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 PRIM – TÌM CÂY BAO TRÙM NHỎ NHẤT**
2. **Bài toán**

G = (V,E) là đơn đồ thị vô hướng lien thông, có trọng số.

V = {1,..,n} là tập các đỉnh. E là tập các cạnh (edge).

Một cây T gọi là cây bao trùm của G nếu T là đồ thị con của G và chứa mọi đỉnh của G.

Vấn đề là tìm cây bao trùm có trọng số nhỏ nhất: MST (Minimal Spanning Tree) của G.

Các thuật toán cơ bản giải bài toán trên là các thuật toán Prim và Ksuskal. Trong phần này, ta giới thiệu thuật toán Prim.

1. **Ý tưởng**

Thuật toán Prim xây dựng một đồ thị con T của G như sau:

* Đầu tiên chọn tùy ý 1 đỉnh của G đặt vào T.
* Quá trình sau còn thực hiện trong khi T chưa chứa hết các đỉnh của G:
* Mỗi bước, tìm một cạnh có trọng số nhỏ nhất nối 1 đỉnh trong T với 1 đỉnh ngoài T. Thêm cạnh này vào T.
* Kết thúc thuật toán Prim cho ta một cây bao trùm có trọng số nhỏ nhất (MST) của đồ thị G.

Tính tham lam của thuật toán Prim là tại mỗi bước them vào T một cạnh cá trọng số nhỏ nhất nối một đỉnh trong T và một đỉnh ngoài T.

1. **Mô tả thuật toán**

Input G = (V,E)

Output T =(V,.) là MST của G.

Mô tả:

* Gọi U là tập con của V.
* Khởi động T = (U,.) = ∅; //đồ thị con rỗng.
* Khởi động U = {1}; //chọn đỉnh 1 đặt vào T.
* Trong khi U ≠ V

Tìm cạnh (u,v) có trọng số nhỏ nhất, với u ∈ U và v ∉ U. Thêm đỉnh v này vào U. Thêm (u,v ) vào T.

Lời giải của bài toán là lời giải tối ưu.

*Minh họa:*

Xét đồ thị sau:

3

6

2

8

10

2

1

5

4

6

5

3

4

1

2

Áp dụng thuật toán Prim, bắt đầu tứ đỉnh 1, ta xây dựng được 1 MST của đồ thì trên:

5

4

2

2

6

1

1

3

5

2

3

Hoạt động của thuật toán:

|  |  |  |
| --- | --- | --- |
| Bước | Các cạnh chọn | Tập U |
| 0 | - | {1} |
| 1 | (3,1) | {1,3} |
| 2 | (2,3) | {1,3,2} |
| 3 | (4,2) | {1,3,2,4} |
| 4 | (5,4) | {1,3,2,4,5} |
| 5 | (6,5) | {1,3,2,4,5,6} |

1. **Cài đặt**

Để tiến hành cài đặt thuật toán, ta cần mô ta dữ liệu:

Đồ thị có thể biểu diễn bởi 1 ma trận kề của nó:

.

c[i][j] =

Khi tìm cạnh có trọng số nhỏ nhất nối 1 đỉnh trong U và một đỉnh ngoài U tại mỗi bước, ta sẽ dung 2 mảng để lưu trữ:

* Mảng closest[] : // ==U

Với I V\U thì closest[i] U là đỉnh kề gần I nhất.

* Mảng Lowcost [i] cho trọng số của cạnh (i, Closest[i]).

Tại mỗi bước, ta duyệt mảng lowcost để tìm Closest[k] U sao chotrọng số (k, Closest[k]) = Lowcost [k] là nhỏ nhất. Khi tìm được, ta in cạnh (Closest [k], k), cập nhật vào các mảng Closest và Lowcost, và có k đã thêm vào U. Khi mà ta tìm được một đỉnh k cho cây bao trùm, ta cho Lowcost[k] = , là một giá trị rất lớn, lớn hơn bất kỳ trọng số nào của đồ thị, như vậy đỉnh này sẽ không được kéo dai trong U.

|  |
| --- |
| void Prim(){  int Lowcost[MAX];  int Closest [MAX];  int i,j,k,min=0;  int U [MAX];// mang luu cac dinh da xet  int top=1; U[top] = 1;  cout << "Cay bao trum nho nhat:" << endl;  for (i=2; i<=n;i++){  Lowcost[i] = a[1][i];  Closest[i] = 1;  chuaxet[i] = TRUE;  }  chuaxet[1] = FALSE;  for (i=2;i<=n;i++){  min = Lowcost[2];  k = 2;  for (j=2;j<=n;j++){  if (Lowcost[j]<min){  min = Lowcost [j];  k = j;  }  }  chuaxet[k]=FALSE;  weight = weight + min;  cout << "\nCanh: (" << Closest[k] << "," << k << ")  \n co trong so: "<<Lowcost[k]<<endl;  Lowcost [k]= MAX;  for(j=2;j<=n;j++)  if((chuaxet[j]==TRUE)&&a[k][j]<Lowcost[j]){  Lowcost[j] = a[k][j];  Closest[j]=k;  }  top++;  U[top] = k;  cout<<"Cac dinh da xet:";  for (int tmp =1; tmp<=top; tmp++) cout<<U[tmp]<<" ";  cout<<endl;  }  cout<<"\nTong trong so cay bao trum nho nhat:"<< weight <<endl;  } |

1. **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

Các bước của thuật toán:

* Bước 1: (Khởi tạo)

U := {1}; T := ;

* Bước 2: (bước lặp)
  + B1: Tập U={1}; J={2,3} //J là tập các đỉnh kề với các đỉnh trong U, J V – U.

Trọng số các cạnh (1,2) = 2; (1,3) = 3;

chọn cạnh (1,2)

U = U {2} ={1,2};

T = T {(1,2)}={(1,2)};

* + B2: U ={1,2}; J={3,4}

Trọng số các cạnh: (1,3)=3; (2,3)=4;(2,4)=2

Chọn cạnh (2,4)

U = U {4} ={1,2,4};

T = T {(4,2)}= {(1,2),(2,4)};

* + B3: U = {1,2,4}; J = {3,5,7};

Trọng số các cạnh: (1,3)=3; (2,3)=4;(3,4)=1; (4,5)=4; (4,7)=3

Chọn cạnh (3,4)

U = U {3} ={1,2,4,3};

T = T {(4,3)}= {(1,2),(2,4),(4,3)};

* + B4: U = {1,2,4,3}; J = {5,7};

Trọng số các cạnh: (4,7)=3; (4,5)=4;

Chọn cạnh (4,7)

U = U {7} ={1,2,4,3,7};

T = T {(4,7)}= {(1,2),(2,4) ),(4,3) ),(4,7)};

* + B5: U = {1,2,4,3,7}; J = {5,8,9};

Trọng số các cạnh: (4,5)=4; (7,5)=5;(7,8)=4;(7,9)=1

Chọn cạnh (9,7)

U = U {9} ={1,2,4,3,7,9};

T = T {(9,7)}= {(1,2),(2,4) ),(4,3) ),(4,7),(9,7)};

* + B6: U = {1,2,4,3,7,9}; J = {5,6,8,10};

Trọng số các cạnh:

(4,5)=4; (7,5)=5;(7,8)=4;(9,6)=10;(9,8)=7;(9,10)=3

Chọn cạnh (10,9)

U = U {10} ={1,2,4,3,7,9,10};

T = T {(10,9)}= {(1,2),(2,4) ),(4,3) ),(4,7),( 9,7),(10,9)};

* + B7: U ={1,2,4,3,7,9,10}; J = {5,6,8};

Trọng số các cạnh:

(4,5)=4; (7,5)=5;(7,8)=4;(9,6)=10;(9,8)=7;(10,6)=6

Chọn cạnh (5,4)

U = U {5} ={1,2,4,3,7,9,10,5};

T=T {(5,4)}= {(1,2),(2,4) ),(4,3),(4,7,(7,9),(9,10) ),(5,4)};

* + B8: U ={1,2,4,3,7,9,10,5};J = {6,8};

Trọng số các cạnh: (5,6)=8; (7,8)=4;(9,8)=7;(9,6)=10,(10,6)=6

Chọn cạnh (7,8)

U = U {8} ={1,2,4,3,7,9,10,5,8};

T = T {(7,8)}

= {(1,2),(2,4),(4,3) ),(4,7),(5,4),(7,9),(9,10),(7,8)};

* + B9: U ={1,2,4,3,7,5,9,10,8};J = {6};

Trọng số các cạnh: (5,6)=8;(8,6) =2; (9,6)=10;(10,6)=6

Chọn cạnh (8,6)

U = U {6} ={1,2,4,3,7,9,10,5,8,6};

T = T {(8,6)}

= {(1,2),(2,4),(4,3) ),(4,7),(5,4),(7,9),(9,10),(7,8),(8,6)};

U=Vkết thúc

2

6

5

2

2

4

2

10

8

4

1

4

1

3

3

1

7

3

9

Tóm tắt:

|  |  |  |
| --- | --- | --- |
| Bước | Các cạnh chọn | Tập U |
| 0 | - | {1} |
| 1 | (1, 2) | {1,2} |
| 2 | (2, 4) | {1,2,4} |
| 3 | (4, 3) | {1,2,4,3} |
| 4 | (4, 7) | {1,2,4,3,7} |
| 5 | (7, 9) | {1,2,4,3,7,9} |
| 6 | (9, 10) | {1,2,4,3,7,9,5} |
| 7 | (4, 5) | {1,2,4,3,7,9,10,5} |
| 8 | (7, 8) | {1,2,4,3,7,9,10,5,8} |
| 9 | (8, 6) | {1,2,4,3,7,9,10,5,8,6} |

***Bộ dữ liệu 2:***

12

6

20

8

7

3

1

5

4

11

9

3

1

3

8

6

10

4

2

5

5

2

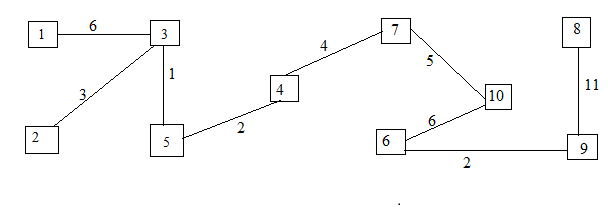
15

9

6

2

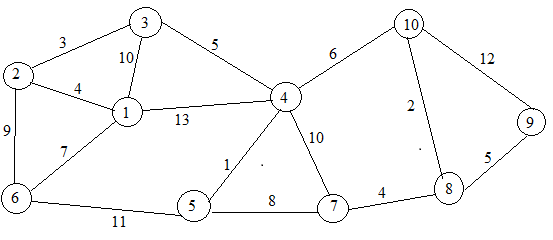
Thực hiện thuật toán Prim bắt đầu từ đỉnh 1, ta được MST:



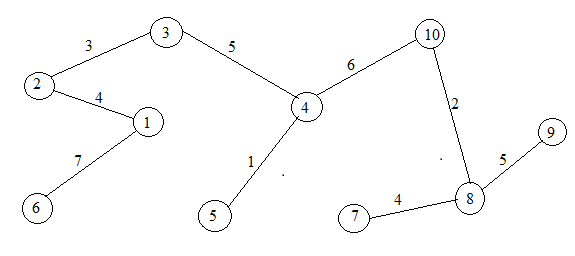
Hoạt động của thuật toán:

|  |  |  |
| --- | --- | --- |
| Bước | Các cạnh chọn | Tập U |
| 0 | - | {1} |
| 1 | (1, 3) | {1,3} |
| 2 | (3, 5) | {1,3,5} |
| 3 | (5, 4) | {1,3,5,4} |
| 4 | (3, 2) | {1,3,5,4,2} |
| 5 | (4, 7) | {1,3,5,4,2,7} |
| 6 | (7, 10) | {1,3,5,4,2,7,10} |
| 7 | (10, 6) | {1,3,5,4,2,7,10,6} |
| 8 | (6, 9) | {1,3,5,4,2,7,10,6,9} |
| 9 | (9, 8) | {1,3,5,4,2,7,10,6,9,8} |

***Bộ dữ liệu 3:***

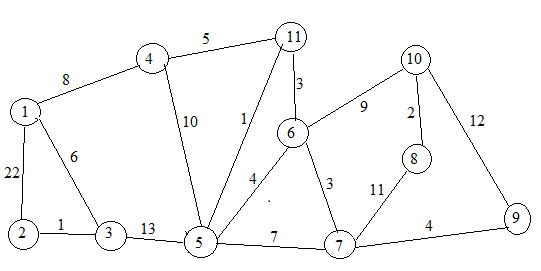
******

Thực hiện thuật toán Prim bắt đầu từ đỉnh 1 ta được MST:

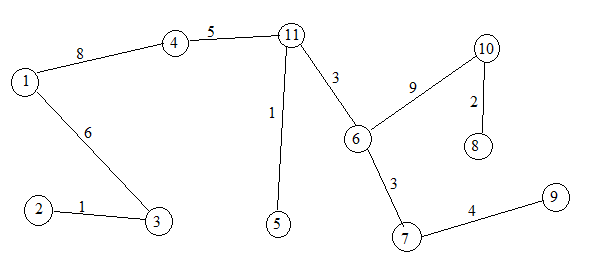


Hoạt động của thuật toán:

|  |  |  |
| --- | --- | --- |
| Bước | Các cạnh chọn | Tập U |
| 0 | - | {1} |
| 1 | (1, 2) | {1,2} |
| 2 | (2, 3) | {1,2,3} |
| 3 | (3, 4) | {1,2,3,4} |
| 4 | (4, 5) | {1,2,3,4,5} |
| 5 | (4, 10) | {1,2,3,4,5,10} |
| 6 | (10, 8) | {1,2,3,4,5,10,8} |
| 7 | (8, 7) | {1,2,3,4,5,10,8,7} |
| 8 | (8, 9) | {1,2,3,4,5,10,8,7.9} |
| 9 | (1, 6) | {1,2,3,4,5,10,8,7,9,6} |

***Bộ dữ liệu 4:*********

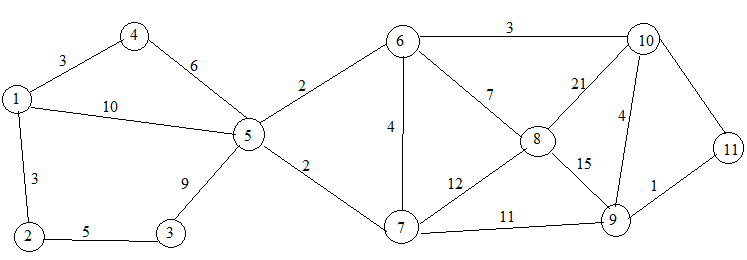
Thực hiện thuật toán Prim bắt đầu từ đỉnh 1 ta được MST:

******

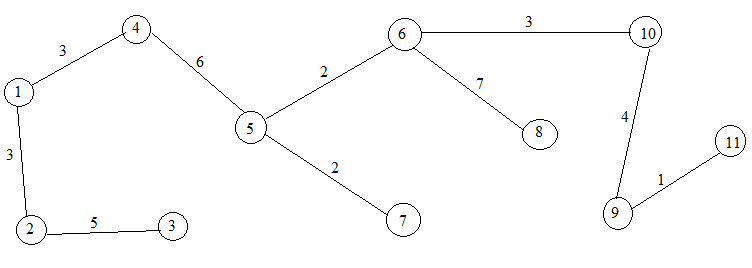
Hoạt động của thuật toán

|  |  |  |
| --- | --- | --- |
| Bước | Các cạnh chọn | Tập U |
| 0 | - | {1} |
| 1 | (1, 3) | {1,3} |
| 2 | (3, 2) | {1,3,2} |
| 3 | (1, 4) | {1,3,2,4} |
| 4 | (4,11) | {1,3,2,4,11} |
| 5 | (11, 5) | {1,3,2,4,11,5} |
| 6 | (11, 6) | {1,3,2,4,11,5,6} |
| 7 | (6, 7) | {1,3,2,4,11,5,6,7} |
| 8 | (7, 9) | {1,3,2,4,11,5,6,7.9} |
| 9 | (6, 10) | {1,3,2,4,11,5,6,7,9,10} |
| 10 | (10, 8) | {1,3,2,4,11,5,6,7,9,10,8} |

***Bộ dữ liệu 5:***

******

Thực hiện thuật toán Prim bắt đầu từ đỉnh 1 ta được MST:



Hoạt động của thuật toán

|  |  |  |
| --- | --- | --- |
| Bước | Các cạnh chọn | Tập U |
| 0 | - | {1} |
| 1 | (1, 2) | {1,2} |
| 2 | (1, 4) | {1,2,4} |
| 3 | (2, 3) | {1,2,4,3} |
| 4 | (4, 5) | {1,2,4,3,5} |
| 5 | (5, 6) | {1,2,4,3,5,6} |
| 6 | (5, 7) | {1,2,4,3,5,6,7} |
| 7 | (6, 10) | {1,2,4,3,5,6,7,10} |
| 8 | (10, 9) | {1,2,4,3,5,6,7,10,9} |
| 9 | (9, 11) | {1,2,4,3,5,6,7,10,9,11} |
| 10 | (6, 8) | {1,2,4,3,5,6,7,10,9,11,8} |

1. **Độ phức tạp của thuật toán:**

*Độ phức tạp của thuật toán theo lý thuyết:*

* Ở vòng lặp for thứ nhất có:
  + 2(n-1) phép gán.
  + (n-1) phép so sánh

= 2(n-1) + (n-1) =3(n-1)

* Ở vòng lặp for (lớn) thứ 2 có:
  + Số phép gán = 3(n-1) + +

= 3(n+1) + (n-3+1) + + 2(n-2+1) +

= + 7n -7

* + Số phép so sánh = (n-1) + +

= (n -1) + + 2(n-2+1) +

= + 4n -7

= 2 + 11n – 14

Vậy độ phức tạp của thuật toán theo lý thuyết là O(

*Độ phức tạp của thuật toán theo thực nghiệm:*