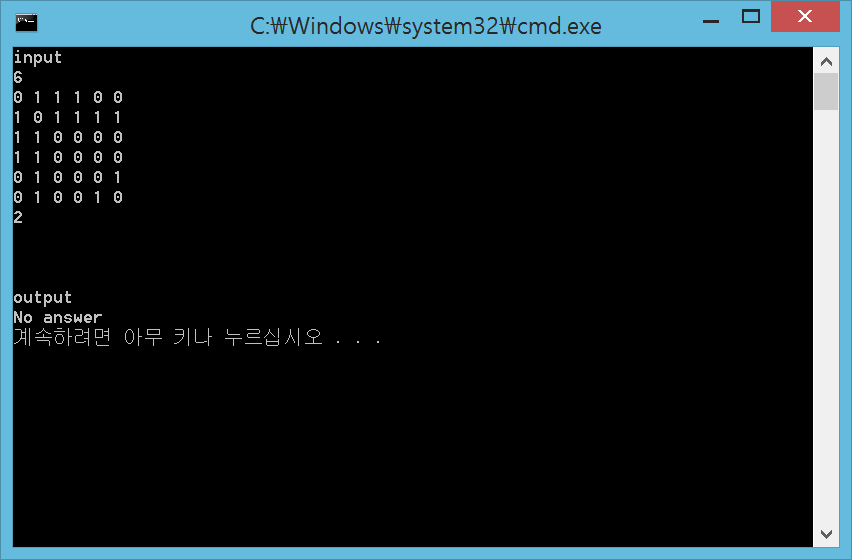
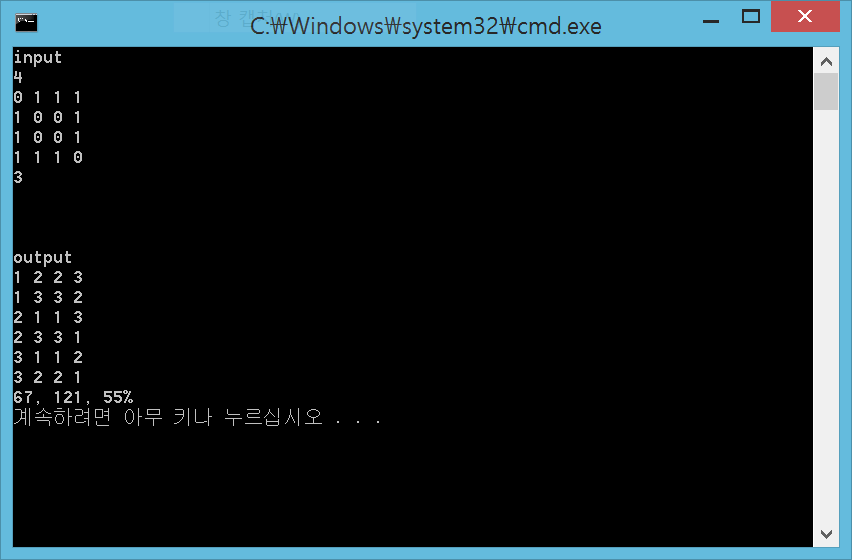
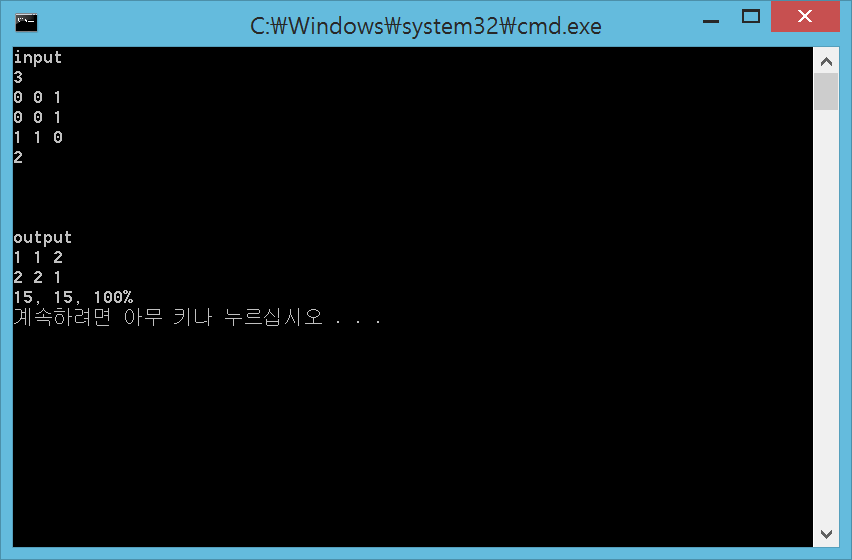
HW#3

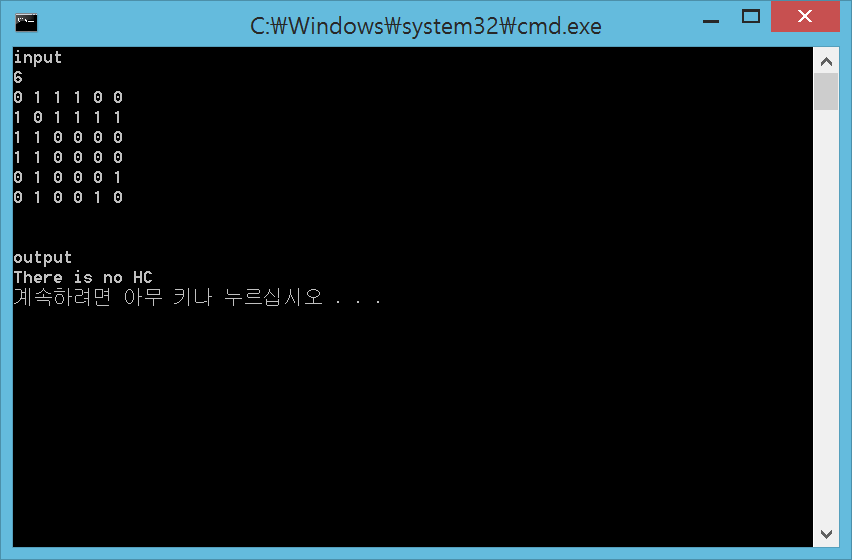
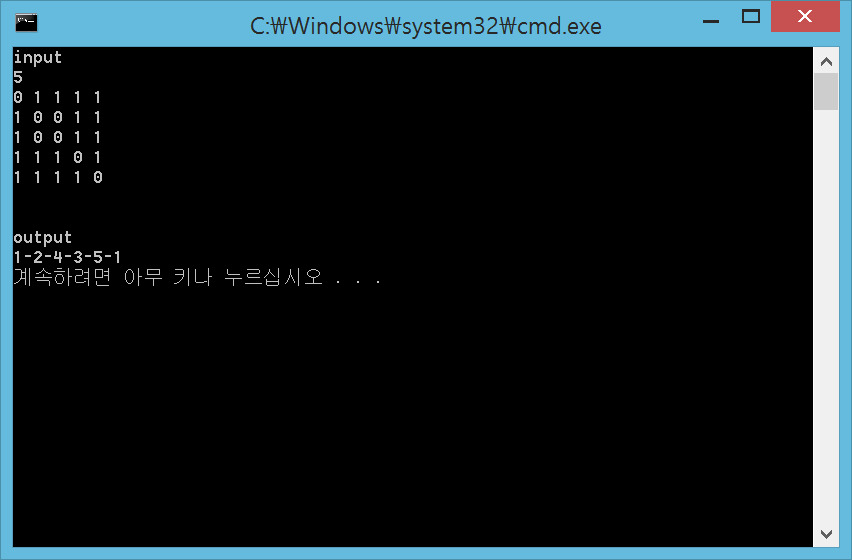
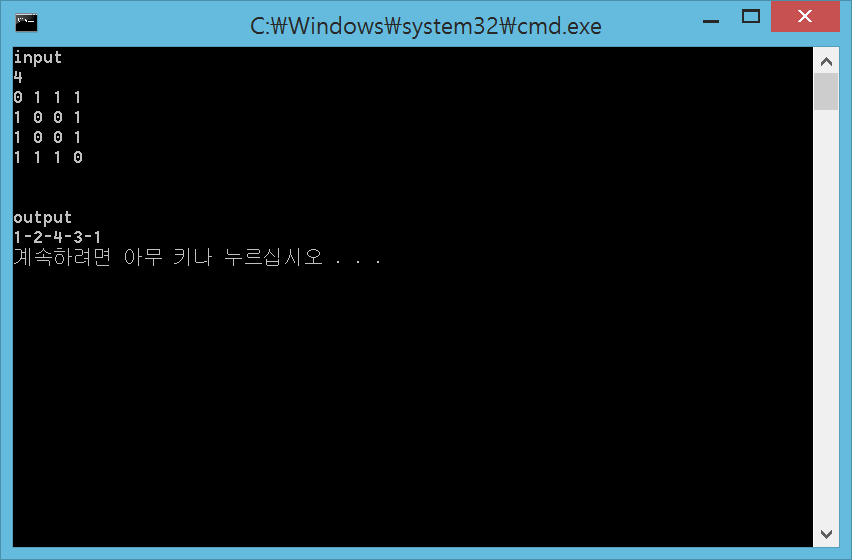
1515655 컴퓨터과학과 임소희

- 결과

<1번>



<2번>



해밀턴 사이클이 아닌 경우 (아래 그래프는 위의 matrix 표현한 것)

**5**

**1**

**2**

**4**

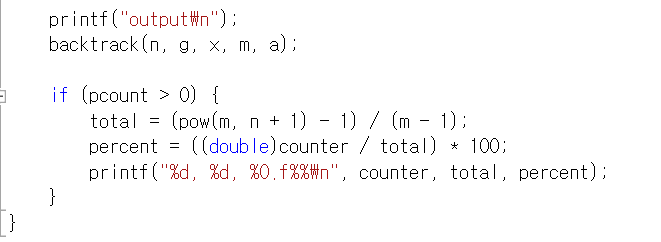
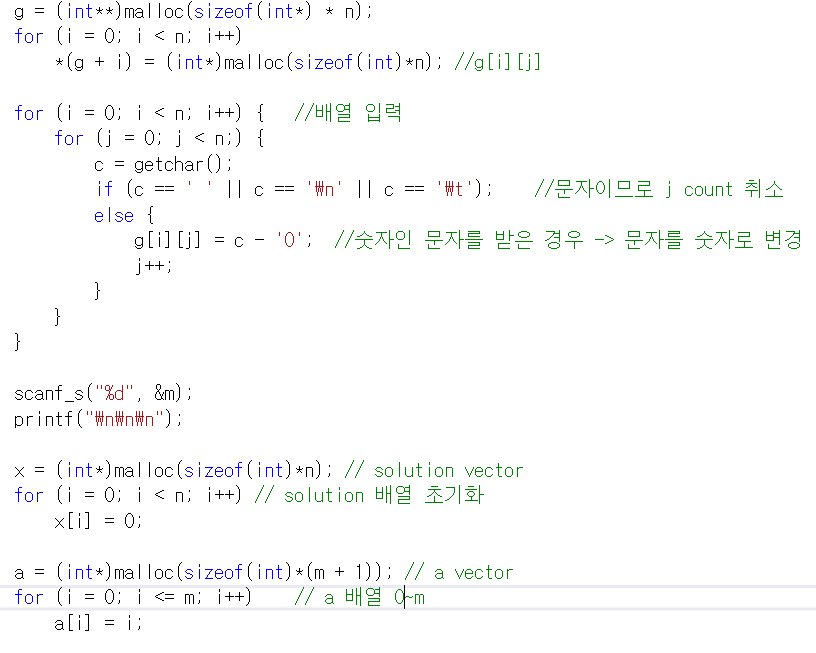
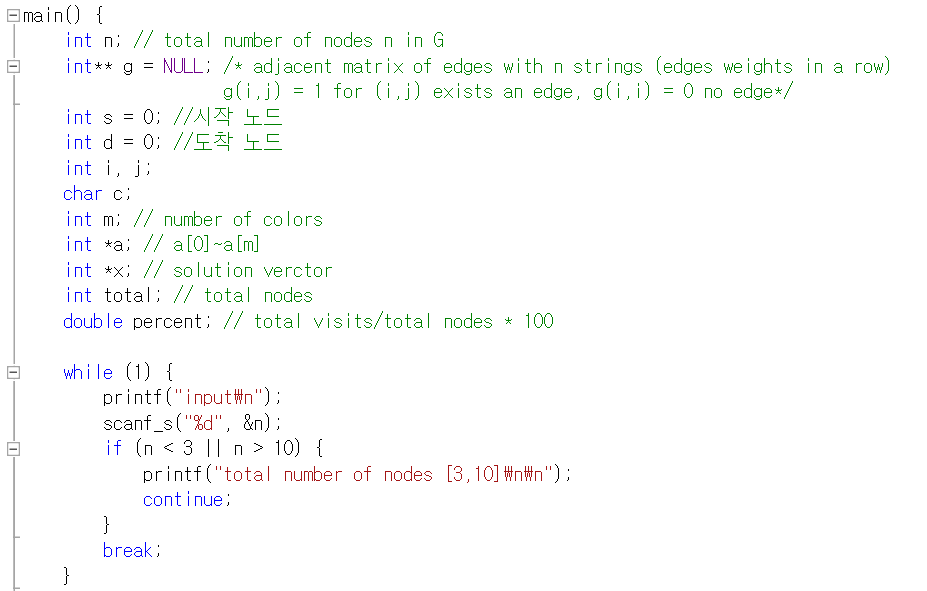
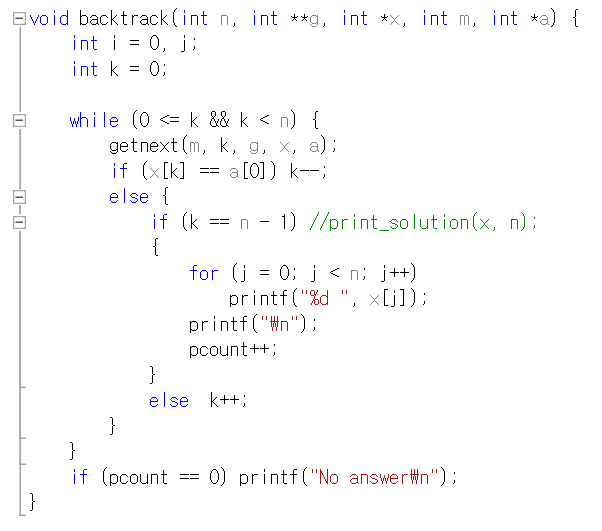
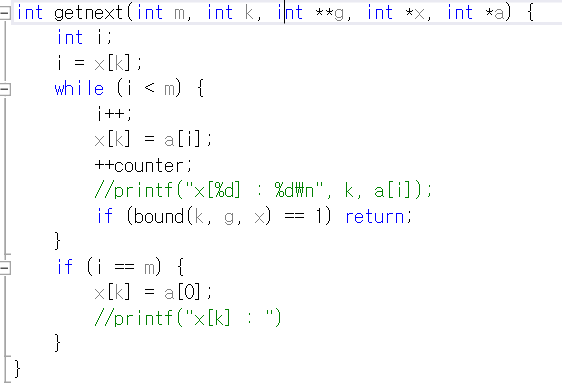
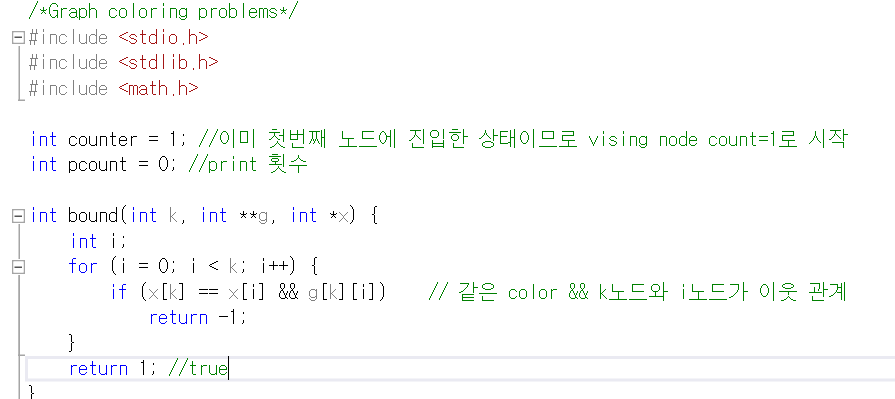
**1**

**6**

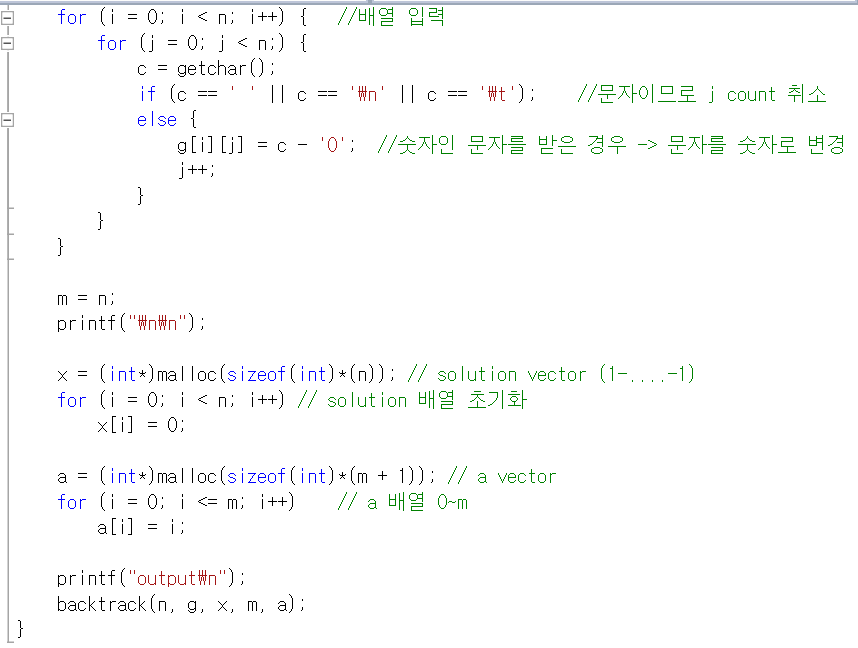
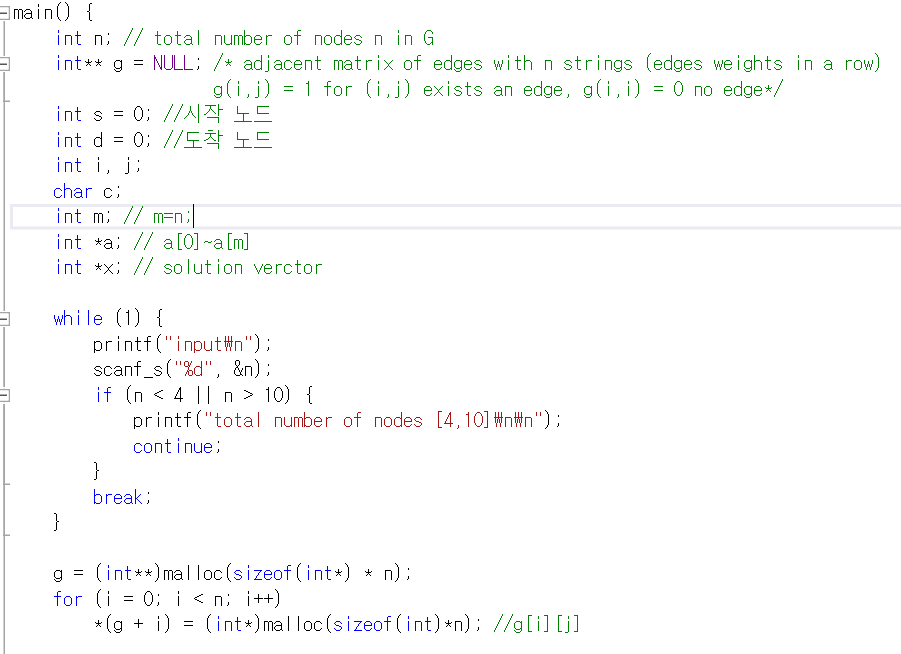
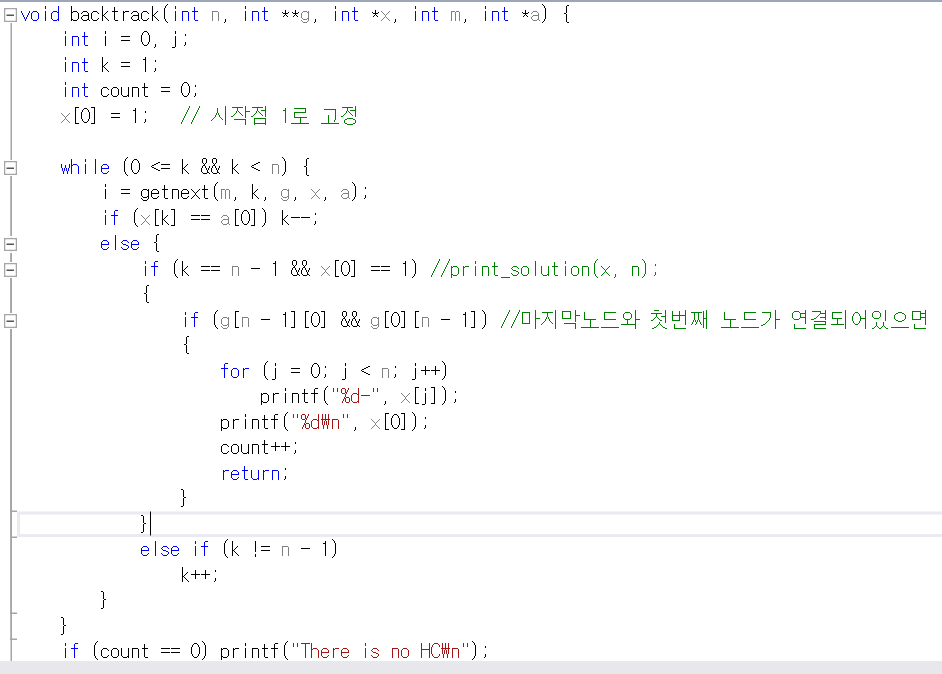
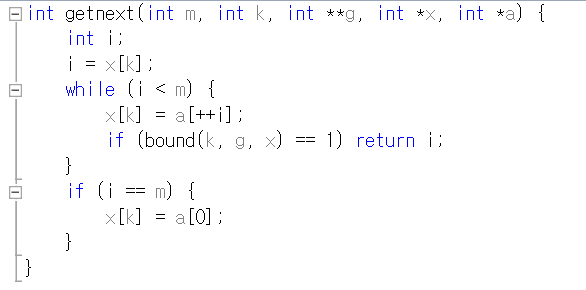
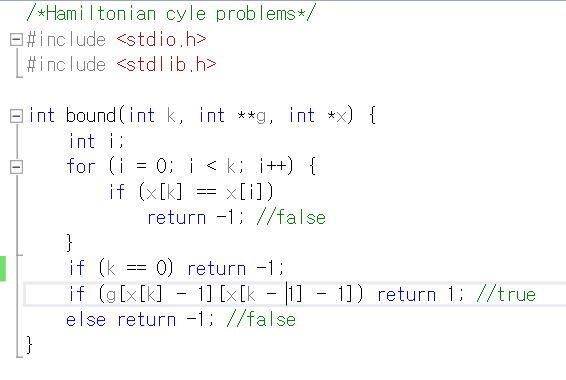
**3**

- 코드

<1번>



<2번>



Graph coloring problems와 Hamiltonian cycle problems를 풀기 위한 코드이다 (0의 경우 no edge이다).

backtracking 알고리즘을 이용하여 greedy 알고리즘이 당시의 선택이 최선이자 최종의 선택인 반면, backtracking 알고리즘은 선택을 번복할 수 있어 결과에 있어 좀 더 만족스러운 답을 얻을 수 있다.

BACKTRACK(n) 🡪 GETNEXT(k) 🡪 Bound(k) 순으로 함수를 호출하여 답을 찾아낸다.

BACKTRACK(n)으로 노드 탐색을 시작하여 GETNEXT(k)로 동일 레벨의 노드를 이동하며 값(당시 최선의 답)을 구한다. 최선의 답이란 초기화와 답이 될 수 있는 값들을 모아 놓은 set(a set이라 칭하자) 내의 값(0~m)을 정답set에 넣어주는 것이다. m보다 작은 경우는 현재 가진 값에 1을 증가시키고, m인 경우는 더 이상 증가시킬 수 없으므로 0으로 초기화한다.

m보다 작은 경우일 때 Bound를 수행한다. Bound(k)로 특정 문제(문제 마다 bound 구성이 다름)를 해결하는 조건을 만족하는지 판별하여 T/F를 보내고, F이면 answer state가 이 노드의 subtree에 없으므로 다른 노드(sibling node)를 살펴본다는 의미이다. T이면 answer state가 이 노드의 subtree 내에 있다는 의미이며, GETNEXT로부터 BACKTRACK으로 돌아온다.

현재 보는 노드가 초기화 되었으면 이전 노드(부모 노드)로 돌아가서 그 노드부터 선택을 다시 한다(backtrack). 현재 노드가 마지막 노드이면 노드에 담긴 답들을 출력하고(leaf node), 노드를 아직 덜 둘러보았다면 자식 노드로 이동한다(child level 증가).

정답을 print해도 skip을 비롯한, 아직 경로 탐색을 다 한 상태가 아니라면 더 최선의 답을 위해 남은 경로를 둘러볼 필요가 있다. 마지막 노드(경로 표현하는 트리가 아닌 실제 그래프 내의 마지막 노드)를 본 상태라도 GETNEXT 함수를 수행할 수 있다.

종료를 하는 조건은 트리로 따지면 root노드보다 더 상위레벨로 backtrack하는 것이다. GETNEXT에서 현재 노드가 a set의 마지막 값이라면 초기화 시키고, BACKTRACK으로 돌아가 backtrack하고 이 과정을 계속 반복한다. root노드도 초기화 되어 backtrack하면 갈 노드가 없으므로 BACKTRACK(n) 함수가 종료된다. 결국 모든 트리를 탐색하며 (답이 없다고 판단되면 skip) 최선의 답을 찾은 셈이다.

1번 Bound 조건 : 현재 노드가 이웃 노드와 같은 색이 아니면 충족

2번 Bound 조건 : 현재 노드가 유일하며(정답set에 단 1개만 있어야 함), 이전 노드와 연결되어 있으면 충족