جاده‌کشی در لی‌لی‌پوت

پویا آقاحسینی ۹۵۱۳۰۰۶

ابتدا میخواهیم گزاره اول را بررسی کنیم:
فرض میکنیم مسیر جهت دار وجود دارد و فرض خلف میکنیم که جاده بد یا همان یال برشی در گراف موجود است. در صورتی که یال برشی موجود باشد دو سر آن یال رئوس u, v را در نظر میگیریم که با حذف یال این دو راس(شهر) از هم غیر قابل دسترس میشوند، بین این دو راس تنها یک مسیر وجود دارد که از یال برشی میگذرد، حال اگر این یال برشی را جهت دهی کنیم ، تنها از یکی به دیگری میتوان رفت و بین این دو راس تنها یک مسیر از یکی به دیگری وجود دارد که با فرض در تناقض است.
حال برای برگشت آن فرض میکنیم جاده بد یا یال برشی ای نداریم، باید با کمک استقرا ثابت کنیم که میتوان گرافی ساخت که از هر راسی به بقیه مسیر باشد:
ابتدا برای پایه استقرا که یک راس باشد حکم برقرار است و از یک راس به خودش میتوان رفت. فرض میکنیم حکم برای $k-1$ راس برقرار است و برای هر راس دلخواهی به راس دیگری مسیر وجود دارد، حال راس جدید V_k را اضافه میکنیم، برای اینکه از V_k به هر راسی مسیر باشد یک یال و برعکس از هر راسی به V_k مسیر وجود داشته باشد یک یال دیگر احتیاج داریم که دو یال از V_k به گرافی که در فرض داشتیم متصل میکنیم.
افزایش این دو یال به این صورت است که از V_k به راس دلخواه u در گراف یالی جهت دار وصل میکنیم و چون طبق از u به همه رئوس مسیر داشتیم حال از V_k نیز از طریق u مسیر داریم. سپس از راس دلخواه u' در گراف یالی جهت دار به V_k وصل میکنیم و چون طبق فرض از هر راسی به u' مسیر وجود داشت و حال از u' نیز به V_k مسیر داریم، از هر راسی از طریق u' به V_k مسیر داریم. چون گراف ساده است، $u \neq u'$ است.

گزاره اول اثبات شد، حال الگوریتم به این صورت است که در انتها یک SCC جهت‌دار از گراف باقی می‌ماند و همانطور که برای یافتن SCC از DFS استفاده می‌کردیم، در این الگوریتم نیز از DFS استفاده میکنیم با این تفاوت که در هر مرحله چک میکنیم که حتما Back Edge داشته باشیم چون در غیر این صورت یال برشی داریم چون پس از جهت‌دهی دیگر از یکی به دیگری مسیر نداریم. اثبات به این گونه است که یال (v, w) در صورتی برشی است که از زیردرخت راس w به خود v یا راس های بالاتر آن یال نباشد.
پس از چک کردن یال برشی با یک DFS گراف را جهت دار و با DFS دیگری همبند بودن گراف را چک میکنیم، این مراحل میتواند در طی یک عملیات DFS نیز صورت پذیرد. سودوکد آن به شرح زیر است:

```
vertexno=0;
findcut(v): \\Same as DFS except we save the number of the earliest node reachable from v.
vertexno+=1
isvisited[v]=True
first[v]=vertexno
earliest[v] = vertexno
for w in adj[v]:
    if w was not visited: \\ recursively call the function
        findcut(w)
        if earliest[w]<earliest[v]:\\if we have a back edge
            earliest[v]=earliest[w]
        \\ when going back, check for back edges:
```

```
        if earliest[w]>first[v]:
            return false
    elif w!=parent[v] (w was not v's parent) \\ which means we have seen the child sooner:
    earliest[v]  = min(earliest[v], vertexno[w]);
return true

\\ we checked for cut edges now we run a dfs and
\\ direct every edge meanwhile checking that the graph is connected:
direct(v):
    visited[v]=True
    for w in adj[v]:
        if visited[w] is False:
            directEdge(v,w) \\ we direct it be removing v from adj[w]
            direct(w)
\\right after the previous dfs we check connectivity:
for v in Graph:
        if visited[v] is False:
            return False
    return true
```

پیچیدگی زمانی : یک DFS برای چک کردن یال برشی $O(V+E)$ زمان میبرد و دو DFS برای جهت دار کردن گراف و چک کردن همبندی نیز $O(V+E)$ ، پس کل الگوریتم $O(V+E)$ زمان میبرد.