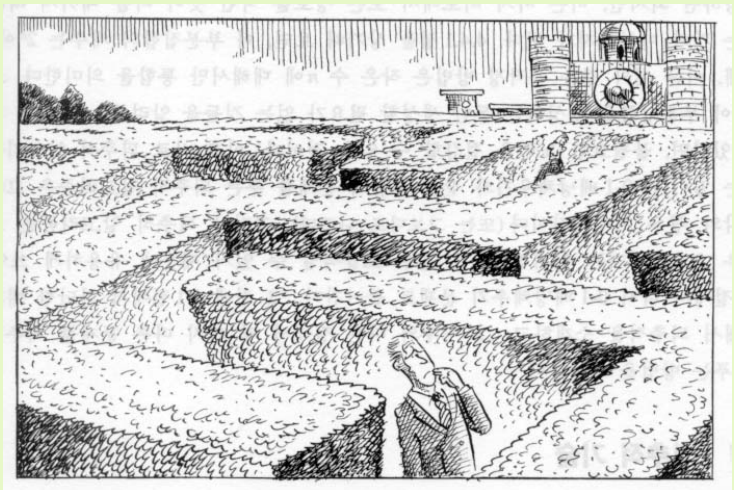
되추적
(Backtracking)

강의 슬라이드 #5
도경구



도경구 역, 알고리즘, 사이텍미디어, 1999

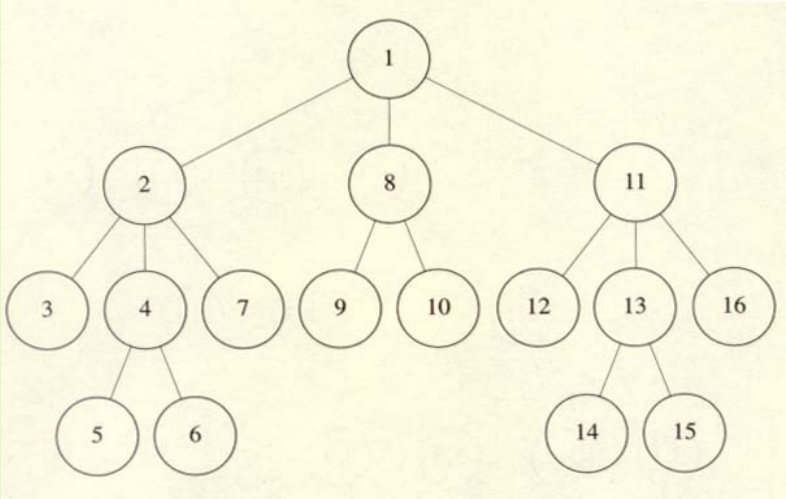
깊이우선검색(Depth-First Search)

- 뿌리마디(root)가 되는 마디(node)를 먼저 방문한 뒤, 그 마디의 모든 후손마디(descendant)들을 차례로 (보통 왼쪽에서 오른쪽으로) 방문한다.
  (= preorder tree traversal).

```
void depth_first_tree_search (node v) {
     node u;
     visit v;
     for (each child u of v)
          depth_first_tree_search(u)
}
```

2001-03-19 알고리즘 강의 슬라이드 5 3

깊이우선검색의 예



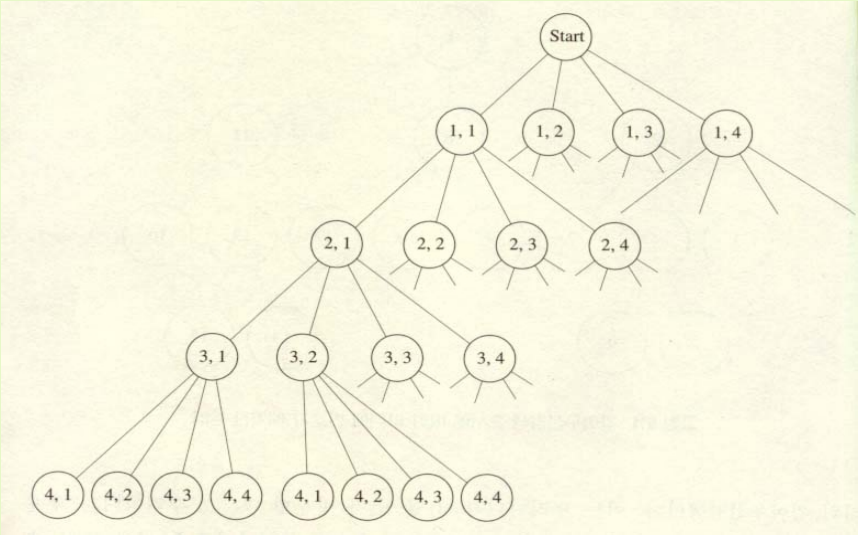
2001-03-19 알고리즘 강의 슬라이드 5 4

4-Queens 문제

- 4개의 Queen을 서로 상대방을 위협하지 않도록 4 × 4 서양장기(chess)판에 위치시키는 문제이다. 서로 상대방을 위협하지 않기 위해서는 같은 행이나, 같은 열이나, 같은 대각선 상에 위치하지 않아야 한다.
- **무작정 알고리즘**: 각 Queen을 각각 다른 행에 할당한 후에, 어떤 열에 위치하면 해답은 얻을 수 있는지를 차례대로 점검해 보면 된다. 이때, 각 Queen은 4개의 열 중에서 한 열에 위치할 수 있기 때문에, 해답을 얻기 위해서 점검해 보아야 하는 모든 경우의 수는 4 × 4 × 4 × 4 = 256가지가 된다.

2001-03-19 알고리즘 강의 슬라이드 5 5

4-Queens 문제의 상태공간트리



2001-03-19 알고리즘 강의 슬라이드 5 6

상태공간트리(State Space Tree)

- 뿌리마디에서 잎마디(leaf)까지의 경로는 해답후보(candidate solution)가 되는데, 깊이우선검색을 하여 그 해답후보 중에서 해답을 찾을 수 있다.
- 그러나 이 방법을 사용하면 해답이 될 가능성이 전혀 없는 마디의 후손마디(descendant)들도 모두 검색해야 하므로 비효율적이다.

되추적 기술

- **마디의 유망성**:
  - ✓ 전혀 해답이 나올 가능성이 없는 마디는 유망하지 않다(non-promising)고 하고,
  - ✓ 그렇지 않으면 유망하다(promising)고 한다.
- 되추적이란?
  - ✓ 어떤 마디의 유망성을 점검한 후, 유망하지 않다고 판정이 되면 그 마디의 부모마디(parent)로 돌아가서("backtrack") 다음 후손마디에 대한 검색을 계속하게 되는 절차이다.

되추적 알고리즘의 개념

- 되추적 알고리즘은 상태공간트리에서 깊이우선검색을 실시하는데,
  - ✓ 유망하지 않은 마디들은 가지쳐서(pruning) 검색을 하지 않으며,
  - ✓ 유망한 마디에 대해서만 그 마디의 자식마디(children)를 검색한다.
- 이 알고리즘은 다음과 같은 절차로 진행된다.
  1. 상태공간트리의 깊이우선검색을 실시한다.
  2. 각 마디가 유망한지를 점검한다.
  3. 만일 그 마디가 유망하지 않으면, 그 마디의 부모마디로 돌아가서 검색을 계속한다.

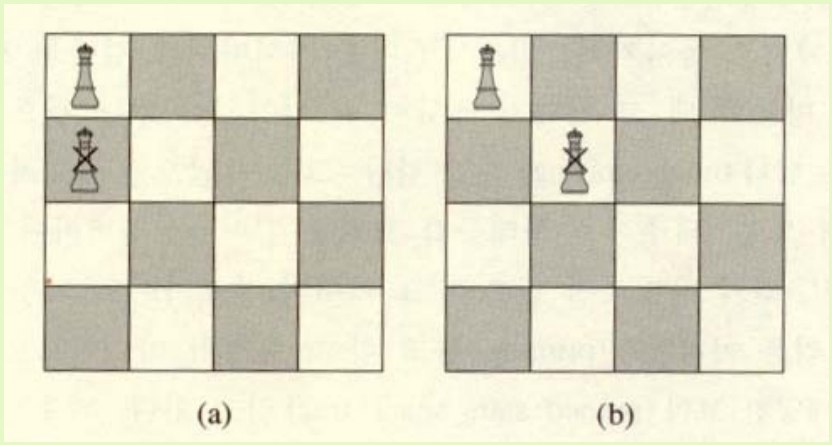
2001-03-19 알고리즘 강의 슬라이드 5 9

되추적 알고리즘

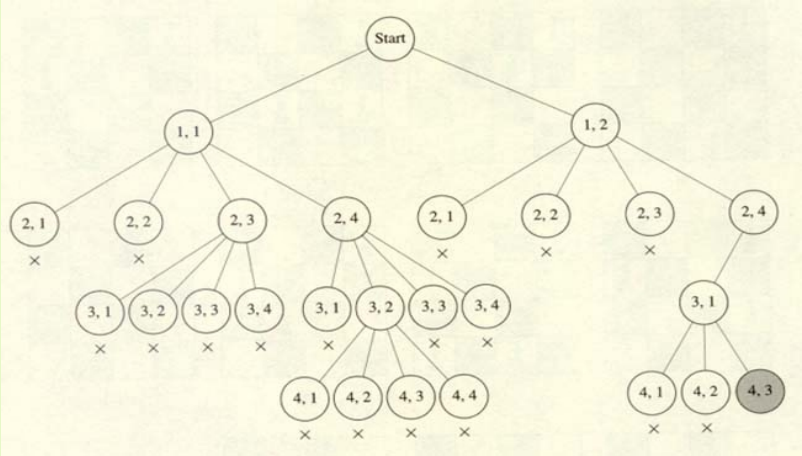
- 일반 되추적 알고리즘:

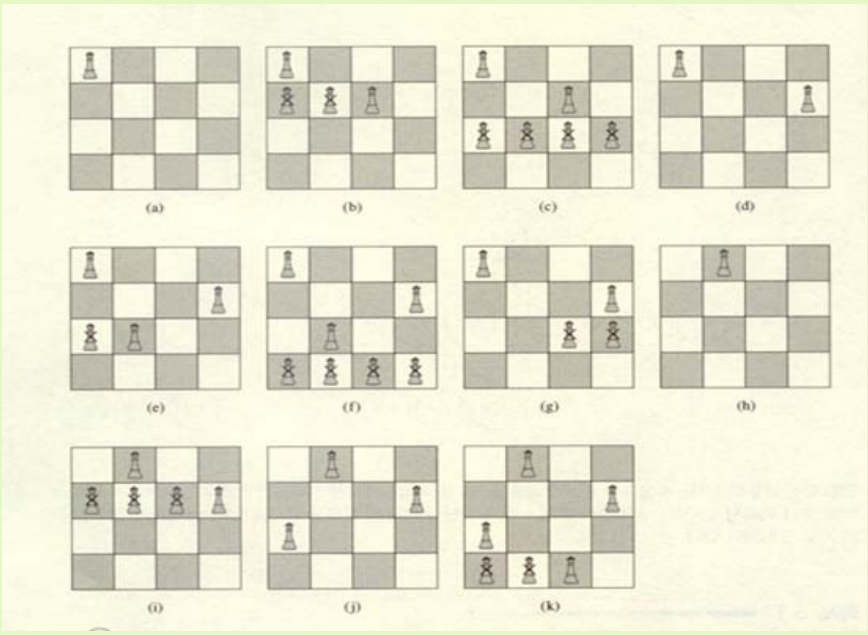
```
void checknode (node v) {
  if (promising(v))
    if (there is a solution at v)
      write the solution;
    else
      for (each child u of v)
        checknode(u);
}
```

2001-03-19 알고리즘 강의 슬라이드 5 10



4-Queens 문제의 상태공간트리 (되추적)





깊이우선검색 vs. 되추적

- 검색하는 마디 개수의 비교
  - ✓ 순수한 깊이우선검색 = 155 마디
  - ✓ 되추적 = 27 마디

4-Queens 문제: 되추적(개량)

```
void expand (node v) {
  for (each child u of v)
    if (there is a solution at u)
      write the solution;
    else
      expand(u);
}
```

- 이 개량된 알고리즘은 유망성 여부의 점검을 마디를 방문하기 전에 실시하므로, 그만큼 방문할 마디의 수가 적어져서 더 효율적이다.
- 그러나 일반 알고리즘이 이해하기는 더 쉽고, 일반 알고리즘을 개량된 알고리즘으로 변환하기는 간단하므로, 앞으로 이 강의에서의 모든 되추적 알고리즘은 일반 알고리즘과 같은 형태로 표시한다.

2001-03-19 알고리즘 강의 슬라이드 5 15

n-Queens 문제

- *n*개의 Queen을 서로 상대방을 위협하지 않도록 $n \times n$ 서양장기(chess) 판에 위치시키는 문제이다. 서로 상대방을 위협하지 않기 위해서는 같은 행이나, 같은 열이나, 같은 대각선 상에 위치하지 않아야 한다.
- ***n*-Queens 문제의 되추적 알고리즘**: 4-Queens 문제를 *n*-Queens 문제로 확장 시키면 된다.

2001-03-19 알고리즘 강의 슬라이드 5 16

n-Queens 문제의 분석 I

- 상태공간트리 전체에 있는 마디의 수를 구함으로서, 가지 친 상태공간트리의 마디의 개수의 상한을 구한다. 깊이가 i인 마디의 개수는 n^i개 이고, 이 트리의 깊이는 n이므로, 마디의 총 개수는 상한(upper bound)은:

$$1 + n + n^2 + n^3 + \cdots + n^n = \frac{n^{n+1} - 1}{n - 1}$$

  따라서 $n = 8$일 때, $\frac{8^9 - 1}{8 - 1} = 19{,}173{,}961$.
- 그러나 이 분석은 별 가치가 없다. 왜냐하면 되추적함으로서 점검하는 마디 수를 얼마나 줄였는지 상한값을 구해서는 전혀 알 수 없기 때문이다.

2001-03-19 알고리즘 강의 슬라이드 5 17

n-Queens 문제의 분석 II

- 유망한 마디만 세어서 상한을 구한다. 이 값을 구하기 위해서는 어떤 두 개의 Queen이 같을 열상에 위치할 수 없다는 사실을 이용하면 된다.
- 예를 들어 $n = 8$일 경우를 생각해 보자. 첫번째 Queen은 어떤 열에도 위치시킬 수 있고, 두 번째는 기껏해야 남은 7열 중에서만 위치시킬 수 있고, 세 번째는 남은 6열 중에서 위치시킬 수 있다. 이런 식으로 계속했을 경우 마디의 수는 $1 + 8 + 8 \times 7 + 8 \times 7 \times 6 + \ldots + 8! = 109{,}601$가 된다. 이 결과를 일반화 하면 유망한 마디의 수는

$$1 + n + n(n-1) + n(n-1)(n-2) + \cdots + n!$$

  을 넘지 않는다.

2001-03-19 알고리즘 강의 슬라이드 5 18

사색

- 위 2가지 분석 방법은 알고리즘의 복잡도를 정확히 얘기 해주지 못하고 있다.
- 왜냐하면:
  - ✓ 대각선을 점검하는 경우를 고려하지 않았다. 따라서 실제 유망한 마디의 수는 훨씬 더 작을 수 있다.
  - ✓ 유망하지 않은 마디를 포함하고 있는데, 실제로 해석의 결과에 포함된 마디 중에서 유망하지 않은 마디가 훨씬 더 많을 수 있다.

2001-03-19 알고리즘 강의 슬라이드 5 19

n-Queens 문제의 분석 III

- 유망한 마디의 개수를 정확하게 구하기 위한 유일한 방법은 실제로 알고리즘을 수행하여 구축된 상태공간트리의 마디의 개수를 세어보는 수 밖에 없다.
- 그러나 이 방법은 진정한 분석 방법이 될 수 없다. 왜냐하면 분석은 알고리즘을 실제로 수행하지 않고 이루어져야 하기 때문이다.

2001-03-19 알고리즘 강의 슬라이드 5 20

알고리즘의 수행시간 비교

n	알고리즘 1†로 검사한 마디의 개수	알고리즘 2‡로 검사한 해답후보의 개수	되추적으로 검사한 마디의 개수	되추적으로 유망함을 알아낸 마디의 개수
4	341	24	61	17
8	19,173,961	40,320	15,721	2057
12	9.73×10^{12}	4.79×10^{8}	1.01×10^{7}	8.56×10^{5}
14	1.20×10^{16}	8.72×10^{10}	3.78×10^{8}	2.74×10^{7}

* 해답을 모두 찾는 데 필요한 검사횟수를 나타냄.
† 알고리즘 1은 되추적 없이 상태공간 트리를 깊이우선 검색함.
‡ 알고리즘 2는 각 여왕말을 다른 행과 열에 위치하는 $n!$개의 해답후보를 생성함.

2001-03-19 알고리즘 강의 슬라이드 5 21

Monte Carlo 기법을 사용한 백트랙킹 알고리즘의 수행시간 추정

- Monte Carlo 기법은 어떤 입력이 주어졌을 때 점검하게 되는 상태공간트리의 "전형적인" 경로를 무작위(random)로 생성하여 그 경로 상에 있는 마디의 수를 센다. 이 과정을 여러 번 반복하여 나오는 결과의 평균치를 추정치로 한다.
- 이 기법을 적용하기 위해서는 다음 두 조건을 반드시 만족하여야 한다.
  - ✓ 상태공간트리의 같은 수준(level)에 있는 모든 마디의 유망성 여부를 점검하는 절차는 같아야 한다.
  - ✓ 상태 공간트리의 같은 수준에 있는 모든 마디는 반드시 같은 수의 자식 마디를 가지고 있어야 한다.

  n-Queens 문제는 이 두 조건을 만족한다.

2001-03-19 알고리즘 강의 슬라이드 5 22

Monte Carlo 기법을 사용한 백트랙킹 알고리즘의 수행시간 추정

1. 뿌리마디의 유망한 자식마디의 개수를 m_0이라고 한다.
2. 상태공간트리의 수준 1에서 유망한 마디를 하나 랜덤하게 정하고, 이 마디의 유망한 자식마디의 개수를 m_1이라고 한다.
3. 위에서 정한 마디의 유망한 마디를 하나 랜덤하게 정하고, 이 마디의 유망한 자식마디의 개수를 m_2라고 한다.
4. …
5. 더 이상 유망한 자식마디가 없을 때까지 이 과정을 반복한다.

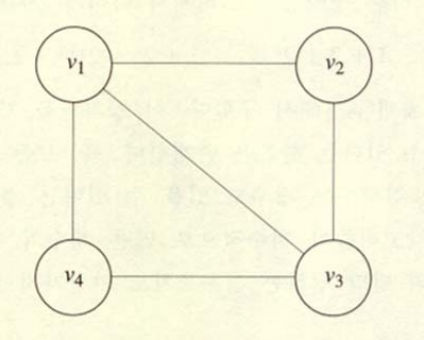
여기서 m_i는 수준 i에 있는 마디의 유망한 자식마디의 개수의 평균의 추정치 이다. 수준 i에 있는 한 마디의 자식마디의 총 개수를 t_i라고 하면 (유망하지 않은 마디도 포함), 되추적 알고리즘에 의해서 점검한 마디의 총 개수의 추정치는

$$1 + t_0 + m_0 t_1 + m_0 t_1 t_2 + \cdots m_0 m_1 \cdots m_{i-1} t_i + \cdots$$

2001-03-19 알고리즘 강의 슬라이드 5 23

그래프 색칠하기(Graph Coloring)

- **지도에 *m*가지 색으로 색칠하는 문제**
  - ✓ *m*개의 색을 가지고, 인접한 지역이 같은 색이 되지 않도록 지도에 색칠하는 문제



이 그래프에서 두가지 색으로 문제를 풀기는 불가능하다.

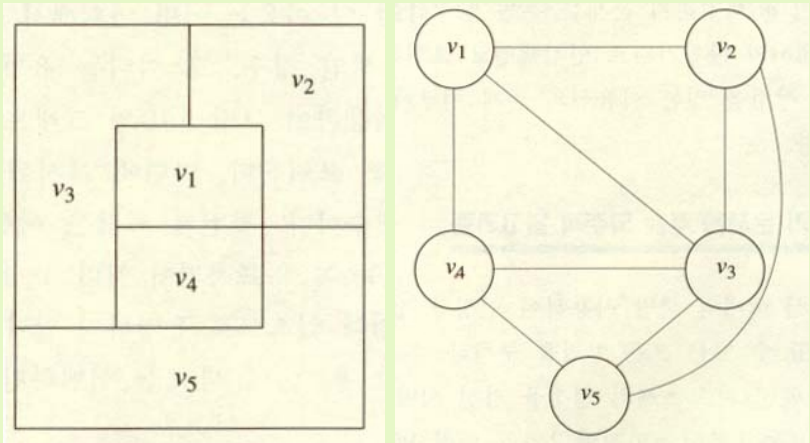
세 가지 색을 사용하면 총 6가지의 해답을 얻을 수 있다.

2001-03-19 알고리즘 강의 슬라이드 5 24

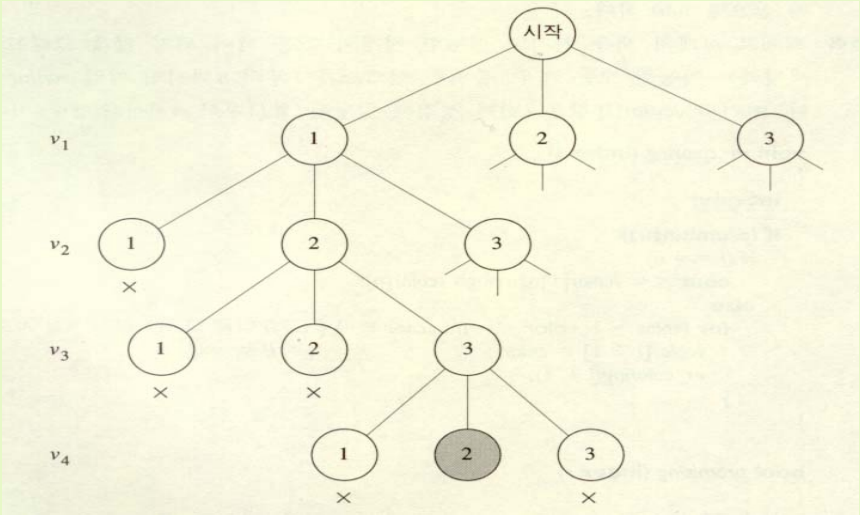
평면그래프(Planar Graph)

- 평면 상에서 이음선(edge)들이 서로 엇갈리지 않게 그릴 수 있는 그래프.
- 지도에서 각 지역을 그래프의 정점으로 하고, 한 지역이 어떤 다른 지역과 인접해 있으면 그 지역들을 나타내는 정점들 사이에 이음선을 그으면, 모든 지도는 그에 상응하는 평면그래프로 표시할 수 있다

지도와 평면그래프



그래프 색칠하기 되추적 해법



그래프 색칠하기: 분석

- 상태공간트리 상의 마디의 총수는

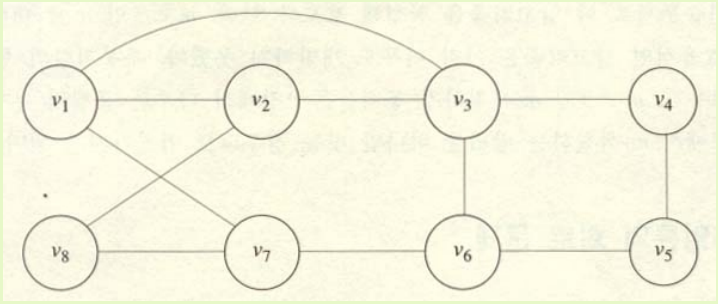
$$1 + m + m^2 + \cdots + m^m = \frac{m^{m+1} - 1}{m - 1}$$

가 된다.

- 여기서도 Monte Carlo 기법을 사용하여 수행시간을 추정할 수 있다.

해밀토니안 회로

- 연결된 비방향성 그래프에서, 해밀토니안 회로(Hamiltonian Circuits) / 일주여행경로(tour)는 어떤 한 마디에서 출발하여 그래프 상의 각 정점을 한번씩 만 경우하여 다시 출발한 정점으로 돌아오는 경로이다.



해밀토니안 회로 문제

- **해밀토니안 회로 문제**: 연결된 비방향성 그래프에서 해밀토니안 회로를 결정하는 문제
- 되추적 방법을 적용하기 위해서 다음 사항을 고려해야 한다.
  - ✓ 경로 상의 i번째 정점은 그 경로 상의 $(i-1)$번째 정점과 반드시 이웃 해야 한다.
  - ✓ $(n-1)$번째 정점은 반드시 0번째 정점(출발점)과 이웃 해야 한다.
  - ✓ i번째 정점은 처음 $i-1$개의 정점이 될 수 없다.
- 상태공간트리 상의 마디 수는

$$1 + (n-1) + (n-1)^2 + \cdots + (n-1)^{(n-1)} = \frac{(n-1)^n - 1}{n-2}$$