1. **GIỚI THIỆU CHUNG VỀ THUẬT TOÁN THAM LAM**
2. **Ý tưởng**

Phương pháp tham lam là kỹ thuật thiết kế thường được dung để giải các

bài toán tối ưu. Phương pháp được tiến hành trong nhiều bước. Tại mỗi bước, theo một lựa chọn nào đó ( xác định bằng một hàm chọn), sẽ tìm ra 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?

Các lời giải tìm được bằng phương pháp tham lam thường là chấp nhận

được theo điều kiện nào đó, chưa chắc là tối ưu.

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 với giá trị nhỏ nhất ( lớn nhất).

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

1. **Mô hình**

Chọn S từ tập A.

Tính chất tham lam của thuật toán định hướng bởi hàm Chọn.

* Khởi động S = ∅;
* Trong khi A ≠ ∅:
  + Chọn phần tử tốt nhất của A gán vào x: x= Chọn(A);
  + Cập nhật các đối tượng để chọn: A = A – {x};
  + Nếu S∪{x} thỏa mãn yêu cầu bài toán thì
    - Cập nhật lời giải: S = S∪{x};

Thủ tục thuật toán tham lam có thể cài đặt như sau:

Input A[1..n]

Output S//lời giải

Greedy (A,n) ≡ S = ∅;

while ( A ≠ ∅)

{

x = Chọn(A);

A = A-{x};

If(S∪{x}chấp nhận được)

S = S∪{x};

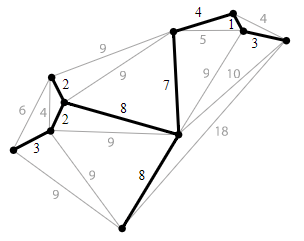
}

Return S;

1. **THUẬT TOÁN KRUSKAL – TÌM CÂY BAO TRÙM NHỎ NHẤT**
2. **Bài toán**

Với một đồ thị liên thông, vô hướng cho trước, cây bao trùm của nó là một đồ thị con có dạng cây và có tất cả các đỉnh liên thông với nhau. Một đồ thị có thể có nhiều cây bao trùm khác nhau. Chúng ta gán một trọng số cho mỗi cạnh. Khi đó một **cây bao trùm nhỏ nhất** là một cây bao trùm có tổng trọng số trên các cạnh bé hơn hoặc bằng tổng trọng số của tất cả các cây bao trùm khác.

Tổng quát hơn, bất kỳ một đồ thi vô hướng nào (không nhất thiết phải liên thông) đều có một **rừng bao phủ nhỏ nhất**, là hội của các cây bao trùm nhỏ nhất của các thành phần liên thông của nó.

Hình 1: Hình ảnh thể hiện một cây bao trùm nhỏ nhất trong một đồ thị.

2. Tư tưởng bài toán

Thuật toán Kruskal dựa trên mô hình xây dựng cây bao trùm bằng thuật toán hợp nhất

-Luôn djuy trì một rừng cây đã thăm

-Mỗi bước lựa chọn một cạnh nhỏ nhât để gộp 2 cây lại thành một cây

Kruskal cần một sử dụng cấu trúc dữ liệu **union-find** để hỗ trợ. Cấu trúc dữ liệu cần đảm bảo các yêu cầu sau:

makeset(u) : tạo một tập {u} trong O(1)

find(u) : trả lại tập chứa đỉnh u trong O(1)

union(u,v) : hợp 2 tập chứa đỉnh u và v trong O(1)

3. Mô tả thuật toán:

Giả sử ta cần tìm cây bao trùm nhỏ nhất của đồ thị G. Thuật toán bao gồm các bước sau:

* Khởi tạo rừng F (tập hợp các cây), trong đó mỗi đỉnh của G tao thành một cây riêng biệt.
* Khởi tạo tập S chứa tất cả các cạnh của G
* Chừng nào S còn khác rỗng và F gồm hơn một cây
* Xóa cạnh nhỏ nhất trong S
* Nếu cạnh đó nối hai cây khác nhau trong F thì them nó vào F và hợp nhất hai cây kề với nó làm một.
* Nếu không thì loại bỏ cạnh đó

Kỹ thuật đánh nhãn đỉnh:

* Ngay tại bước 1 của thuật toán, ta gắn đỉnh I của đồ thj một nhãn la i
* Trong bước 2:
  + Nếu hai đầu cạnh e có cùng nhãn (tức là nhãn của e.v1 và nhãn của e.v2 bằng nhau) thì T+{e} sẽ tạo chu trình, ta không đưa e vào T
  + Ngược lại thì ta đưa e vào T và thực hiện công việc ghép nhãn bằng cách:
    - Label1 = Min(Label(e.v1), Label(e.v2))
    - Label2 = Max(Label(e.v1), Label(e.v2))
    - Sửa nhãn của tất cả các đỉnh nào có nhãn là Label2 thành nhãn Label1

**Cài đặt**

+ Mô tả G bằng ma trận có trọng số cạnh c[i,j].

+ D là mảng 1 chiều , nếu Di =k thì đỉnh i đang xét thuộc vào cây thứ k, nếu Di=0 thì đỉnh I đang xét đó chưa thuộc cây nào

+ Tìm min c[i][j] , trừ các cạnh mà tại đó tạo thành chu trình

+ Thêm cạnh vừa tìm vào cây T

-Nếu Di=Dj=0 , cạnh (i,j) chưa thuộc vào cây nên khi lấy 2 đỉnh này vào tập cạnh, ta cho chúng thuộc vào 1 cây mới , khi đó k=k+1; và Di=Dj=k

-Nếu Di=0 và Dj!=0, khi đó I chưa thuộc cây T, j đã thuộc cây T rồi, , ta gép I vào cùng cây chưa j , Di=Dj

- Nếu Dj=0 và Di!=0 thì khi đó j chưa thuộc cây T, I đã thuộc cây T rồi, ta gép j vào cùng cây chứa I, Dj=Di;

-Nếu Di!=Dj và Di!=0,Dj!=0 thì j thuộc 2 cây khác nhau trong T , ta gép 2 cây này thành 1

|  |
| --- |
| void Kruskal(){  int\*D=new int[n];  Egde\*L= new Egde[n-1];  int min, Dem=0,Sum=0,T=0,Temp;  for(int i=0;i<n;i++)  D[i]=0;  do{  min=32767;  for(int i=0;i<n;i++)  for(int j=0;j<n;j++)  if(c[i][j]>0 &&min>c[i][j]&&!(D[i]!=0&&D[i]==D[j])){  min=c[i][j];  L[Dem].x=i;  L[Dem].y=j;  }  //Tao cay moi  if(D[L[Dem].x]==0&&D[L[Dem].y]==0){  T++;  D[L[Dem].x]=D[L[Dem].y]=T;    }  // Dua dinh tuong ung vao cay  if(D[L[Dem].x]==0&&D[L[Dem].y]!=0)  D[L[Dem].x]=D[L[Dem].y];  // Dua dinh tuong ung vao cay  if(D[L[Dem].x]!=0 && D[L[Dem].y]==0)  D[L[Dem].y]=D[L[Dem].x];  // Gep 2 cay thanh 1 cay moi  if(D[L[Dem].x]!=D[L[Dem].y]&&D[L[Dem].y]!=0){  Temp=D[L[Dem].x];  for(int i=0;i<n;i++)  if(Temp==D[i])  D[i]=D[L[Dem].y];  }  Sum+=min;  Dem++;    }  while(Dem<n-1);  printf("\nTrong so cua cay khung con nho nhat la: ");  printf(" %d",Sum);  printf("\nCay khung con nho nhat la: ");  for(int i=0;i<n-1;i++)  printf(" \n(%d - %d)\n",L[i].x+1,L[i].y+1);  } |

**III.Thực hiện các bước của thuật toán**

***Bộ dữ liệu 1:***

8

6

2

5

4

2

2

6

2

4

10

5

8

4

1

4

1

10

3

7

3

3

3

9

7

1

**Bước 1:** Khởi tạo T = φ, D(T) = 0

**Bước 2:** Sắp xếp các cạnh theo chiều không giảm trọng số

|  |  |  |
| --- | --- | --- |
| Điểm đầu | Điểm cuối | Trọng số |
| 3 | 4 | 1 |
| 7 | 9 | 1 |
| 1 | 2 | 2 |
| 2 | 4 | 2 |
| 6 | 8 | 2 |
| 1 | 3 | 3 |
| 4 | 7 | 3 |
| 9 | 10 | 3 |
| 2 | 3 | 4 |
| 4 | 5 | 4 |
|  |  |  |
| 7 | 8 | 4 |
| 5 | 7 | 5 |
| 6 | 10 | 6 |
| 8 | 9 | 7 |
| 5 | 6 | 8 |
| 6 | 9 | 10 |

**Bước 3:** Lặp

|  |  |  |
| --- | --- | --- |
| STT | Cạnh được xét | T ∪ e |
| 1 | E\(3,4) | T ∪ (3,4); D(T) = 1 |
| 2 | E\(7,9) | T = T ∪ (7,9); D(T) = 1 + 1 = 2 |
| 3 | E\(1,2) | T = T ∪ (1,2); D(T) = 2 + 2 = 4 |
| 4 | E\(2,4) | T = T ∪ (2,4); D(T) = 4 + 2 = 6 |
| 5 | E\(6,8) | T = T ∪ (6,8); D(T) = 6 + 2 = 8 |
| 6 | E\(1,3) | Tao thanh chu trinh |
| 7 | E\(4,7) | T = T ∪ (4,7); D(T) = 8 + 3 = 11 |
| 8 | E\(9,10) | T = T ∪ (9,10); D(T) = 11 + 3 = 14 |
| 9 | E\(2,3) | Tao thanh chu trinh |
| 10 | E\(4,5) | T = T ∪ (4,5); D(T) = 14 + 4 = 18 |
| 11 | E\(7,8) | T = T ∪ (7,8); D(T) = 18 + 4 = 22 |

Thuật toán dừng lại khi ta đã tìm được 9 cạnh.

Trong so cua cay khung con nho nhat la: 22

***Bộ dữ liệu 2:***

12

20

6

8

7

3

1

5

4

11

9

3

1

3

8

6

10

4

2

5

5

2

15

9

6

2

**Bước 1:** Khởi tạo T = φ, D(T) = 0

**Bước 2:** Sắp xếp các cạnh theo chiều không giảm trọng số

|  |  |  |
| --- | --- | --- |
| Điểm đầu | Điểm cuối | Trọng số |
| 3 | 5 | 1 |
| 4 | 5 | 2 |
| 6 | 9 | 2 |
| 2 | 3 | 3 |
| 3 | 4 | 3 |
| 4 | 7 | 4 |
| 7 | 10 | 5 |
| 1 | 3 | 6 |
| 6 | 10 | 6 |
| 8 | 9 | 11 |
| 7 | 8 | 12 |
| 5 | 6 | 15 |
| 3 | 7 | 20 |

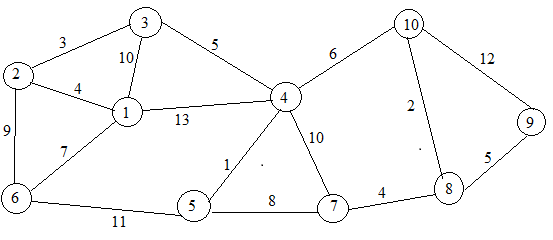
**Bước 3:** Lặp

|  |  |  |
| --- | --- | --- |
| STT | Cạnh được xét | T ∪ e |
| 1 | E\(3,5) | T ∪ (3,5); D(T) = 1 |
| 2 | E\(4,5) | T = T ∪ (4,5); D(T) = 1 + 2 = 3 |
| 3 | E\(6,9) | T = T ∪ (6,9); D(T) = 3 + 2 = 5 |
| 4 | E\(2,3) | T = T ∪ (2,3); D(T) = 5 + 3 = 8 |
| 5 | E\(3,4) | Tao thanh chu trinh |
| 6 | E\(4,7) | T = T ∪ (4,7); D(T) = 8 + 4 = 12 |
| 7 | E\(7,10) | T = T ∪ (7,10); D(T) = 12 + 5 = 17 |
| 8 | E\(1,3) | T = T ∪ (1,3); D(T) = 17 + 6 = 23 |
| 9 | E\(6,10) | T = T ∪ (6,10); D(T) = 23 + 6 = 29 |
| 10 | E\(8,9) | T = T ∪ (8,9); D(T) = 29 + 11 = 40 |

Thuật toán dừng lại khi ta đã tìm được 9 cạnh.

Trong so cua cay khung con nho nhat la: 40

***Bộ dữ liệu 3:***

******

**Bước 1:** Khởi tạo T = φ, D(T) = 0

**Bước 2:** Sắp xếp các cạnh theo chiều không giảm trọng số

|  |  |  |
| --- | --- | --- |
| Điểm đầu | Điểm cuối | Trọng số |
| 4 | 5 | 1 |
| 8 | 10 | 2 |
| 2 | 3 | 3 |
| 1 | 2 | 4 |
| 7 | 8 | 4 |
| 3 | 4 | 5 |
| 8 | 9 | 5 |
| 4 | 10 | 6 |
| 1 | 6 | 7 |
| 5 | 7 | 8 |
| 2 | 6 | 9 |
| 1 | 3 | 10 |
| 9 | 10 | 12 |

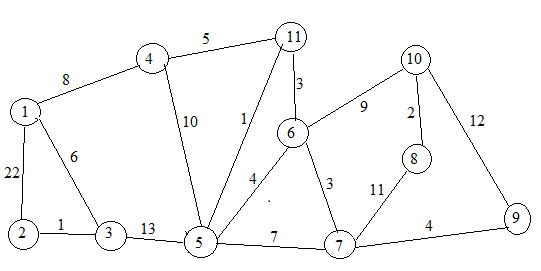
**Bước 3:** Lặp

|  |  |  |
| --- | --- | --- |
| STT | Cạnh được xét | T ∪ e |
| 1 | E\(4,5) | T ∪ (4,5); D(T) = 1 |
| 2 | E\(8,10) | T = T ∪ (8,10); D(T) = 1 + 2 = 3 |
| 3 | E\(2,3) | T = T ∪ (2,3); D(T) = 3 + 3 = 6 |
| 4 | E\(1,2) | T = T ∪ (1,2); D(T) = 6+4 = 10 |
| 5 | E\(7,8) | T = T ∪ (7,8); D(T) = 10+4 = 14 |
| 6 | E\(3,4) | T = T ∪ (3,4); D(T) = 14+5=19 |
| 7 | E\(8,9) | T = T ∪ (8,9); D(T) = 19+5=24 |
| 8 | E\(4,10) | T = T ∪ (4,10); D(T) = 24+6=30 |
| 9 | E\(1,6) | T = T ∪ (1,6); D(T) = 30+7=37 |

Thuật toán dừng lại khi ta đã tìm được 9 cạnh.

Trong so cua cay khung con nho nhat la: 37

***Bộ dữ liệu 4:***

******

**Bước 1:** Khởi tạo T = φ, D(T) = 0

**Bước 2:** Sắp xếp các cạnh theo chiều không giảm trọng số

|  |  |  |
| --- | --- | --- |
| Điểm đầu | Điểm cuối | Trọng số |
| 2 | 3 | 1 |
| 5 | 11 | 1 |
| 8 | 10 | 2 |
| 6 | 7 | 3 |
| 6 | 11 | 3 |
| 7 | 9 | 4 |
| 4 | 11 | 5 |
| 1 | 3 | 6 |
| 5 | 7 | 7 |
| 1 | 4 | 8 |
| 6 | 10 | 9 |
| 4 | 5 | 10 |
| 7 | 8 | 11 |
| 3 | 5 | 13 |
| 1 | 2 | 22 |

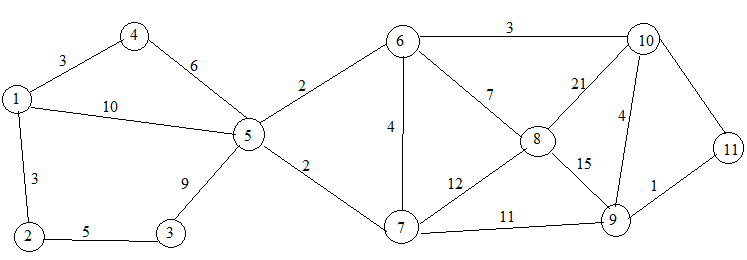
**Bước 3:** Lặp

|  |  |  |
| --- | --- | --- |
| STT | Cạnh được xét | T ∪ e |
| 1 | E\(2,3) | T ∪ (2,3); D(T) = 1 |
| 2 | E\(5,11) | T = T ∪ (5,11); D(T) = 1 + 1 = 2 |
| 3 | E\(8,10) | T = T ∪ (8,10); D(T) = 2+2=4 |
| 4 | E\(6,7) | T = T ∪ (6,7); D(T) = 4+3=7 |
| 5 | E\(6,11) | T = T ∪ (6,11); D(T) = 7+3=10 |
| 6 | E\(7,9) | T = T ∪ (7,9); D(T) = 10+4=14 |
| 7 | E\(4,11) | T = T ∪ (4,11); D(T) = 14+5=19 |
| 8 | E\(1,3) | T = T ∪ (1,3); D(T) = 19+6=25 |
| 9 | E\(5,7) | Tao thanh chu trinh |
| 10 | E\(1,4) | T = T ∪ (1,4); D(T) = 25+8=33 |
| 11 | E\(6,10) | T = T ∪ (6,10); D(T) = 33+9=42 |

Thuật toán dừng lại khi ta đã tìm được 10 cạnh.

Trong so cua cay khung con nho nhat la: 42

***Bộ dữ liệu 5:***

******

**Bước 1:** Khởi tạo T = φ, D(T) = 0

**Bước 2:** Sắp xếp các cạnh theo chiều không giảm trọng số

|  |  |  |
| --- | --- | --- |
| Điểm đầu | Điểm cuối | Trọng số |
| 9 | 11 | 1 |
| 5 | 6 | 2 |
| 5 | 7 | 2 |
| 1 | 2 | 3 |
| 1 | 4 | 3 |
| 6 | 10 | 3 |
| 9 | 10 | 4 |
| 2 | 3 | 5 |
| 4 | 5 | 6 |
| 6 | 8 | 7 |
| 3 | 5 | 9 |
| 1 | 5 | 10 |
| 7 | 9 | 11 |
| 7 | 8 | 12 |
| 8 | 10 | 21 |

**Bước 3:** Lặp

|  |  |  |
| --- | --- | --- |
| STT | Cạnh được xét | T ∪ e |
| 1 | E\(9,11) | T ∪ (9,11); D(T) = 1 |
| 2 | E\(5,6) | T = T ∪ (5,6); D(T) = 1 + 2= 3 |
| 3 | E\(5,7) | T = T ∪ (5,7); D(T) = 3+2=5 |
| 4 | E\(1,2) | T = T ∪ (1,2); D(T) = 5+3=8 |
| 5 | E\(1,4) | T = T ∪ (1,4); D(T) = 8+3=11 |
| 6 | E\(6,10) | T = T ∪ (6,10); D(T) = 11+3=14 |
| 7 | E\(9,10) | T = T ∪ (6,10); D(T) = 14=4=18 |
| 8 | E\(2,3) | T = T ∪ (2,3); D(T) = 18+5=23 |
| 9 | E\(4,5) | T = T ∪ (4,5); D(T) = 23+6=29 |
| 10 | E\(6,8) | T = T ∪ (6.8); D(T) = 29+7=36 |

Thuật toán dừng lại khi ta đã tìm được 10 cạnh.

Trong so cua cay khung con nho nhat la: 36

**IV. Độ phức tạp của thuật toán:**

Nếu E là số cạnh và V là số đỉnh của đồ thị thì thuật toán Kruskal chạy trong thời gian O(E\*log(V))

Có thể đạt được thời gian này bằng phương pháp sau: sắp xếp tất cả các cạnh theo trọng số trong thời gian O(E\*log(E)). Điều này cho phép thực hiện “xóa cạnh nhỏ nhất trong S” trong thời gian O(1). Sau đó sử dụng cấu trúc dữ liệu union-find để lưu trữ thông tin đỉnh nào nằm ở tập nào. Ta cần thực hiện O(E) thao, hai thao tác tìm và một thao tác hợp cho mỗi cạnh. Như vậy tổng thời gian là O(E\*log(E)) = O(E\*log(V))