[此博文通过手机撰写(手机访问sina.cn)](http://news.sina.com.cn/437/2008/0703/24.html)

## 最大流增广路算法的扩展,多路增广

 (2006-10-06 22:22:27)

[IMG_257转载▼](http://blog.sina.com.cn/s/javascript:;)

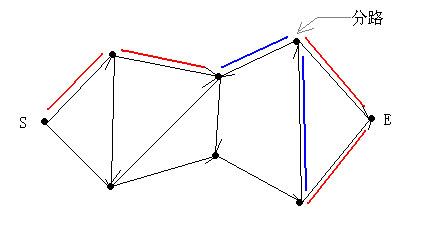
|  |  |
| --- | --- |
|  | 分类： [程序设计](http://blog.sina.com.cn/s/articlelist_1258114940_1_1.html" \t "http://blog.sina.com.cn/s/_blank) |

一般的,在执行增广路算法时,都是先用BFS或DFS从源到汇找到一条增广路,记录下应修改的流量,然后再顺着路倒回去增广.反复这个过程直到增广路找不到了为止.

显然的,我们做了很多无用功,假设有两条很长的增广路,前面大部分都是重叠的,只是在最后关头分了个岔,而程序却把前面很长的路走了两次.

为什么要这样?

不妨把两条增广路合并起来,不止是两条,所有的增广路都可以按其前缀合并起来,而形成一棵增广树.找增广树可以用DFS,正如生成搜索树一样.



简而言之,对于当前的每一个结点记录一个可提供的最大流量,源点的可供流量显然是无穷大,当推到下一个点时,最大流量取边的容量和上个点提供的最大流量的较小值.

当到达汇点时,自然DFS开始回朔,这时按当前点的已用流量增广当前点与他父亲相连的边,同时将增广值累加到他父亲的已用流量上,并在他父亲的可提供流量上减掉这个值,以便在搜索他父亲剩下的儿子时,让所有儿子的已用流量总和不大于父亲的可提供流量,不然就出错了.

我们按照以下步骤做:

search node(available)

{

  1.得到node的可提供流量available

  2.search node's all son(可提供流量)

    {

       inc(已用流量,当前儿子实际增广的流量)

       dec(可提供流量,<同上>)

    }

  3.用得到的已用流量的值增广边(node-node's father)

  4.return 边的增广量 到 node'fahter

}

每进行一次以上步骤,我们就完成了一次多路增广,并且返回了一个值到源点,即DFS的根.这个值表示本次操作中将流量扩大了多少.重复操作直到返回值为0.得到最大流的数值即累加每次的返回值.

理论复杂度依旧,实际效果非常好.