6.5 Chain matrix multiplication

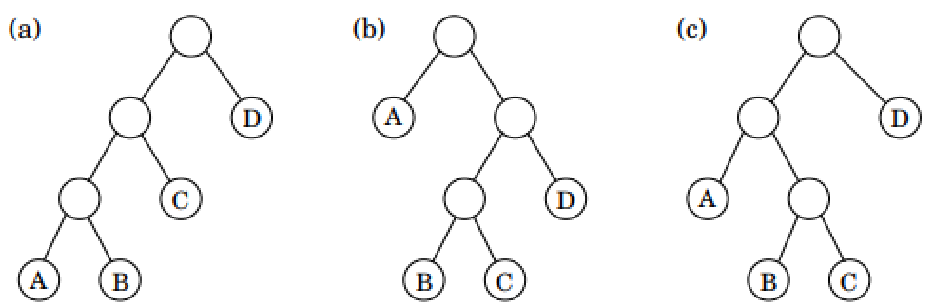
4개의 행렬에 대해서 행렬곱을 하려고 한다.

행렬곱은 어떻게 묶어서 계산하느냐에 따라 cost가 달라질 수도 있음

Parenthesization	Cost computation	Cost
$A \times ((B \times C) \times D)$	$20 \cdot 1 \cdot 10 + 20 \cdot 10 \cdot 100 + 50 \cdot 20 \cdot 100$	120, 200
$(A \times (B \times C)) \times D$	$20 \cdot 1 \cdot 10 + 50 \cdot 20 \cdot 10 + 50 \cdot 10 \cdot 100$	60, 200
$(A \times B) \times (C \times D)$	$50 \cdot 20 \cdot 1 + 1 \cdot 10 \cdot 100 + 50 \cdot 1 \cdot 100$	7, 000

- 그림에서 볼 수 있듯이, greedy 접근은 실패할 수도 있음(매번 the cheapest matrix multiplication 을 선택)
- $A_1 \times A_2 \times \cdots \times A_n$A1×A2×⋯×An 를 계산하고 싶을 때, optimal order 을 어떻게 결정하는 것이 좋을까?

Figure 6.7 **(a)** $((A \times B) \times C) \times D$; **(b)** $A \times ((B \times C) \times D)$; **(c)** $(A \times (B \times C)) \times D$.



각각의 트리에 대해 해보는 방법말고, DP로!

서브트리에 해당하는 subproblems 을 다음과 같이 정의할 수 있음

$C(i, j) =$ **minimum cost of multiplying** $A_i \times A_{i+1} \times \cdots \times A_j$.

C(i,j)의 서브트리에 생각해보자

- 이 서브트리의 가장 윗 지점인 첫 분기점은, 두 가지로 곱을 나눌 것임
- $A_i \times \cdots \times A_k$ 와 $A_{k+1} \times \cdots \times A_j$,$i \le k < j$
- 즉, $C(i,j)$는 $C(i,k)+C(k+1,j)+m_{i-1} \cdot m_k \cdot m_j$(마지막 항 = 두 행렬곱의 연산량)
- 이렇게 나누는 k 값 중에서 가장 작은 값을 찾아야 하므로, 다음과 같은 최종 식을 만들 수 있음
- $C(i,j)=\min_{i \le k < j}\{C(i,k)+C(k+1,j)+m_{i-1} \cdot m_k \cdot m_j\}$

그래서 알고리즘은

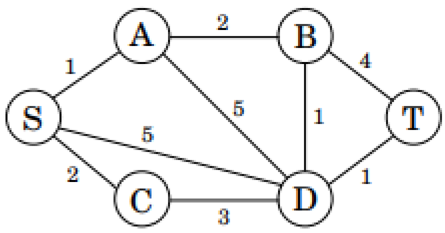
$$
\begin{aligned}
&\texttt{for } i=1 \texttt{ to } n\texttt{:} \quad C(i,i)=0\\
&\texttt{for } s=1 \texttt{ to } n-1\texttt{:}\\
&\quad \texttt{for } i=1 \texttt{ to } n-s\texttt{:}\\
&\qquad j=i+s\\
&\qquad C(i,j)=\min\{C(i,k)+C(k+1,j)+m_{i-1}\cdot m_k\cdot m_j : i\le k<j\}\\
&\texttt{return } C(1,n)
\end{aligned}
$$

6.6 Shortest paths

Shortest reliable path

최단 경로에 가까우면서, 각 노드들을 최소한으로 거쳐가는 경로를 찾고 싶다

Figure 6.8 We want a path from s to t that is both short *and* has few edges.



-
- 각 노드 v 와 정수 i≤ki≤k 에 대해, dist(v,i)를 the length of the shortest path from s to v that uses i edges 로 정의
- dist(v,0)은 모두 ∞로 정의하고, 정점 s 에 대해서만 0 으로 초기화.
- 아래식을 이용해서 update

$$\texttt{dist}(v,i) = \min_{(u,v)\in E}\{\texttt{dist}(u,i-1)+\ell(u,v)\}.$$

All-pairs shortest paths

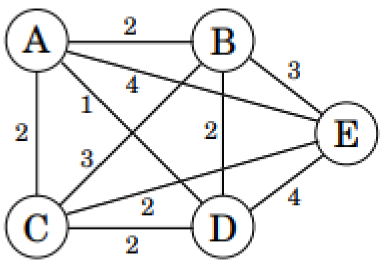
모든 쌍에 대해 최단경로를 알고 싶다!

```
for i = 1 to n:
    for j = 1 to n:
        dist(i,j,0) = ∞
for all (i,j) ∈ E:
    dist(i,j,0) = ℓ(i,j)
for k = 1 to n:
    for i = 1 to n:
        for j = 1 to n:
            dist(i,j,k) = min{dist(i,k,k−1) + dist(k,j,k−1),  dist(i,j,k−1)}
```

The traveling salesman problem

정확히 한번씩 모든 도시를 들렀다가 다시 돌아오는 최단 이동경로는?

Figure 6.9 The optimal traveling salesman tour has length 10.




```
C({1},1) = 0
for s = 2 to n:
    for all subsets S ⊆ {1,2,...,n} of size s and containing 1
        C(S,1) = ∞
        for all j ∈ S, j ≠ 1:
            C(S,j) = min{C(S − {j}, i) + d_ij : i ∈ S, i ≠ j}
return min_j C({1,...,n}, j) + d_j1
```

- 최대 $2^n \cdot n$ 개의 subproblem 이 존재하고, 각 문제를 풀 때 linear time 이 걸린다.
- 따라서 $O(n^2 2^n)$

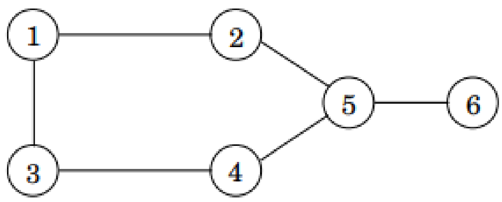
6.7 Independent sets in trees

independent set이란?

인접하지 않는 노드들의 집합을 뽑는 것

- 예시) {1,5}=가능, {1,4,5}=불가능, {2,3,6}=가능+가장 큰 indep. set 임

Figure 6.10 The largest independent set in this graph has size 3.



이를 위한 알고리즘은

- 아무 노드 r을 root로 잡고, 트리를 전개한다.
- 이제 각 노드를 서브트리로 정의할 수 있게 된다.
  - 그 노드 밑으로 걸려 있는 트리로 생각하면 된다.
- subproblem
  - I(u)=size of largest indep. set of subtree hanging from u
- Our final goal
  - I(r)

DP는 rooted tree에서 bottom-up 방식으로 접근

- 특정 노드 u 밑에 모든 자식 노드들 w의 I(w)값을 안다고 하자.
- 그렇다면, 노드 u의 I(u)를 어떻게 계산할 수 있을까?
  - 2가지 경우로 생각하면 끝
    - 1) u가 포함되는 경우
    - 2) u가 포함되지 않는 경우
- 1) 의 경우 = u의 자식 노드들은 포함되지 않았다는 뜻 = 자식의 자식 노드들의 I(w) 합에 +1(u가 추가될 것임)
- 2) 의 경우 = u의 자식 노드들의 I(w)합을 그대로 들고옴

$$I(u) = \max\left\{1 + \sum_{\text{grandchildren } w \text{ of } u} I(w), \sum_{\text{children } w \text{ of } u} I(w)\right\}.$$

→ subproblems의 개수는 vertices의 개수와 정확히 동일하다.