جاده کشی اولیه را یک گراف ساده و جاده کشی جدید را یک DG در نظر می‌گیریم. پس:

در DG از هر $v \in V$ به $w \in V$ و برعکس مسیر وجود دارد.

$\boxed{\text{الف}}$ ↓ ↑ $\boxed{\text{ب}}$

در SG نظیر، یال برشی وجود ندارد.

*

$\boxed{\text{الف}}$:

فرض خلف می‌کنیم: یال برشی وجود دارد، یعنی ۲ رأس $v, w \in V$ وجود دارد که درصورتی که یال بین آن‌ها قطع شود، دیگر مسیری بین این ۲ وجود نخواهد داشت.

حال یال (v,w) را در نظر بگیرید. درصورتی که در DG نظیر این یال به صورت (v,w) باشد، از سمت w به v نمی‌توان مسیری ساخت که w را به v متصل کند زیرا تنها راه ارتباطی

همان یال v و w می باشد و مسیر دیگری وجود

ندارد (زیرا (v,w) برش است.) به همین

ترتیب نمی توان یال جهت دار (w,v) را

در نظر گرفت زیرا در آن صورت مسیری از v به

w نمی توان ساخت. بنابراین فرض خلف

غلط بوده و DG نظیر که جاده کشی درستی باشد وجود

نخواهد داشت.

* : چون گراف ساده است بنابراین جاده بد (یال بد)
همان یال برشی است. زیرا بین ۲ رأس دلخواه
در گراف ساده مسیری وجود ندارد اگر و تنها اگر این ۲ رأس در
۲ مؤلفه ی همبندی باشند.
بنابراین در هر اثبات از یال برشی به عنوان یال بد استفاده شده است.

ب :

از استقرای قوی روی رئوس در گراف G استفاده می کنیم.
پایه : برای ۱ رأس که بدیهی است.
فرض : برای n رأس $k >$ می توان یک گراف جهت دار
ارائه داد که strongly connected باشد.
حکم : برای k رأس (همراه با V_k) هم چنین DG وجود دارد.

رأس V_k را اضافه می کنیم. می دانیم که $G - V_k$ یک
مؤلفه همبندی بود. چون هیچ یال برشی نداریم پس
V_k حداقل با ۲ یال به مؤلفه همبندی (فرض کنید به
w و w') متصل هستند. از فرض می دانیم از هر رأس
$u \in V_G$ (به طوری که $u \neq w, w'$) از u به w و w'
راهی وجود دارد. حال اگر در DG ۲ یال جهت دار

(w, v_k) و (v_k, w) را گزارش کنیم از هر $u \in V_G$

به w مسیری هست مانند $u, \dots, w$ پس مسیر

$u, \dots, w, v_k$ نیز در DG وجود دارد پس از هر

راسی به v_k مسیر وجود دارد.

همچنین چون از w به هر $u \in V_G$ مسیر

وجود دارد پس مسیر $v_k, w, \dots, u$ در DG

تعریف می‌شود بنابراین از v_k به هر راسی مسیر

وجود دارد.

یال های دیگر v_k را به صورت دلخواه جهت

دهی می‌کنیم.

بنابراین حکم اثبات می‌شود.

الگوریتم : برای پیدا کردن جهت های یال ها از DFS استفاده می کنیم. یعنی به ازای هر پیمایش یال، جهت آن مشخص کرده و یال را حذف می کنیم. همچنین باید وجود یا عدم وجود یال برشی را نیز دریابیم. برای این کار در درخت DFS به ازای هر نود v باید چک کنیم که آیا یک یال Back از رئوس زیر درخت v، به v یا نود بالاتر از v وجود دارد یا خیر. برای این کار، به ازای هر نود در حال پیمایش (هنگام برگشت) این شرط را به ازای همه ی فرزندان نود انجام می دهیم.

```
let t = 0
def dfs(G, v):
    t += 1
    v.first = t
    let b = ∞ & c = ∞
    foreach u in adjacent(v):
        if u.first is not set: // not visited
            remove edge (u,v)   O(1)
            report (u,v) as directed edge   O(1)
            b = min(dfs(G,u), b)
        else:
            c = min(c, u.first)   O(1)
    t += 1
    v.post = t
    if (b < v.first)
        raise(IMPOSSIBLE)
    return min(c,b)
```

O(deg v) itteration

```
def convert_SG_to_SCDG(G):
O(1)       for one v0 ∈ V_G:
O(E) <~~       dfs(G, v0)
O(V)  {    foreach v ∈ V_G:
                if v.first is not set: //not visited
                    raise(IMPOSSIBLE)
```

(چک می‌شود تمامی رئوس دیده شده باشد، یعنی
گراف ورودی ناهمبندی نداشته باشد.)

$$\underbrace{O(|E|)}_{DFS} + \underbrace{O(|V|)}_{\text{چک همبندی}} = O(|V|+|E|)$$

باتوجه به تحلیل DFS که در کلاس انجام شد، و باتوجه
به این که عملیات های اضافه شده به DFS همگی
$O(1)$ بوده اند بنابراین پیچیدگی زمانی این الگوریتم
$O(|V|+|E|)$ خواهد شد.