并查集的启发式策略

主讲人：邓哲也

大纲

按秩合并

路径压缩

代码实现

并查集的启发式策略

通过两种启发式策略，我们可以得到一个几乎与总的操作数 m 成线性关系的运行时间。

第一种启发式是按秩合并。

目的是使包含较少节点的树的根指向包含较多节点的树的根。

第二种启发式是路径压缩。

目的是减少Find-Set操作中访问的节点个数。

按秩合并

目的是使包含较少节点的树的根指向包含较多节点的树的根。

我们并不记录以每个节点为根的子树的大小。

采用一种简化分析的方法。

对每个节点，用秩（rank）表示节点高度的一个上界。

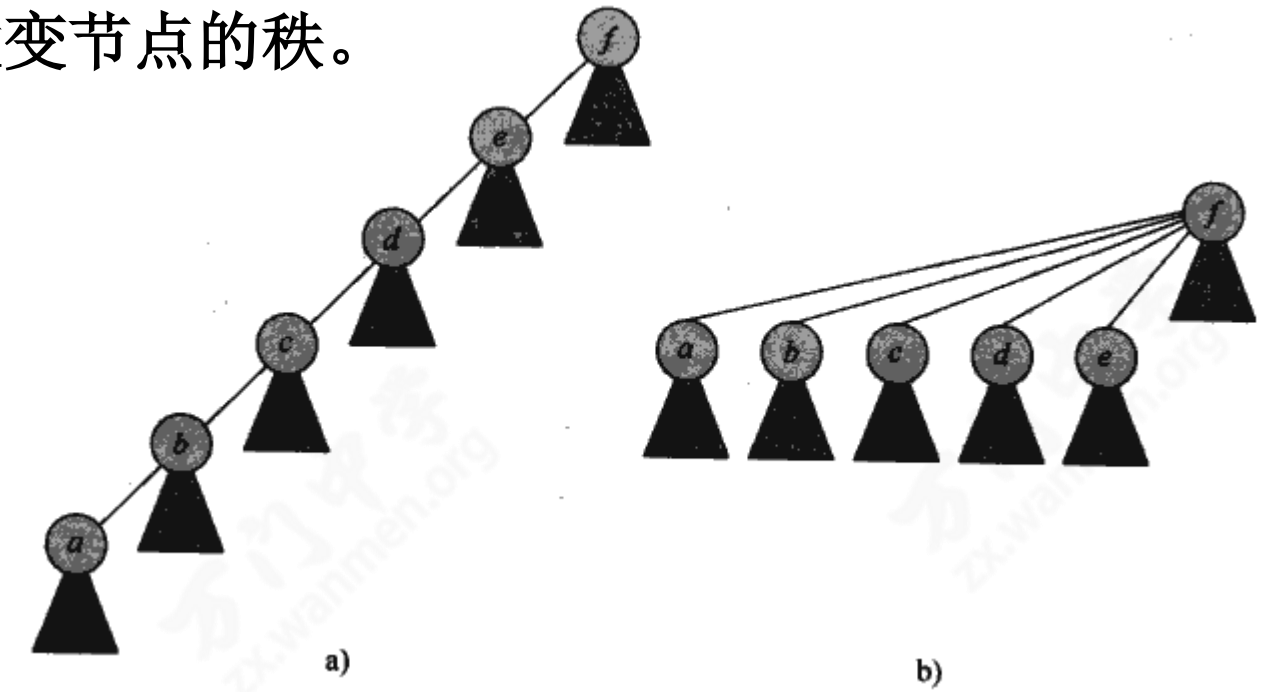
在按秩合并中，具有较小秩的根在 Union 操作中要指向具有较大秩的根。

路径压缩

目的是减少Find-Set操作中访问的节点个数。

在 Find 的过程中，使查找路径上的每个节点都指向根节点。

路径压缩不改变节点的秩。



代码实现

p[x] 表示 x 的父节点，rank[x] 表示 x 的秩。

```
Make-Set(x)
    p[x] = x
    rank[x] = 0
```

Link 过程是由 Union 调用的一个子过程。

代码实现

p[x] 表示 x 的父节点，rank[x] 表示 x 的秩。

Link 过程是由 Union 调用的一个子过程。

```
Union(x, y)
   Link(Find-Set(x), Find-Set(y))

Link(x, y)
   if rank[x] > rank[y]
      p[y] = x
   else
      p[x] = y
      if rank[x] == rank[y]
         rank[y] = rank[y] + 1
```

代码实现

p[x] 表示 x 的父节点，rank[x] 表示 x 的秩。

```
Find-Set(x)
    if x != p[x]
        p[x] = Find-Set(p[x])
    return p[x]
```

第一趟沿查找路径上升，直至找到根

第二趟沿查找路径下降，以便更新每个节点，使之直接指向根。

时间复杂度

当同时使用按秩合并和路径压缩时，最坏情况运行时间为$O(m\alpha(n))$，其中 $\alpha(n)$ 是一个增长极其缓慢的函数。

在各种实际情况中，可以把这个运行时间看作与 m 成线性关系。

下节课再见