Set $wgLogo to the URL path to your own logo image.

为防止广告，目前nocow只有登录用户能够创建新页面。如要创建页面请先登录/注册（新用户需要等待1个小时才能正常使用该功能）。

并查集

什么是并查集？ [编辑]

并查集是一种树型的数据结构，用于处理一些不相交集合（Disjoint Sets）的合并及查询问题。常常在使用中以森林来表示。 进行快速规整。

并查集的主要操作 [编辑]

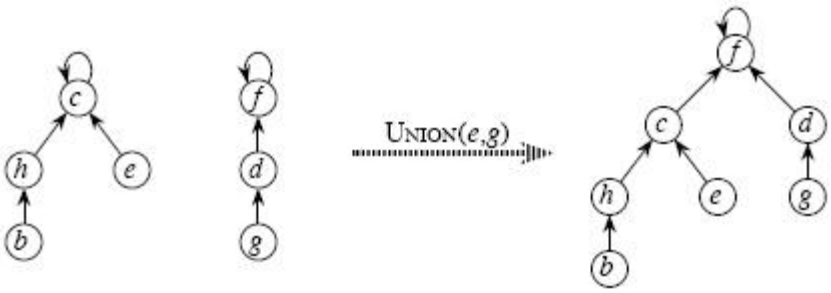
1. 合并两个不相交集合
2. 判断两个元素是否属于同一集合

主要操作的解释及代码 [编辑]

需要注意的是，一开始我们假设元素都是分别属于一个独立的集合里的。

（1） 合并两个不相交集合 操作很简单：先设置一个数组Father[x]，表示x的“父亲”的编号。 那么，合并两个不相交集合的方法就是，找到其中一个集合最父亲的父亲（也就是最久远的祖先），将另外一个集合的最久远的祖先的父亲指向它。

附图一张（摘自CLRS）——Ronice



a图为两个不相交集合，b图为合并后Father(b):=Father(g)

代码(pascal)：

```
Procedure Union(x,y:integer);{其中GetFather是下面将讲到的操作}
 var fx,fy : integer;
  begin
     fx := GetFather(x);
     fy := GetFather(y);
     Father[fx] := fy;{指向最祖先的祖先}
  end;
```

C：

```
inline void union(int x, int y)
{
    // get_father 是下面将讲到的操作
    father[get_father(x)] = get_father(y); //指向最祖先的祖先
}
```

（2） 判断两个元素是否属于同一集合 仍然使用上面的数组。则本操作即可转换为寻找两个元素的最久远祖先是否相同。可以采用递归实现。 （有待补图，制作中） 代码(pascal)：

```
Function Same(x,y:integer):boolean;
begin
  Same:=(GetFather(x)=GetFather(y));
end;
```

C:

```
inline char is_same(int x, int y)
{
    return get_father(x) == get_father(y);
}
```

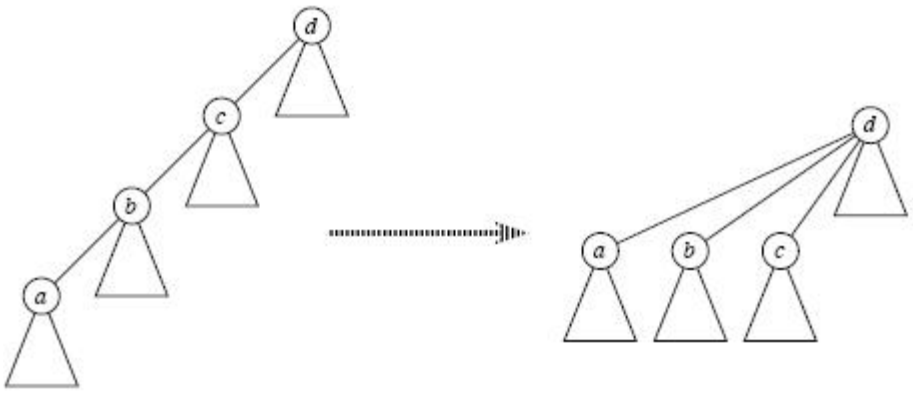
并查集的优化 [编辑]

（1）路径压缩

刚才我们说过，寻找祖先时采用递归，但是一旦元素一多起来，或退化成一条链，每次GetFather都将会使用O（n）的复杂度，这显然不是我们想要的。

对此，我们必须要进行路径压缩，即我们找到最久远的祖先时“顺便”把它的子孙直接连接到它上面。这就是路径压缩了。使用路径压缩的代码如下，时间复杂度基本可以认为是常数的。

附图摘自CLRS：



pascal:

```
Procedure Initialize;
var
  i:integer;
begin
  for i:=1 to maxv do
    Father[i]:=i;
end;
```

```
Function GetFather(v:integer):integer;
begin
  if Father[v]<>v then
    Father[v]:=GetFather(Father[v]);
  exit(Father[v]);
```

```
end;
```

C:

```
void init(void)
{
    int i;
    for (i=0; i<MAX; i++)
        father[i] = i;
}
```

```
int get_father(int v)
{
    if (father[v] != v)
        father[v] = get_father(father[v]);
    return father[v];
}
```

（2）rank合并

合并时将元素少的集合合并到元素多的集合中。 pascal:

```
{ 初始化: fillchar(rank,sizeof(rank),0); }
function judge(x,y:integer):boolean;
 var fx,fy : integer;
  begin
     fx := GetFather(x);
     fy := GetFather(y);
     If fx=fy then exit(true)
              else judge := false;
     if rank[fx]>rank[fy] then father[fy] := fx
                          else
                            begin
                              father[fx] := fy;
                              if rank[fx]=rank[fy] then inc(rank[fy]);
                            end;
  end;
```

C:

```
// 初始化
static int rank[MAX] = { 0 };

char judge(int x, int y)
{
    int fx, fy;
    if ((fx = get_father(x)) == (fy = get_father(y)))
        return 1;

    if (rank[fx] > rank[fy])
        father[fy] = fx;
    else {
        father[fx] = fy;
        if (rank[fx] == rank[fy])
            rank[fy]++;
    }
    return 0;
}
```

时间复杂度 [编辑]

O(n*α(n))

其中α(x),对于x=宇宙中原子数之和,α(x)不大于4

事实上，路经压缩后的并查集的复杂度是一个很小的常数。

源代码 [编辑]

加了所有优化的代码框架：

pascal

c

c++(不含秩的优化)

习题 [编辑]

Noi2002 银河英雄传说

CEOI'99 Parity

Kruskal算法的优化

GNU FDL FREE DOC LICENSE

此页面已被浏览过26,065次。 本页面由NOCOW匿名用户58.49.51.35于2011年12月18日 (星期日) 21:08做出最后修改。 在兰威举和允诺永在、NOCOW用户Onetwogoo、NOCOW匿名用户121.17.46.140和其他的工作基础上。 本站全部文字内容使用GNU Free Documentation License 1.2授权。 隐私权政策 关于NOCOW 免责声明

陕ICP备09005692号

Powered By MediaWiki