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 DIJKSTRA TÌM ĐƯỜNG ĐI NGẮN NHẤT**
2. **Bài toán**

Đồ thị G = (V, E) là đơn đồ thị liên thông (vô hướng hoặc có hướng), có trọng số. V là tập các đỉnh , E là tập các cạnh (cung).

Tìm đường đi ngắn nhất từ đỉnh s0 V đến tất cả các đỉnh còn lại.

1. **Mô tả thuật toán:**
2. Ý tưởng:

Dựa trên một dãy các bước lặp:

* + Bắt đầu từ tập chỉ chứa đỉnh xuất phát a.
  + Mỗi bước lặp thêm 1 đỉnh vào tập đỉnh đã ghé qua.
  + Gán nhãn cho các đỉnh trong mỗi bước lặp: Nhãn của đỉnh w là độ dài của đường đi ngắn nhất từ a đến nó (thông qua các đỉnh trong tập đã thăm).
  + Bước lặp tiếp: Chọn một đỉnh đã được gán nhãn (có giá trị nhãn nhỏ nhất) để tiếp tục. Nhãn của các đỉnh sẽ biến đổi trong các bước lặp, mà ở mỗi bước lặp sẽ có một nhãn tạm thời trở thành chính thức. Nếu nhãn của một đỉnh nào đó trở thành chính thức thì đó cũng chính là chiều dài ngắn nhất của đường đi từ s đến đỉnh đó.

Ký hiệu :

\* L(v) để chỉ nhãn của đỉnh v, tức là cận trên của chiều dài đường đi ngắn

nhất từ s0 đến v.

\* d(s0 ,v) : chiều dài đường đi ngắn nhất từ s0 đến v.

\* m(s0 ,v) là trọng số của cung (cạnh) (s,v).

Thuật toán Dijkstra tìm đường đi ngắn nhất từ đỉnh s đến n-1 đỉnh còn lại được mô tả như sau:

Input: G = (V, E), s0

Output : d(s0,v), mọi v ≠ s0 ;

Mô tả :

o Khởi động :

L(s0) = 0;

L(v) = ∞ , mọi v ≠ s0; //Nhãn tạm thời

S = {s0}; //Tập lưu trữ các đỉnh có nhãn chính thức

o Tại mỗi bước lặp:

- Cập nhật nhãn L(v) các đỉnh v không thuộc S:

Với L(v) = Min{L(v), L(s) + m(s,v)}

Vì chỉ có L(s\*) với s\* là đỉnh vừa duyệt xong ở bước trước là có thay đổi về giá trị nên việc tính lại L(v) chỉ có ý nghĩa với các đỉnh kề s\*:

L(v) = Min{L(v), L(s\*) + m(s\*, v)}

- Tìm đỉnh có nhãn nhỏ nhất s\*:

Đỉnh có nhãn nhỏ nhất s\* kề với một trong các đỉnh s S và

L(s\*) = Min{ L(v): v không thuộc S}

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 hoạt động của thuật toán:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Bước | L(1) | L(2) | L(3) | L(4) | L(5) | L(6) | L(7) | L(8) | L(9) | L(10) | Đỉnh đến | Đường đi từ đỉnh 1 |
| 0 | **0** |  |  |  |  |  |  |  |  |  | 1 | - |
| 1 | - | **2** | 3 |  |  |  |  |  |  |  | 2 | 1 - 2 |
| 2 | - | - | **3** | 4 |  |  |  |  |  |  | 3 | 1 - 3 |
| 3 | - | - | - | **4** |  |  |  |  |  |  | 4 | 1 - 2 - 4 |
| 4 | - | - | - | - | 8 |  | **7** |  |  |  | 7 | 1 - 2 - 4 - 7 |
| 5 | - | - | - | - | **8** |  | - | 11 | 8 |  | 5 | 1 - 2 - 4 - 5 |
| 6 | - | - | - | - | - | 16 | - | 11 | **8** |  | 9 | 1 - 2 - 4 - 7 - 9 |
| 7 | - | - | - | - | - | 16 | - | **11** | - | 11 | 8 | 1 - 2 - 4 - 7 - 8 |
| 8 | - | - | - | - | - | 13 | - | - | - | **11** | 10 | 1- 2 - 4- 7- 9- 10 |
| 9 | - | - | - | - | - | **13** | - | - | - | - | 6 | 1- 2 - 4 - 7 - 8 - 6 |

***Bộ dữ liệu 2:***

12

6

20

8

7

3

1

5

4

1

11

9

3

3

8

6

10

4

2

5

5

2

15

9

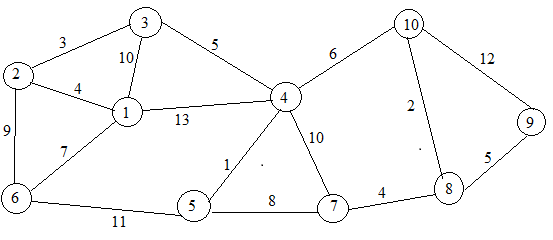
6

2

Hoạt động của thuật toán:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Bước | L(1) | L(2) | L(3) | L(4) | L(5) | L(6) | L(7) | L(8) | L(9) | L(10) | Đỉnh đến | Đường đi từ đỉnh 1 |
| 0 | **0** |  |  |  |  |  |  |  |  |  | 1 | - |
| 1 | - | 8 | **6** |  |  |  |  |  |  |  | 3 | 1 - 3 |
| 2 | - | 8 | - | 9 | **7** |  | 26 |  |  |  | 5 | 1 - 3 - 5 |
| 3 | - | **8** | **-** | 9 | - |  | 26 |  |  |  | 2 | 1 - 2 |
| 4 | - | - | - | **9** | - | 22 | 26 |  |  |  | 4 | 1 - 3 - 4 |
| 5 | - | - | - | - | **-** | 22 | **13** |  |  |  | 7 | 1 - 3 - 4 - 7 |
| 6 | - | - | - | - | - | 22 | - | 25 |  | **18** | 10 | 1 - 3 - 4 - 7 - 10 |
| 7 | - | - | - | - | - | **22** | - | 25 |  | - | 6 | 1 - 3 - 5 - 6 |
| 8 | - | - | - | - | - | - | - | 25 | **24** | **-** | 9 | 1 - 3 - 5 - 6 - 9 |
| 9 | - | - | - | - | - | **-** | - | **25** | - | - | 8 | 1 - 3 - 4 - 7 - 8 |

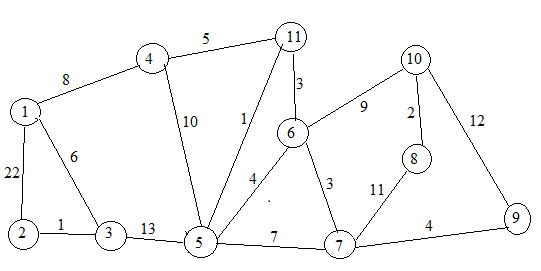
***Bộ dữ liệu 3:***

******

Hoạt động của thuật toán:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Bước | L(1) | L(2) | L(3) | L(4) | L(5) | L(6) | L(7) | L(8) | L(9) | L(10) | Đỉnh đến | Đường đi từ đỉnh 1 |
| 0 | **0** |  |  |  |  |  |  |  |  |  | 1 | - |
| 1 | - | **4** | 10 | 13 |  | 7 |  |  |  |  | 2 | 1 - 2 |
| 2 | - | - | **7** | 13 |  | 7 |  |  |  |  | 3 | 1 - 2 - 3 |
| 3 | - | - | - | 12 |  | **7** |  |  |  |  | 6 | 1 - 6 |
| 4 | - | - | - | **12** | 18 | - |  |  |  |  | 4 | 1 - 2 - 3 - 4 |
| 5 | - | - | - | - | **13** | - | 22 |  |  | 18 | 5 | 1 - 2 - 3 - 4 - 5 |
| 6 | - | - | - | - | - | - | 21 |  |  | **18** | 10 | 1 - 2 - 3 - 4 - 10 |
| 7 | - | - | - | - | - | - | 21 | **20** | 30 | - | 8 | 1 - 2 - 3- 4 -10 -8 |
| 8 | - | - | - | - | - | - | **21** | - | 25 | - | 7 | 1 - 2 - 3 - 4 - 5 -7 |
| 9 | - | - | - | - | - | - | - | - | **25** | - | 9 | 1-2- 3- 4 -10 -8-9 |

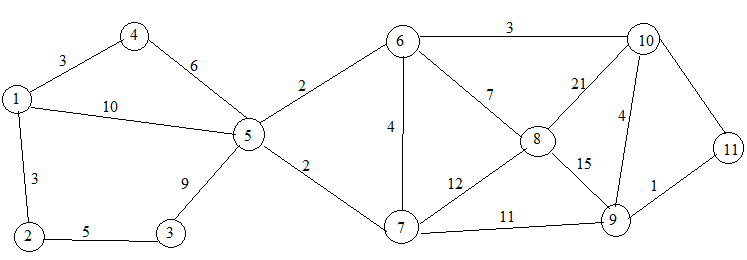
***Bộ dữ liệu 4:***

******

Hoạt động của thuật toán:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Bước | L(1) | L(2) | L(3) | L(4) | L(5) | L(6) | L(7) | L(8) | L(9) | L(10) | L(11) | Đỉnh đến | Đường đi từ đỉnh 1 |
| 0 | **0** |  |  |  |  |  |  |  |  |  |  | 1 | - |
| 1 | - | 22 | **6** | 8 |  |  |  |  |  |  |  | 3 | 1 - 3 |
| 2 | - | **7** | - | 8 | 19 |  |  |  |  |  |  | 2 | 1 – 3 - 2 |
| 3 | - | - | - | **8** | 19 |  |  |  |  |  |  | 4 | 1 - 4 |
| 4 | - | - | - | - | 18 |  |  |  |  |  | **13** | 11 | 1 - 4 - 11 |
| 5 | - | - | - | - | **14** | 16 |  |  |  |  | - | 5 | 1- 4-11-5 |
| 6 | - | - | - | - | - | **16** | 21 |  |  |  | - | 6 | 1- 4-11-6 |
| 7 | - | - | - | - | - | - | **19** |  |  | 25 | - | 7 | 1-4-11-6-7 |
| 8 | - | - | - | - | - | - | - | 30 | **23** | 25 | - | 9 | 1-4-11-6-7-9 |
| 9 | - | - | - | - | - | - | - | 30 | - | **25** | - | 10 | 1-4-11-6-10 |
| 10 | - | - | - | - | - | - | - | **27** | - | - | - | 8 | 1-4-11-6-10-8 |

***Bộ dữ liệu 5:***

******

5

Hoạt động của thuật toán:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Bước | L(1) | L(2) | L(3) | L(4) | L(5) | L(6) | L(7) | L(8) | L(9) | L(10) | L(11) | Đỉnh đến | Đường đi từ đỉnh 1 |
| 0 | **0** |  |  |  |  |  |  |  |  |  |  | 1 | - |
| 1 | - | **3** |  | 3 | 10 |  |  |  |  |  |  | 2 | 1 - 2 |
| 2 | - | - | 8 | **3** | 10 |  |  |  |  |  |  | 4 | 1 - 4 |
| 3 | - | - | **8** | **-** | 9 |  |  |  |  |  |  | 3 | 1 – 2 - 3 |
| 4 | - | - | - | - | **9** |  |  |  |  |  |  | 5 | 1 – 4 - 5 |
| 5 | - | - | - | - | - | **11** | 11 |  |  |  |  | 6 | 1 -4 -5 -6 |
| 6 | - | - | - | - | - | - | **11** | 18 |  | 14 |  | 7 | 1 -4 -5 -7 |
| 7 | - | - | - | - | - | - | - | 18 | 22 | **14** |  | 10 | 1-4-5-6-10 |
| 8 | - | - | - | - | - | - | - | **18** | 18 | - | 19 | 8 | 1-4-5-6-8 |
| 9 | - | - | - | - | - | - | - | - | **18** | - | 19 | 9 | 1-4-5-6-10-9 |
| 10 | - | - | - | - | - | - | - | - | - | - | **19** | 11 | 1-4-5-6-10-11 |

1. **Cài đặt thuật toán**

void Dijkstra() {

int i,k, minp, dht;

int h = 1;

for ( i = 1; i <= n; i++) {

Daxet[i] = 0;

L[i] = INT\_MAX;

}

Daxet[s] = 1;

L[s] = 0;

dht = s;

while (h < n) {

minp = INT\_MAX; // Thay MAX b?ng INT\_MAX

for (i = 1; i <= n; i++) {

if (!Daxet[i]) {

if ((L[dht] + a[dht][i]) < L[i]) {

L[i] = L[dht] + a[dht][i];

Truoc[i] = dht;

}

if (L[i] < minp) {

minp = L[i];

k = i;

}

}

}

Print(s, k, Truoc);

cout << "\n Trong so: " << L[k];

dht = k;

Daxet[dht] = 1;

h++;

}

}

1. **Độ phức tạp của thuật toán:**

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

độ phức tạp thời gian của Thuật toán Dijkstra dưới dạng ký hiệu big-O sử dụng danh sách kề.

Mỗi đỉnh có thể được kết nối với (V-1) đỉnh, do đó số cạnh liền kề với mỗi đỉnh là V - 1. Giả sử E đại diện cho các cạnh V-1 được nối với mỗi đỉnh.

Tìm và cập nhật trọng lượng của mỗi đỉnh liền kề trong heap tối thiểu là O(log(V)) + O(1) hoặc O(log(V)).

Do đó từ bước 1 và bước 2 ở trên, độ phức tạp thời gian để cập nhật tất cả các đỉnh liền kề của một đỉnh là E \* (logV). hoặc E\*logV.

* Do đó độ phức tạp thời gian cho tất cả các đỉnh V là V \* (E \* logV) i.e O(VElogV).