سوال ۸ بلندترین مسیر در DAG

فرض کنید یک گراف جهت‌دار با n راس و m یال داریم به این صورت که اگر بین دو راس $i < j$ یالی وجود داشته باشد جهت آن از i به j خواهد بود. الگوریتمی با پیچیدگی زمانی $O(n+m)$ ارائه دهید که بلندترین مسیر در این گراف را پیدا کند.

با توجه به صورت سؤال که گفته گراف جهت دار است، شرط اگر بین i و j طوریکه j > i یال باشد آن یال از i به j است باعث میشود گراف دور نداشته باشد پس طبق title سوال با یک گراف جهت دار بدون دور یا DAG سروکار داریم. همچنین مجدداً به سبب شرط اگر بین i و j طوریکه j > i یال باشد آن یال از i به j است، ترتیب رئوس 1, 2, ..., n یک Topological Sort از این رئوس است. پس برای پیدا کردن بلندترین مسیر در این گراف از این ترتیب Topological Sort استفاده می‌کنیم و از رأس 1 شروع کرده و پیش می‌ریم تا رأس n ام.

Dynamic Programming ← L[j] : طول بلندترین مسیری هست که به رأس j ختم میشه

این مقدار بصورت بازگشتی با بررسی تمام رئوس قبلی (i هایی که i<j) که به j یال دارند بصورت زیر بدست میاد: $L[j] = \max_{(i,j)\in E}(L[i]+1)$ پس رأسی که هیچ یال ورودی نداره (L[j] = 0)

پس در آغاز آرایه L را به طول n با مقادیر اولیه صفر به ازای همه اندیسهاش تشکیل می دیم، برای اینکه خود مسیر رو هم بتونیم خروجی بدیم آرایه parent به طول n با مقادیر اولیه null تعریف میکنیم برای ذخیره رأس والد در مسیر یافته، حالا از رأس 1 = i شروع میکنیم و به سمت n حرکت میکنیم و برای هر i، همه یالهای خروجی از i به همسایه هاش رو بررسی می کنیم و مقدار L[j] را آپدیت می کنیم ← L[j] = Max(L[j], L[i]+1) پس اگر L[j]<L[i]+1 مقدار L[j] آپدیت شده و parent[j] را هم i قرار میدیم. در نهایت پس از پایان حلقه، طول بلندترین مسیر موجود در کل گراف میشه ماکسیمم مقدار کل آرایه L. برای چاپ بلندترین مسیر هم از رأسی که ماکسیمم L رو داشته شروع میکنیم و به کمک آرایه parent به عقب برمی گردیم تا به رأسی برسیم که parent آن null هست.

این رئوس طی شده رو بصورت برعکس به عنوان طولانی ترین مسیر از ابتدا تا انتها اعلام می کنیم.

Pseudocode:

رئوس گراف که از 1 تا n هستند ← V

یالها ← E

```
LongestPath(V, E):
    L = Array of size n, initialized to 0
    Parent = Array of size n, initialized to NULL
    for u = 1 to n:                                   | O(m+n)
        for each outgoing edge (u, v):                |
            if L[u] + 1 > L[v]:                       |
                L[v] = L[u] + 1                       |
                Parent[v] = u                         |
    max_length = 0                                    | O(n)
    end_node = 1                                      |
    for i = 1 to n:                                   |
        if L[i] > max_length:                         |
            max_length = L[i]                         |
            end_node = i                              |
    path = empty list                                 |
    current = end_node                                |
    while current != NULL:                            |
        add current to the end of path                |
        current = parent[current]                     |
    Reverse(path)                                     |
    Return max_length, path                           |
```

حلقه اصلی روی تمام رئوس آرایه حرکت کرده و داخل آن تمام یالهای خروجی هر رأس را بررسی می کند

پس همسایه‌های گراف دقیقاً یکبار بررسی شده و زمان اجرا $O(m+n)$ است ← یافتن هر مسیر هم در $O(n)$ انجام میشه.