Union-Find算法应用

最小代价生成树的算法。设 S＝ （V，T）是无向图 G ＝（V，T）的无向树。采用 KRUSKAL 算法，求出该最小代价生成树。

begin

1、 T <— φ

2、 V_S <— φ

3、 **for** each vertex v ∈ V **do** add the singleton set {v } to V_S

4、 **while** $\|V_S\|$> 1 **do**

begin

5、 choose (v,w), an edge in E of lowest cost; // MIN

6、 delete (v,w) from E; // DELETE

7、 **if** v and w are in different sets W_1 and W_2 in V_S then // FIND

begin

8、 **replace** W_1 and W_2 in V_S by $W_1 \cup W_2$ // UNION

9、 add (v, w) to T // INSERT

end

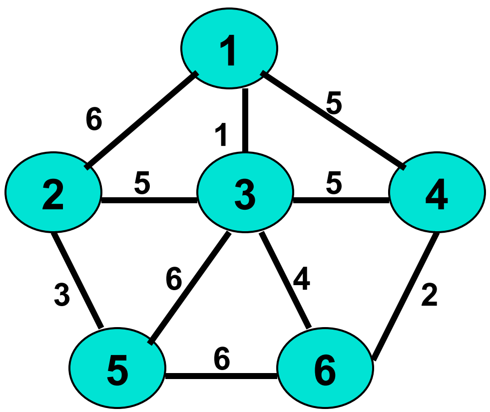
end

end

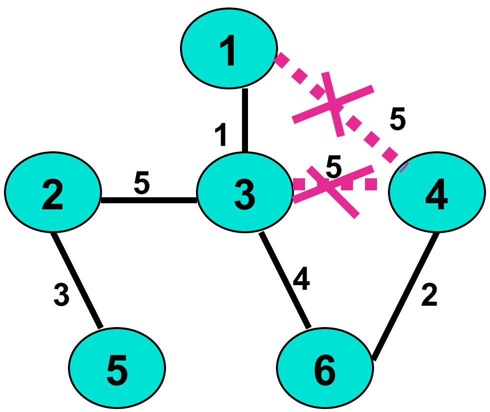
- 实例的执行过程

  最小代价生成树: **S = (V, T)**

G = (V, E)



S = (V, T)



VS = { {1},{2},{3},{4},{5},{6} }

VS = { {1, 3}, {2}, {4}, {5}, {6} }

VS = { {1, 3}, {2}, {5},{4, 6} }

VS = { {1, 3}, {2, 5},{4, 6} }

VS = { {1, 3, 4, 6} , {2, 5} }

VS = { {1, 3, 4, 6, 2, 5} }

Find

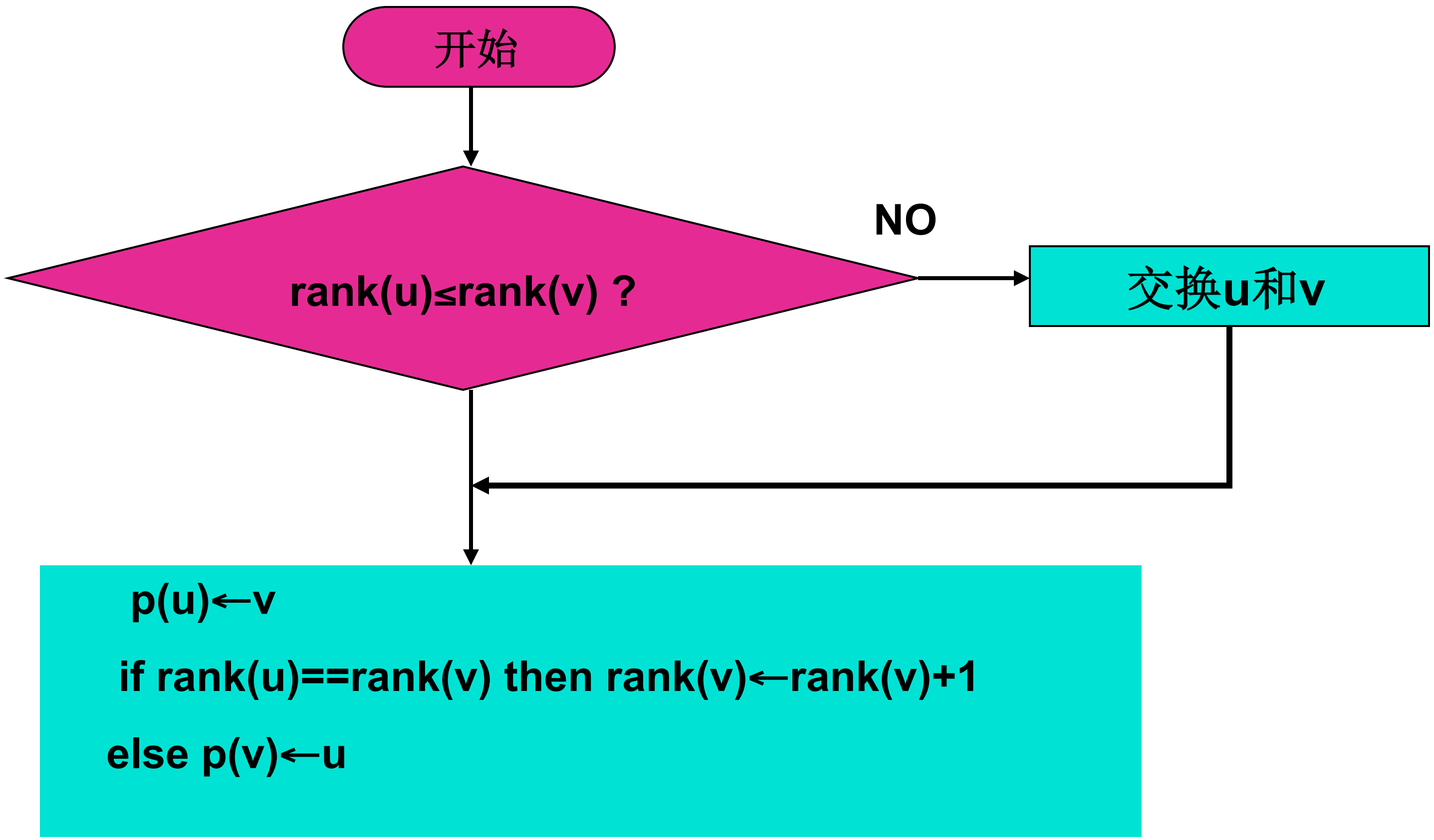
Algorithm 4.6 FIND

Input: A node x

Output: root(x) the root of the tree containing x

1. y←x
2. while p(y)≠null {Find the root of the tree containing x}
3. y←p(y)
4. end while
5. root←y; y←x
6. while p(y)≠null {Do path compression}
7. w←p(y)
8. p(y)←root
9. y←w
10. end while
11. return root

UNION(u,v): 把秩小的树合并到秩大的树上。



Union

Algorithm 4.7 UNION(x,y)

Input: Two elements x and y

Output: The union of the two trees containing x and y. The original trees are destroyed.

1. u←FIND(x); v←FIND(y)
2. if rank(u)≤rank(v) then
3. p(u)←v
4. if rank(u)=rank(v) then rank(v)←rank(v)+1
5. else p(v)←u
6. end if

UNION-FIND问题的树结构及其应用

3、快速的不相交集的合并算法的时间耗费：

- 函数 F 和 G：

F 函数的定义：

$$F(i) = \begin{cases} 1 & i=0 \\ 2^{F(i-1)} & \text{for } i > 0 \end{cases}$$

该函数增长非常快；e.g: F(0)=1; F(1)=2; F(2)=4; F(3)=16;F(4)=2^{16}……

G 函数定义为：G(n)=k; 使得 F(k) >= n 的最小的整数 k

该函数增长非常慢；e.g: G(1)=0; G(2)=1; G(4)=2; G(16)=4; G(2^{16})= 4 ……

- 执行 UNION 的代价是常数，执行 FIND(i) 代价正比于从 i上溯至根时遇到的结点个数。

K	F(K)
0	1
1	2
2	4
3	16
4	65536

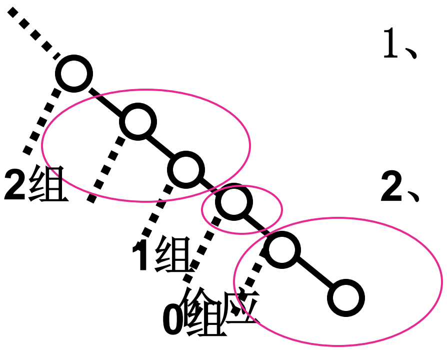
G(r)	r
0	0,1
1	2
2	3,4
3	5~16
4	17 ~ 65536

UNION-FIND问题的树结构及其应用

3、快速的不相交集的合并算法的时间耗费：

- 具有 cn 条 UNION 和 FIND 指令构成的序列σ时间耗费：

3、序列σ中的所有 FIND 指令的时间代价：



1、因为最多存在 G(n) 个不同的秩组，所以对每一条 FIND 指令而言，指派的时间代价最多为 G(n)。所以对 cn 条FIND 指令的代价至多为 cn G(n)。

2、设结点 v 属于秩组 g。则在该秩组，移动的次数不会超过F(g)-F(g-1) 次。所以，指派的代价不会超过 F(g)-F(g-1) 。因此，秩组 g 的总的时间代价为：

g=2 F(2)=4 F(1)=2
F(2) - F(1) = 1

该总代价 ＜＝ N(G) × (F(g)-F(g-1))；N(g) 为秩组 g 的结点总数。

但 $N(g) <= \sum_{r=F(g-1)+1}^{F(g)} n/2^r <= \sum_{r=F(g-1)+1}^{\infty} n/2^r = \frac{n}{2^{F(g-1)+1}} (1+1/2+1/4\ldots\ldots) <= n/F(g)$

∴ 该总代价 <= N(G) × (F(g)-F(g-1)) <= N(G) × F(g) = n

由于最多有 G(n) 个秩组，所以从结点角度进行考虑，执行 cn 条 FIND 指令的代价为 O(nG(n))。

1 ＋ 2 的时间代价为 O(nG(n)) - 序列σ中的所有 FIND 指令的时间代价

- 具有 cn 条 UNION 和 FIND 指令构成的序列σ时间耗费：

1 + 3 = O(n) + O(n G(n)) = O(n G(n)) 近似于线性。