Thuật toán Kruskal

Bởi:

Khoa CNTT ĐHSP KT Hưng Yên

Thuật toán Kruskal

Thuật toán sẽ xây dựng tập cạnh T của cây khung nhỏ nhất H=(V,T) theo từng bước. Trước hết sắp xếp các cạnh của đồ thị G theo thứ tự không giảm của độ dài. Bắt đầu từ tập T=∅ , ở mỗi bước ta sẽ lần lượt duyệt trong danh sách cạnh đã sắp xếp, từ cạnh có độ dài nhỏ đến cạnh có độ dài lớn hơn, để tìm ra cạnh mà việc bổ sung nó vào tập T gồm n-1 cạnh. Cụ thể, thuật toán có thể mô tả như sau:

Procedure Kruskal;

Begin

T:= ∅ ;

While |T| < (n-1) and (E<> ∅) do

Begin

E:=E \ {e};

if (T ? {e} không chứa chu trình) then T:= T ? {e} ;

End;

if (|T| < n-1) then Đồ thị không liên thông;

End;

Ví dụ 5.1

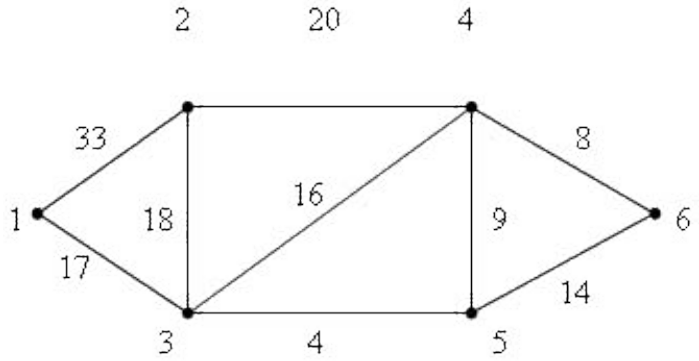
Tìm cây khung nhỏ nhất của đồ thị cho trong hình 5.1 dưới.

Bước khởi tạo. Đặt T:=∅ . Sắp xếp các cạnh của đồ thị theo thứ tự không giảm của độ dài ta có dãy:

(3,5) , (4,6) , (4,5) , (5,6) , (3,4) , (1,3) , (2,3) , (2,4) , (1,2)

dãy độ dài tương ứng của chúng

4, 8, 9, 14, 16, 17, 18, 20, 23.



Hình 5.1 Đồ thị và cây khung nhỏ nhất.

Ở ba lần gặp đầu tiên ta lần lượt bổ sung vào tập T các cạnh (3,5) , (4,6) , (4,5). Rõ ràng nếu thêm cạnh (5,6) vào T thì sẽ tạo thành 2 cạnh (4,5), (4,6) đã có trong T chu trình. Tình huống tương tự cũng xảy ra đối với cạnh (3,4) là cạnh tiếp theo của dãy. Tiếp theo ta bổ sung cạnh (1,3), (2,3) vào T và thu được tập T gồm 5 cạnh:

T = { (3,5) , (4,6) , (4,5) , (1,3) , (2,3) }

Chính là tập cạnh của cây khung nhỏ nhất cần tìm.

Chứng minh tính đúng đắn của thuật toán.

Rõ ràng đồ thị thu được theo thuật toán có n-1 cạnh và không có chu trình, vì vậy theo định lý 4.1 nó là cây khung của đồ thị G. Như vậy, chỉ còn phải chỉ ra rằng T có độ dài nhỏ nhất. Giả sử tồn tại cây S của đồ thị G mà c(S) < c(T). Ký hiệu e_k là cạnh đầu tiên trong dãy các cạnh của T xây dựng theo thuật toán vừa mô tả không thuộc S. khi đó đồ thị con của G sinh bởi cây S được bổ sung cạnh e_k sẽ chứa một chu trình C duy nhất đi qua e_k. Do chu trình C phải chứa cạnh e thuộc S nhưng không thuộc T nên đồ thị con thu được từ S bằng cách thay cạnh e của nó bởi cạnh e_k (ký hiệu đồ thị là S') sẽ là cây khung. Theo cách xây dựng $c(e_k) \leq c(e)$ do đó $c(S') \leq c(S)$, đồng thời số cạnh chung của S' và T đã tăng thêm 1 so với số cạnh chung của S và T. Lặp lại quá trình trên từng bước một ta có thể biến đổi S thành T và trong mỗi bước tổng độ dài không tăng, tức là $c(T) \leq c(S)$. Mâu thuẫn thu được chúng tỏ T là cây khung nhỏ nhất.

Về việc lập trình thực hiện thuật toán.

Khối lượng tính toán nhiều nhất của thuật toán chính là ở bước sắp xếp các cạnh của đồ thị theo thứ tự không giảm của độ dài để lựa chọn cạnh bổ sung. Đối với đồ thị m cạnh cần phải thực hiện mlogm phép toán để sắp xếp các cạnh của đồ thị thành dãy không giảm theo độ dài. Tuy nhiên, để xây dựng cây khung nhỏ nhất với n-1 cạnh, nói chung ta không cần phải sắp thứ tự toàn bộ các cạnh mà chỉ cần xét phần trên của dãy đó chứa r < m cạnh. Để làm việc đó ta có thể sử dụng các thủ tục sắp xếp dạng Vun đống (Heap Sort). Trong thủ tục này, để tạo đống đầu tiên ta mất cỡ O(m) phép toán, mỗi phần tử tiếp theo trong đống có thể tìm sau thời gian O(log m). Vì vậy, với cải tiến này thuật toán sẽ mất thời gian cỡ O(m+p) log m) cho việc sắp xếp các cạnh. Trong thực tế tính toán số p nhỏ hơn rất nhiều so với m.

Vấn đề thứ hai trong việc thể hiện thuật toán Kruskal là việc lựa chọn cạnh để bổ sung đòi hỏi phải có một thủ tục hiệu quả kiểm tra tập cạnh T ? {e} có chứa chu trình hay không. Để ý rằng, các cạnh trong T ở các bước lặp trung gian sẽ tạo thành một rừng. Cạnh e cần khảo sát sẽ tạo thành chu trình với các cạnh trong T khi và chỉ khi cả hai đỉnh đầu của nó thuộc vào cùng một cây con của rừng nói trên. Do đó, nếu cạnh e không tạo thành chu trình với các cạnh trong T, thì nó phải nối hai cây khác nhau trong T. vì thế, để kiểm tra xem có thể bổ sung cạnh e vào T ta chỉ cần kiểm tra xem nó có nối hai cây khác nhau trong T hay không. Một trong các phương pháp hiệu quả để thực hiện việc kiểm tra này là ta sẽ phân hoạch tập các đỉnh của đồ thị ra thành các tập con không giao nhau, mỗi tập xác định bởi một cây con trong T(được hình thành ở các bước do việc bổ sung cạnh vào T). chẳng hạn, đối với đồ thị trong ví dụ 3, đầu tiên ta có sáu tập con 1 phần tử: {1}, {2}, {3}, {4}, {5}, {6} . Sau khi bổ sung cạnh (3, 5), ta có các tập con {1}, {2}, {3,5}, {4}, {6} . Ở bước thứ 3, ta chọn cạnh (4, 5), khi đó hai tập con được nối lại và danh sách các tập con là {1}, {2} , {3, 5, 4, 6} . Cạnh có độ dài tiếp theo là (4,6), do hai đầu của nó thuộc vào cùng một tập con {3,4,5,6} , nên nó sẽ tạo thành chu trình trong tập này. Vì vậy cạnh này không được chọn. Và thuật toán sẽ tiếp tục chọn cạnh tiếp theo để khảo sát …

Như vậy, để giải quyết vấn đề thứ hai này ta phải xây dựng hai thủ tục: Kiểm tra xem hai đầu u, v của cạnh e=(u,v) có thuộc vào hai tập con khác nhau hay không, và trong trường hợp câu trả lời là khẳng định, nối hai tập con tương ứng thành một tập. Chú ý rằng mỗi tập con trong phân hoạch có thể lưu trữ như là một cây có gốc, và khi đó mỗi gốc sẽ được sử dụng làm nhãn nhận biết tập con tương ứng.

Chương trình trên Pascal thực hiện thuật toán Kruskal với những nhận xét vừa nêu có thể viết như sau:

(* TÌM CÂY KHUNG NHỎ NHẤT THEO THUẬT TOÁN KRUSKAL CỦA ĐỒ THỊ CHO BỞI DANH SÁCH CẠNH *)

```
uses crt;
type
arrn=array[1. .50] of integer;
arrm= array[1...50] of integer;
var
n,m, minL:integer;
Dau, cuoi, W:arrm;
DauT, CuoiT, Father:arrn;
Connect:boolean;
Procedure Nhapdl;
Var
i:integer;
Fanme:string;
F:text;
Begin
Write('Cho ten file du lieu:') readln(fname);
Assign(f,fname); reset(f);
Readln(f,n,m);
For i:=1 to m do readln(f, Dau[i], Cuoi[i], W[i]);
Close(f);
End;
Procedure Indulieu;
```

```
Var i:integer;
Begin
Writeln('So dinh ',n,'. So canh ',m);
Writeln('Dinh dau Dinh cuoi Do dai');
For i:=1 to m do
Writeln(Dau[i]:4, Cuoi[i}:10, W[i]:12);
End;
Procedure Heap(First, Last:integer);
Var j,k,t1,t2,t3 : integer;
Begin
J:=first;
While (j<=trunc(last/2)) do
Begin
If (2*j<last) and W[2*j+1]<W[2*j]) then K:= 2*j+1
Else k"=2*j;
If W[k]<W[j} then
Begin
T1:=Dau[j]; t1:=Cuoi[j]; t3:=W[j];
Dau[j]:=Dau[k];Cuoi[j]:=Cuoi[k]; W[j]:=W[k];
Dau[k]:=t1; Cuoi[k]:=t2; W[k]:=t3;
J:=k;
End
```

```
Else j:=Last;
End;
End;
Function Find(i:integer):integer;
Var Tro:integer;
Begin
Tro:=i;
While Father[Tro]>0 do Tro:=Father[Tro];
Find:=Tro;
End;
Procedure Union(i,j:integer);
Var x:integer;
Begin
x:=father[i]+father[j];
if father[i]>father[j] then
begin
father[i]:=f;
father[j]:=x;
end
else
begin
father[j]:=i;
```

```
father[i]:=x;
end;
End;
Procedure Kruskal;
Var
I, Last, u,v,r1,r2, Ncanh, Ndinh:integer;
Begin
(* Khoi tao mang Father danh dau cay con va khoi tao Heap *)
for i:= 1 to n do father[i]:=-1;
for i:=trunc(m/2) downto 1 do Heap(i,m);
last:=m; Ncanh:=0; Ndinh:=0;
MinL:=0;
Connect:=true;
While (Ndinh<n-1) and (Ncanh<m) do
Begin
Ncanh:=Ncanh+1;
u:=dau[1];
v:=Cuoi[1];
(* Kiem tra u va v co thuoc cung mot cay con *)
r1:=find(u);
r2:=find(v);
if r1<>r2 then
```

```
begin
(* Ket nap canh (u,v) vao cay khung *)
Ndinh:=Ndinh+1; Union(r1, r2);
DauT[Ndinh]:=u;
CuoiT[Ndinh]:=v;
MinL:=MinL+W[1];
end;
(* To chuc lai Heap *)
Dau[1]:=Dau[Last];
Cuoi[1]:=Cuoi[Last];
W[1]:=W[Last];
Last:=Last-1;
End;
If Ndinh<>n-1 then Connect:=false;
End;
Procedure Inketqua;
Var i:integer;
Begin
Writeln('*******************************');
Writeln('***** Ket qua tinh toan ********');
Writeln('*******************************');
Writeln('Do dai cua cay khung nho nhat: ',MinL);
```

```
Writeln('Cac canh cua cay khung nho nhat');
For i:=1 to n-1 do
        Writeln('(',DauT[i]:2,',',CuoiT[i]:2,')');
Writeln('*******************************');
End;
Begin
Clrscr;
Nhapdl;'
Indulieu;
Kruskal;
If connect then Inketqua
Else
Writeln(' Do thi khong lien thong');
Readln;
End.
```

File dữ liệu của bài toán trong ví dụ 3 có dạng sau:

```
7 9
3 5 4
4 6 8
4 5 9
5 6 14
3 4 16
```

1 3 17

2 3 18

2 4 20

1 2 23