

Tham lam thường đề cập đến hai chiến lược tối ưu cục bộ cơ bản:

Chọn phương án tốt trước (‘chọn miếng ngon trước’) lý do gọi là thuật toán tham lam. Chiến lược này thường được áp dụng khi xây dựng dần từng phần của nghiệm tối ưu. Thuật toán sẽ đánh giá các lựa chọn theo một tiêu chuẩn nào đó và sắp xếp từ nhỏ tới lớn. Rồi tiến hành lựa chọn theo trình tự đó.

Cải tiến cái đang có thành cái tốt hơn. Chiến lược này thường được bắt đầu bằng 1 hay 1 vài phương án. Sau đó bằng cách thức nào đó, các phương án được điều chỉnh để có giá trị tốt hơn. Quá trình điều chỉnh dừng lại khi không điều chỉnh thêm được nữa hoặc sự cải thiện rất nhỏ hoặc hết thời gian cho phép….  phần lớn các giải thuật hiện nay áp dụng chiến lược này.

* Một cách tổng quát, thực hiện phương pháp Tham lam qua các bước:

1. Tìm lựa chọn sao cho các bước tiếp theo chỉ việc giải quyết một bài toán con
2. Chứng minh: với sự lựa chọn Tham lam tại mỗi bước  **luôn tìm được** 1 giải pháp tối ưu (cho bài toán ban đầu)
3. Chỉ ra: với sự lựa chọn Tham lam tại mỗi bước  giải pháp tối ưu của bài toán con còn lại kết hợp với sự lựa chọn Tham lam này **sẽ đi đến** một giải pháp tối ưu (cho bài toán ban đầu)

## **II.5. Chứng minh tính đúng đắn**

1. **Lập luận biến đổi (Exchange Argument)**

Giả sử cần chứng minh thuật toán A cho lời giải đúng. A(I) là lời giải tìm đ­ược bởi thuật toán A đối với bộ dữ liệu I. Còn O là lời giải tối ư­u của bài toán với bộ dữ liệu này.

Ta cần tìm cách xây dựng phép biến đổi ⱷ để biến đổi O thành O’ sao cho :

* O’ cũng tốt không kém gì O (Nghĩa là O’ vẫn tối ­ưu)
* O’ giống với A(I) nhiều hơn O.

Giả sử đã xây dựng đ­ược phép biến đổi vừa nêu. Để chứng minh tính đúng đắn dựa vào hai sơ đồ chứng minh sau

1)  **CM bằng phản chứng :**Giả sử A không đúng đắn, hãy tìm bộ dữ liệu I sao cho A(I) khác với lời giải tối ­ưu của bài toán. Gọi O là lời giải tối ưu giống với A(I) nhất => A(I) khác O. Dùng phép biến đổi ⱷ chúng ta có thể biến đổi O  O’ sao cho O’ vẫn tối ư­u và O’giống với A(I) hơn => mâu thuẫn giả thiết O là lời giải tối ­ưu giống với A(I) nhất.

2)  **CM trực tiếp :** O là lời giải tối ­u. Biến đổi O  O’ giống với A(I) hơn là O. Nếu O’ = A(I) thì A(I) chính là ph­ương án tối ư­u ngư­ợc lại biến đổi O’   O’’ giống với A(I) hơn. Cứ thế ta thu đ­ược dãy O’, O’’ ,O’’’ …..ngày càng giống hơn, và chỉ có một số hữu hạn điều kiện để so sánh nên chỉ sau một số hữu hạn lần phép biến đổi sẽ kết thúc và đó là tại A(I).

# **III- MỘT SỐ BÀI TOÁN ÁP DỤNG THUẬT TOÁN THAM LAM**

## **III.1. Bài toán người du lịch**

1. **Mô tả bài toán**

Có n thành phố được đánh số theo thứ tự từ 1 đến n. Người du lịch xuất phát từ một thành phố và ghé thăm các thành phố còn lại, mỗi thành phố thăm duy nhất một lần sau đó trở về nơi xuất phát. Biết chi phí đi lại từ thành phố i đến thành phố j là Cij. Hãy tìm hành trình có chi phí thấp nhất cho người du lịch.

1. **Ý tưởng giải quyết bài toán**

Đây là bài toán tìm chu trình có trọng số nhỏ nhất trong một đơn đồ thị vô hướng có trọng số.Thuật toán tham lam cho bài toán này là chọn thành phố có chi phí nhỏ nhất tính từ thành phố hiện thời đến các thành phố chưa qua.

*Phân tích :*

* Đầu vào: số thành phố n, chi phí tử thành phố i đến thành phố j (Cij).
* Đầu ra: Hành trình tối ưu và chi phi tương ứng

1. **Mô tả bài toán bằng thuật toán tham lam như sau:**

Không mất tính tổng quát ta giả sử người du lịch xuất phát từ thành phố 1. Mỗi chu trình đường đi TP1,TPi1, TPi2 ,TPi3,.,TPin, TP1 có thể đặt tương ứng 1-1 với một hoán vị (i1,i2,…..,in) của 2,3,….,n. Gọi C(ik-il)là chi phí đi từ thành phố ik đến thành phố il. Khi đó chi phí của một chu trình là tổng các chi phí từng chặng:

Đặt f(x) = C(1-i1)+C(i1-i2)+…..+ C(in-1 - in) C(in - 1)

Ký hiệu D là tập tất cả các hoán vị của n-1 số 2,3,….n, có thể phát biểu bài toán người du lịch dưới dạng sau:

{f(x) -> min; x thuộc D}

1. **Chương trình mã giả minh họa cho thuật toán**

*Thủ tục minh họa:*

Chu\_trinh:= rỗng;

Chi\_phi:=0;

Vi\_tri:=1;// xuất phát từ thành phố 1

For k:=1 to n-1 do//thăm tất cả các thành phố

Begin

//chọn thành phố X chưa tới sao cho chi phí C(vitri-X) nhỏ nhất

Chu\_trinh:=chu\_trinh +(vi\_tri,X);

Chi\_phi:=chi\_phi+C(vi\_tri-x);

Vi\_tri := x;

End;

//trở về nơi xuất phát

Chu\_trinh:=chu\_trinh +(vi\_tri,1);

Chi\_phi:=chi\_phi +C(vitri-1);

End.

1. **Đánh giá độ phức tạp**

Thao tác chọn đỉnh thích hợp trong n đỉnh được tổ chức bằng một vòng lặp để duyệt. Nên chi phí cho thuật toán xác định bởi hai vòng lặp lồng nhau , nên T(n) là O(n2).

## **III.2.Bài toán mã Huffman**

1. **Mô tả bài toán**

Mã Huffman là kỹ thuật được dùng phổ biến và rất hữu hiệu cho việc nén dữ liệu (data compression) tiết kiệm từ 20% đến 90% phụ thuộc vào đặc điểm của dữ liệu được nén.

Xét bài toán mã hóa ký tự. Giải thuật tham lam Huffman sử dụng bảng tần suất xuất hiện các ký tự để tạo ra 1 cách tối ưu biểu diễn mỗi kí tự như 1 chuỗi nhị phân (binary string).

Giả sử ta có một tập tin 100 ký tự cần nén. Chỉ có 6 ký tự khác nhau trong file với tần số xuất hiện như trong hình. Trong đó ký tự a xuất hiện 45 lần.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Ký tự | a | b | c | d | e | f |
| Tần số | 45 | 13 | 12 | 16 | 9 | 5 |
| Từ mã có chiều dài cố định | 000 | 001 | 010 | 011 | 100 | 101 |
| Từ mã có chiều dài bất định | 0 | 101 | 100 | 111 | 1101 | 1100 |

Có nhiều cách để biểu diễn thông tin. Giả sử biểu diễn mỗi ký tự dưới dạng 1 chuỗi nhị phân duy nhất. Nếu dùng mã chiều dài cố định (fixed length) thì cần 3 bit để biểu diễn 6 ký tự a=000, b=001…, f=101. 1 file có 100 ký tự vậy cần 300 bits để mã hóa cả file đó.

Mã có chiều dài thay đổi (variable-length code) có thể làm việc tốt hơn một mã có chiều dài cố định (fixed-length code), b tằng cách gán cho những ký tự hay xuất hiện mã ngắn và những ký tự ít xuất hiện mã dài hơn.

Ví dụ : mã hóa như sau: a=0 , b=101, …, f=1100. Khi đó sẽ cần (45\*1+13\*3+12\*3+16\*3+9\*4+5\*4)=224

* Tiết kiệm gần 25% . Đây là mã tối ưu cho bài toán

Mã phi-tiền tố (prefix-free code): những mã trong đó không có từ mã nào là tiền tố của vài từ mã khác. Điều này có thể chỉ ra rằng việc nén dữ liệu tối ưu có thể thực hiện được bằng việc mã hoá ký tự có thể luôn luôn được hoàn thành với một mã tiền tố, vì vậy không mất tính tổng quát trong việc giới hạn sự chú ý đến mã tiền tố.

Mã hóa bằng mã ký tự nhị phân: Việc mã hóa luôn luôn đơn giản đối với bất cứ mã ký tự nhị phân nào. Ta chỉ cần ghép nối các mã biểu diễn mỗi ký tự của tập tin đó với nhau.

Ví dụ:

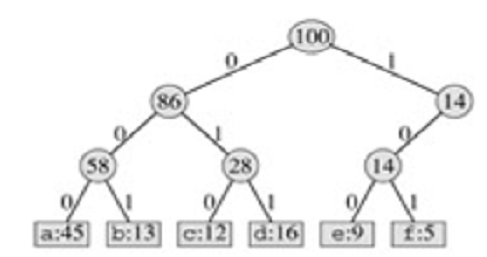
Áp dụng mã tiền tố chiều dài thay đổi để mã hóa file ký tự abc 0.101.100 =0101100 với “.” Là ký hiệu nối.

Quá trình giải mã (decoding) cần biểu diễn một cách thích hợp các mã tiền tố sao cho có thể dễ dàng khôi phục lại được các từ mã (codeword) lúc đầu.

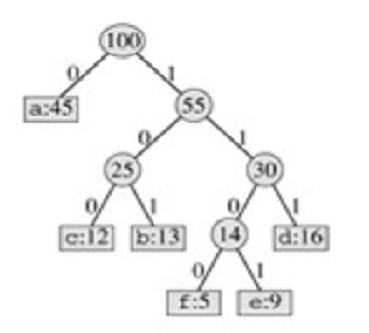
Biểu diễn mã tiền tố bằng cây nhị phân

Mã tiền tố có thể được biểu diễn dưới dạng cây nhị phân (binary tree) mà các lá của nó là các ký tự được mã hóa. Mã nhị phân của mỗi từ (binary codework) là đường đi từ gốc tới lá. Với quy ước 0 là đi đến con trái, 1 là đi đến con phải.

Ví dụ mã tiền tố bằng cây nhị phân:



Mã chiều dài cố định



Mã tiền tố bất định (mã tiền tố tối ưu)

Xây dựng mã Huffman

Huffman đã đề xuất một giải thuật **tham lam** để xây dựng một mã nhị phân tiền tố tối ưu được gọi là giải thuật mã hóa Huffman (Huffman code).

1. **Nội dung - Ý tưởng giải quyết bài toán**

* Giả sử C là tập n ký tự và mỗi ký tự c∈C có tần số xuất hiện là f[c].
* Giải thuật Huffman xây dựng cây T tương ứng với mã tối ưu theo phương thức bottom-up.
* Giải thuật bắt đầu với một tập gồm |C| lá và thực hiện một chuỗi gồm |C|-1 thao tac “merging” (trộn) để tạo ra cây cuối cùng.
* Một hàng đợi ưu tiên nhỏ nhất Q, được lập trên khoá f, được sử dụng để nhận biết 2 đối tượng có tần số nhỏ nhất để kết hợp với nhau. Kết quả của sự kết hợp 2 phần tử đó là một phần tử mới mà tần số của nó là tổng của các tần số của 2 phần tử được kết hợp.
* Cây cuối cùng biểu diễn một mã tiền tố tối ưu. Mã (codeword) của mỗi ký tự là một chuỗi các (chữ cái) các nhãn cạnh (edge labels) trên đường dẫn (path) đi từ gốc đến ký tự đó.

1. **Chương trình mã giả minh họa cho thuật toán**

*HUFFMAN(C)*

*1 n← |C|*

*2 Q←C*

*3 fori 1 ton - 1*

*4 do allocate a new node z*

*5 left[z] ←x← EXTRACT-MIN (Q)*

*6 right[z] ←y← EXTRACT-MIN (Q)*

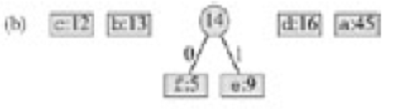
*7 f [z] ←f [x] + f [y]*

*8 INSERT(Q, z)*

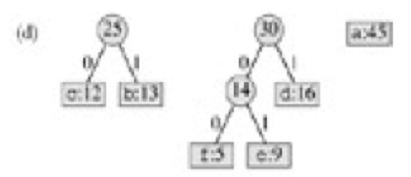
*9 return EXTRACT-MIN(Q) ▹Return the root of the tree.*

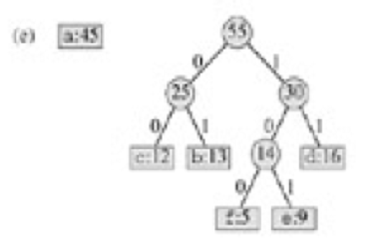
Minh họa các bước thực hiện:

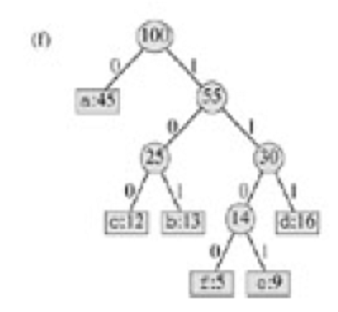












1. **Đánh giá độ phức tạp**

Giả sử Q được thực thi như 1 cây nhị phân (binary min-heap) Đối với tập C chứa n ký tự, việc khởi tạo Q được thực thi với thời gian O(n).Vòng lặp for (3-8) được thực thi chính xác n-1 lần, và vì mỗi phép toán đống yêu cầu thời gian là O(logn), nên vòng lặp đóng góp O(nlogn) vào thời gian chạy thuật toán.Vì vậy , tổng thời gian chạy của giải thuật Huffman trên tập hợp gồm n ký tự là O(nlogn)

**IV.THUẬT TOÁN THAM LAM VÀ BÀI TOÁN XẾP LỊCH CHO CÁC HOẠT ĐỘNG HAY LẬP LỊCH CÔNG VIỆC**

1. **Mô tả bài toán**

* Xét S = {a1,a2,..an } là tập các hoạt động muốn sử dụng tài nguyên (vd:hội trường)
* Mỗi hoạt động ai sẽ có thời điểm bắt đầu là si và thời điểm kết thúc là fi, với điều kiện 0 ≤ si < fi < ∞. Nếuhoạt động ai đượcchọn, thì nó sẽ độc chiếm tài nguyên trong khoảng thời gian [si, fi). Hoạt động ai và aj đượcgọi là tương thích lẫn nhau nếu như khoảng thời gian [si, fi) và [sj, fj) là không giao nhau.
* Bài toán xếp lịch xếp lịch cho các hoạt động (hay còn gọi là bài toán lập lịch công việc) là một bài toán tối ưu tổ hợp. Bài toán được đặt tên từ vấn đề tìm ra được số task có thể hoàn thành nhiều nhất mỗi lần thực hiện công việc.

Một số cách phát biểu nội dung bài toán:

Ngày CN, Minh có rất nhiều dự định muốn hoàn thành chẳng hạn như làm bài tập lớn môn A, viết bài viblo, ôn tập môn B, nghe nhạc, chạy bộ, ... Mỗi công việc đều có thời gian lý tưởng để bắt đầu và kết thúc của nó. Tuy nhiên có một số công việc không thể cùng hoàn thành bởi công việc trước chưa kết thúc công việc sau đã bắt đầu. Chẳng hạn

|  |  |  |
| --- | --- | --- |
| **Task** | **start** | **end** |
| làm btl môn A | 8 (h) | 10 (h) |
| viết bài viblo | 9 | 11 |
| ôn tập môn B | 10 | 12 |
| nghe nhạc | 14 | 15 |
| chạy bộ | 17 | 18 |

Chúng ta không thể hoàn thành cả 2 task là viết bài viblo và làm btl môn A hay ôn tập môn B. Nếu chọn thực hiện viết bài viblo bạn chỉ có thể hoàn thành 3/5 task trong khi nếu không chọn viết bài viblo bạn có thể hoàn thành 4/5 task.  
Chúng ta cần giúp Minh tìm ra được số task có thể hoàn thành nhiều nhất mỗi khi ngày CN tới.

1. **Yêu cầu:** Mỗi thời điểm chỉ có 1 hoạt động sử dụng tài nguyên chung.
2. **Mục tiêu:** Chọn được 1 tập lớn nhất các hoạt động tương thích với nhau (khoảng thời gian thực hiện không giao nhau)=>Tận dụng tối đa tài nguyên.

*Ví dụ ai và aj là tương thích nếu si ≥ fj hoặc sj ≥ fi*

4. **Phân tích bài toán**

Ý tưởng bài toán: Dễ thấy để chọn được ra nhiều task nhất ta sẽ chọn ra task có thời gian kết thúc sớm nhất và đảm bảo thời gian bắt đầu của nó muộn hơn thời gian kết thúc của các task được chọn trước đó. Riêng với task đầu tiên ta chỉ cần chọn ra task kết thúc sớm nhất

5. Ý tưởng bài toán

Xét một bài toán con khác rỗngSij,và nếu am là một hoạt động trong Sij có thời điểm kết thúc sớm nhất:

fm = min{fk : ak ϵSij}.Thì:

* Hoạt động am được sử dụng trong một tập con lớn nhất nào đó của các hoạt động tương thích lẫn nhau của Sij.
* Bài toán con Sim là rỗng, do đó nếu chọn am thì chỉ còn duy nhất bài toán con khác rỗngSmj.

Mục tiêu:

* + Giảm số các bài toán con và số cách chọn:
    - Chỉ duy nhất một bài toán con được sử dụng trong giải pháp tối ưu (một bài toán con khác rỗng)
    - Chỉ cần 1 chọn lựa cho bài toán con: chọn hoạt động nào có thời gian kết thúc sớm nhất trong Sij (dễ dàng).
  + Có thể giải mỗi bài toán con theo pp top down (thay vì bottom up trong lập trình động).
    - Vì khi chọn am chắc chắn lời giải Smj sẽ được dùng trong lời giải tối ưu của Sij ->không cần giải Smj trước khi giải Sij
* Để giải bài toán con Sij ,đầu tiên chọn hoạt động am trong Sij có thời gian kết thúc sớm nhất rồi mới tìm lời giải cho bài toán con Smj .

5. **Phân tích chiến lược lựa chọn tham lam:**

Việc lập lịch công việc có thể theo 2 chiến lược sau:

-Ưu tiên công việc có thời gian hoàn thành ít hơn trước

-Ưu tiên công việc có mốc thời gian sớm hơn trước

1. Chương trình minh họa thuật toán

Input:

* Mảng s và f biểu diễn thời điểm bắt đầu và kết thúc của các hoạt động
* Giả sử n hoạt động đã xếp theo thứ tự tăng dần của thời điểm kết thúc.Theo CT(16.1):

f0 ≤ f1 ≤ f2 ≤...≤fn ≤ fn+1

Output:

* Trả về một tập lớn nhất của các hoạt động tương thích với nhau trong Si,n+1
* Chương trình :

*RECURSIVE-ACTIVITY-SELECTOR (s ,f ,i, n)*

*mi+1*

***while*** *m < n and sm < fi Tìm hoạt động đầu tiên trong Si,n+1*

***do*** *m m+1*

***if*** *m < n*

***then*** *return {am} U* ***RECURSIVE-ACTIVITY-SELECTOR****(s ,f ,m ,n)*

***else*** *return Ø*

1. **Đánh giá độ phức tạp**

Thời gian chạy RECURSIVE-ACTIVITY-SELECTOR (s,f,0,n) là O(n).

**V. BÀI TẬP VÍ DỤ VỀ BÀI TOÁN LẬP LỊCH CÔNG VIỆC**

**Bài toán :**

Cho n việc cần phải hoàn thành, mỗi việc thực hiện trong 1 đơn vị thời gian. Việc i sẽ đem lại wi tiền thưởng nếu hoàn thành đúng hạn đi. Tìm cách thực hiện các công việc để có lợi nhuận cao nhất mà thời gian thực hiện là cho trước.

Ví dụ: Tạo file có dạng:

8 //số lượng công việc

12 //thời gian giới hạn

1 20 4 //tên công việc- lương - số tháng hoàn thành

2 40 2

3 50 5

4 40 6

5 90 3x

6 60 7

7 10 9

8 30 2

Code giải thuật tham lam với Dev-Cpp dùng cho bài tập này:

*#include<stdio.h>*

*#include<conio.h>*

*#include<math.h>*

*#include<string.h>*

*typedef struct{*

*char ten[100];*

*float gt,tg,tp;*

*}dv;*

*void sx( dv a[], int m) //hàm sắp xếp theo tỉ lệ giá trị/thời gian*

*{*

*int i,j;*

*dv dc;*

*for( i=0; i< m; i++)*

*for(j=i+1; j<m; j++ )*

*{*

*if(a[i].tp<a[j].tp)*

*{*

*dc=a[i];*

*a[i]=a[j];*

*a[j]=dc;*

*}*

*}*

*}*

*main ()*

*{*

*dv a[100];*

*FILE\*f;*

*f=fopen("congviec.inp","r");*

*int i,m;*

*float n, kt=0,gt=0;*

*fscanf(f,"%d",&m);*

*fscanf(f,"%f",&n);*

*for(i=0; i<m; i++) //nhập dữ liệu*

*{*

*fscanf(f,"%s",&a[i].ten);*

*fscanf(f,"%f",& a[i].gt);*

*fscanf(f,"%f",& a[i].tg);*

*a[i].tp= a[i].gt/a[i].tg;*

*}*

*fclose(f);*

*for(i=0; i<m; i++) //kiểm tra dữ liệu*

*{*

*if(a[i].tp<0||n<=0)*

*{*

*kt++;*

*printf("du lieu sai");*

*break;}}*

*if(kt==0)*

*{*

*sx(a,m);*

*printf ("cong viec sap xep: ");*

*for( i=0; i< m; i++)*

*printf(" %s;",a[i].ten);*

*printf ("\n cong viec se lam:"); //đáp án*

*for(i=0; i<m; i++)*

*{*

*if((kt+a[i].kl)<n)*

*{*

*kt+=a[i].tg;*

*gt+=a[i].gt;*

*printf(" %s ",a[i].ten);*

*}*

*}*

*printf("\n thoi gian toi da: %.2f thang",kt);*

*printf("\n gia tri lon nhat la: %.2f trieu VND",gt);*

*}*

*getch();*

*}*

* Đáp án: 5 2 8 1, thời gian tối đa là: 11 tháng để có giá trị max là 180.

**TÀI LIỆU THAM KHẢO:**

* Phương pháp tham lam đầu tiên xuất hiện trong tài liệu tối ưu tố hợp trong một bài báo năm 1971 của Edmonds [85], qua lý thuyết về matroids có trong bài báo năm1935 của Whitney [314].
* Giải thuật mã hoá Huffman dược phát minh năm 1952[162]; Lelewer and  
  Hirschberg .
* Lý thuyết mở rộng về matroid trong lý thuyết tham lam được khám phả đầu tiên bởi Korte và Lovasz.
* Giáo trình:” Cấu trúc dữ liệu và giải thuật “ khoa công nghệ thông tin- đại học HÀNG HẢI xuất bản 2014-2015.
* Một số nguồn trên internet.