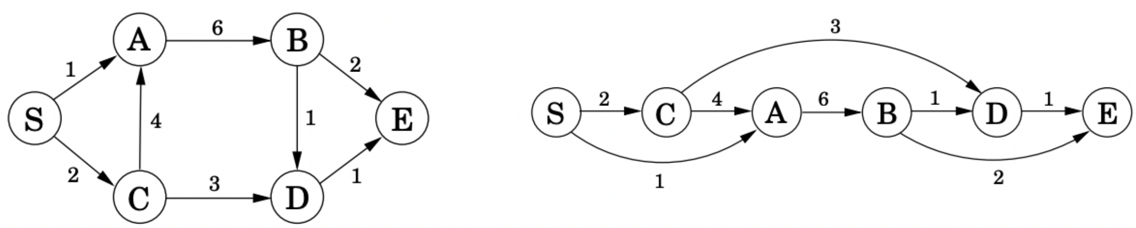
Chap6. Dynamic programming

6.1 shortest paths in dags, revisited

Figure 6.1 A dag and its linearization (topological ordering).



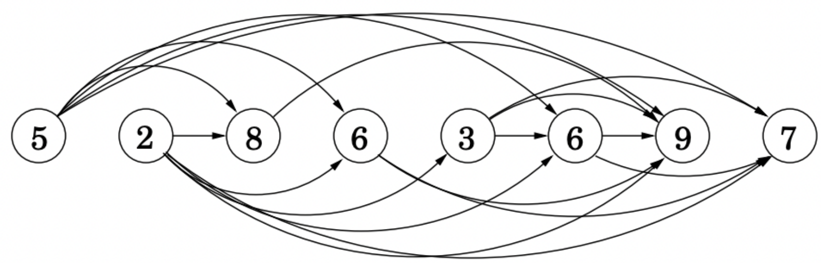
- DAG(directed acyclic graphs)의 특징을 이용해, 오른쪽 그림과 같이 노드들을 linearized 하게 할 수 있음
- DAG 를 linearized 함으로써, 노드 SS (시작점) 로부터 다른 노드 사이의 거리를 알아내는데 도움을 줄 수 있음
    - DD 까지의 거리를 알고 싶다면, predecessors 인 BB 와 CC 를 조사
        - 따라서, 2 개의 길만 비교하기만 하면 됨
        - dist(D)=min{dist(B)+1,dist(C)+3}dist(D)=min{dist(B)+1,dist(C)+3}
    - 모든 노드에 대해 이러한 관계로 표현할 수 있음
    - 만약 우리가, 위 그림의 왼쪽에서 오른쪽 방향으로 distdist 를 계산한다고 하면,
노드 vv 에 도달할 때까지, dist(v)dist(v)를 계산하기 위해 필요한 모든 정보들을 이미 다 갖고 있다고 말할 수 있다.
    - 따라서, 모든 거리들을 단 하나의 single pass 로 계산가능함 (아래와 같이)

initialize all $\texttt{dist}(\cdot)$ values to ∞
$\texttt{dist}(s) = 0$
for each $v \in V \backslash \{s\}$, in linearized order:
    $\texttt{dist}(v) = \min_{(u,v)\in E}\{\texttt{dist}(u) + l(u,v)\}$

6.2 Longest increasing subsequences

- <u>TASK : to find the increasing subsequence of greatest length</u>

Figure 6.2 The dag of increasing subsequences.



- solution space 를 더 잘 이해하기 위해서, 가능한 모든 transitions 를 가진 그래프를 그려보자.
  - 각 aiai 에 대해 노드 ii 를 생각
  - i<j and ai<aji<j and ai<aj 를 만족하는, 가능한 모든 aiai 와 ajaj 를 directed edges (i,j)(i,j)로 연결
- 이렇게 되면, one-to-one correspondence 가 존재
  - between increasing subsequences and paths in dag.
- 결국, 우리의 목표는 dag 안에서 가장 긴 path 를 찾는 것으로 단순화할 수 있게 된다.

for $j = 1, 2, \ldots, n$:
    $L(j) = 1 + \max\{L(i) : (i, j) \in E\}$
return $\max_j L(j)$

6.3 Edit distance

S	−	N	O	W	Y		−	S	N	O	W	−	Y
S	U	N	N	−	Y		S	U	N	−	−	N	Y

Cost: 3 **Cost: 5**

- costcost : 문자가 서로 다른 컬럼의 개수
- 두 문자열 사이의 editdistanceeditdistance 는 그들의 best alignment 의 costcost 를 말함

- 결국, Edit distance = minimum number of editsedits 를 말함.