‘‘잔디밭의 개미굴’’ 문제 풀이

작성자: 구재현

편의상 그래프 이론의 용어를 사용한다. 즉, 개미굴은 그래프이고, 방은 정점, 통로는 간선이다.

부분문제 1

모든 배정의 경우의 수는 2^N 가지이다. 고로, 그래프가 주어졌을 때, 최대의 개미가 들어가는 배정을 $O(2^N \times N)$ 시간에 찾을 수 있다. 원래 그래프에 대한 최적 배정을 찾고, 모든 $N(N-1)/2$ 개의 추가 간선에 대해서 여전히 최적 배정의 크기가 유지되는가를 확인한다. 시간 복잡도는 $O(2^N \times N^3)$ 이다.

부분문제 3

최적 배정의 크기를 찾는 문제는 흔히 **최대 독립 집합** 문제라고 하여 일반적인 그래프에서 효율적으로 푸는 것이 불가능하다. 개미굴이 트리라는 점을 사용하여 문제를 효율적으로 해결해야 한다.

다음과 같은 사실이 크게 도움이 된다:

Theorem. $|V| \geq 2$ 인 임의의 연결 그래프 $G = (V, E)$ 와 어떠한 차수 1 정점 v 에 대해, v 를 포함하는 최대 독립 집합이 존재한다.

Proof. 그렇지 않다고 가정하자. 임의의 최대 독립 집합을 골랐을 때, 이 집합에는 v 에 인접한 유일한 정점 w 가 포함된다. 만약 포함되지 않으면 v 를 넣을 수 있고 최대성에 모순이기 때문이다. 집합에서 w 를 지우고 v 를 넣는다고 해도, 여전히 인접한 두 정점이 모두 선택되는 일은 없다. 고로 v 를 포함하는 최대 독립 집합을 찾았고, 가정에 모순이다. ■

이를 사용하면 간선이 추가되지 않은 트리에서는 쉽게 문제를 해결할 수 있다. DFS를 사용하여 재귀적으로 해결하는데, 리프에서는 무조건 해당 정점을 독립 집합에 추가하고, 그 외 노드들에 대해서는 만약 자식이 하나라도 선택되었다면 추가하지 않고, 그렇지 않다면 추가할 수 있기 때문이다. 이 알고리즘은, 임의의 차수 1 이하인 노드를 찾아서 이 노드와 인접한 노드를 반복적으로 지워주는 알고리즘과 동치이기 때문에, 정당하다.

트리에 간선이 하나 추가되었을 때, 그래프는 하나의 사이클의 각 노드에 트리가 달린 꼴을 한다. 각 사이클 노드에 달린 트리에 대해 DFS로 독립적으로 해결한다. 이 경우, 사이클에 속하지 않은 노드들에 대해서는 차수 1인 노드가 반복적으로 지워진 것과 동치이기 때문에 올바른 배정이지만, 사이클에 속한 노드들은 인접한 간선들을 억지로 무시한 것이기 때문에 올바른 배정이 아니고, 실제로 사이클 상 인접한 노드가 골라지는 배정을 얻게 될 것이다.

이렇게 얻은 배정에서 어떠한 사이클 상 노드가 골라지지 않았다면, 이 노드에 인접한 사이클 밖 노드가 선택되었기 때문에, 이 노드는 제거된 것으로 취급하여도 좋다. 사이클 상 노드가 골라졌다면, 이 노드는 제거되지 않았다고 생각할 수 있다. 여기서 두 가지 케이스가 존재한다.

- 만약 모든 사이클 상 노드가 제거되지 않았다면, 사이클 밖 배정을 고민할 필요가 없으니 사이클 상에서 독립 집합을 고르면 된다. 즉, 사이클 크기를 2로 나눈 몫이 답에 추가된다.
- 사이클 상 제거된 노드가 있다면, 사이클 안에서 제거되지 않은 노드들은 선의 형태를 이룬다. 각각의 선에서 독립 집합을 고른 후 합쳐주면 된다. 즉, 각 선의 크기를 2로 나눈 값을 반올림하고, 답에 더한다.

트리, 그리고 트리에 간선이 하나 추가된 경우 모두 $O(N)$ 에 최적 배정의 크기를 구할 수 있다. 이를 $O(N^2)$ 번 반복하니 시간 복잡도는 $O(N^3)$ 이다.

부분문제 4

부분문제 4는 $O(N^2)$ 알고리즘을 구현하였을 때 정답을 받을 수 있게 구성되었다. (효율적인 알고리즘은 부분문제 5 이상도 맞을 수 있다.) 이 문제의 $O(N^2)$ 풀이는 다양하지만, 여기서는 의도된 만점 풀이의 방향에 있는 $O(N^2)$ 풀이를 설명한다.

다음 사실을 사용한다. 주어진 트리를 T 라 하고, T 에 간선 (i,j) 가 추가된 것을 $T+(i,j)$ 라고 하자.

Theorem. $S = \{v | v \in V(T), v$ 가 모든 최대 독립 집합에 속함$\}$ 이라고 하자. (i,j) 가 평화로운 쌍이 아니라는 것과, $i \in S, j \in S$ 임이 동치이다.

Proof. $\rightarrow$: (i,j) 가 평화로운 쌍이 아니라면, T 의 모든 최대 크기 독립 집합을 보았을 때, i 와 j 는 모두 그 집합에 속해야 한다. 그렇지 않다면 해당 집합을 $T+(i,j)$ 의 독립 집합으로 취할 수 있어, 평화로운 쌍이 아니라는 가정에 모순이기 때문이다.

$\leftarrow$: 대우명제를 증명한다. (i,j) 가 평화로운 쌍이라면, $T+(i,j)$ 의 최대 독립 집합의 크기가 T 의 최대 독립 집합의 크기와 동일하다. $T+(i,j)$ 의 독립 집합은 T 의 독립 집합이니, 모든 T 의 최대 독립 집합을 살펴보았을 때, 이 중 $T+(i,j)$ 에서도 최대 독립 집합인 것이 존재한다. 이는, i,j 중 하나는 이 집합에 속하지 않는다는 것이다. 고로 i,j 가 모두 S 에 속할 수 없다. ■

Theorem을 사용하면, S 의 원소를 구했을 때 답을 얻는 것은 간단하다. S 의 원소를 구하기 위해서는, 각 정점을 지워보고, 지웠을 때 생기는 서브트리들에 대해 부분문제 3의 방법으로 최대 독립 집합의 크기를 구하여, 그 크기의 합이 원래 독립 집합의 크기보다 작은지 확인하면 된다. 시간 복잡도는 $O(N^2)$ 이다.

부분문제 7

부분문제 3에서 트리의 최대 독립 집합을 구하는 과정을 살펴보면, 알고리즘은 임의의 정점을 루트로 잡은 후, 계산 과정에서 트리의 모든 서브트리에 대해서 최대 독립 집합의 크기를 구한다는 사실을 알 수 있다. 고로, 부분문제 4에서 우리가 구하는 *지웠을 때 생기는 서브트리 중*, 루트 방향으로 향하는 서브트리가 아닌 모든 서브트리에 대해서는 $O(N)$ 시간에 답을 전부 구해놓을 수 있다. 이 정보를 구해 두면, 모든 루트 방향으로 향하는 서브트리에 대해서도 문제를 해결할 수 있다.

구체적으로, 부분문제 3의 알고리즘은 각 노드 v 에 대해서 v 의 서브트리의 독립 집합 크기, 그리고 v 가 그 독립 집합에 선택되었는지 여부를 계산한다. 여기서 우리가 빠르게 계산하고 싶은 값은, 각 노드 v 에 대해서, v 를 루트로 하였을 때 $par(v)$ 의 서브트리에서 독립 집합의 크기, 그리고 $par(v)$ 가 그 독립 집합에 선택되었는지의 여부이다.

이 값을 루트에서부터 top-down으로 계산하자. 각 노드 v 에 대해서, 위 답의 계산 결과는 거의 $par(v)$ 의 서브트리 계산 결과와 일치하는데, 다른 점은 1) v 로 가는 서브트리를 빼야 한다 2) $par(v)$ 에서 $par(par(v))$ 로 가는 서브트리를 더해야 한다 는 점이다. 2) 의 서브트리 값은 계산 순서에 따라서 이미 알고 있고, 1) 의 서브트리 값은 초기에 다 구해 놓았으니, 이제 이 변홧값을 계산해야 한다. 이 값을 $deg(v)$ 에 계산하면, 전체 시간 복잡도가 $O(N^2)$ 가 된다. $onchild(v)$ 를, v 의 자식 중 독립 집합에 포함된 자식의 수라고 하자. 이 값을 맨 처음 서브트리에 대해 문제를 해결할 때 전처리해 두면, $par(v)$ 가 v 가 루트일 때 독립 집합에 들어가는 지 여부를 빠르게 계산할 수 있다. 이 값이 저장되면, 독립 집합 크기는 이 값과 상술한 계산 결과들을 조합하여 계산할 수 있다. 고로 각 노드에 대해 필요한 값을 $O(1)$ 에 계산할 수 있다. 시간 복잡도는 $O(N)$ 이다.