

2017156019 염민규 알고리즘HW9

과제를 진행한 인원 : 1명(Alone)

개요

NP-Complete를 알아보기 전에, P에 대해 알아보자. P(polynomial time)란 다항시간에 결정론적으로 해결 가능한 문제들의 집합이다. 다시 말해, 어떤 결정 문제 X에 대해 주어지는 입력이 참인지 거짓인지를 다항시간에 결정론적으로 해결하는 알고리즘이 존재하면 X는 P에 속한다고 할 수 있다.

그렇다면 NP란 무엇인가? NP(non-deterministic polynomial time)는 다항시간에 비결정론적으로 해결 가능한 문제들의 집합으로, 어떤 결정 문제 X에 대해 주어지는 입력이 참인지 거짓인지를 다항시간에 비결정론적으로 해결하는 알고리즘이 존재하면 X는 NP에 속한다고 할 수 있다.

또한 NP의 또 다른 정의는 다항 시간에 결정론적으로 검증 가능한 문제들의 집합이다. 어떤 결정 문제 X에 대해 어떤 입력과 입력 크기의 어떤 다항식으로 표현되는 크기를 갖는 증거를 함께 제공 받았을 때, 이를 다항시간에 결정론적으로 검증하는 알고리즘이 존재하면 X는 NP에 속한다고 할 수 있다.

(출처 : <https://gazelle-and-cs.tistory.com/74>)

과제를 늦게 시작해 예습느낌으로 진행했어야 할 과제였지만, 일정때문에 미루고 미뤄 복습의 성격을 띄게 되어버렸다. 이왕 이렇게 된거, 오늘 배운 내용 되새긴다는 느낌으로 과제를 수행 하여야겠다.

오늘 배웠던 NP-완전 문제들에 더해 또 어떤 문제들이 있는지 찾아보도록 하겠다.

NP-완전문제

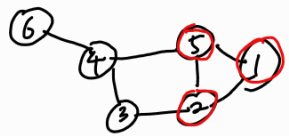
1. Clique problem(클릭 문제)

클릭 문제는 NP-완전인 그래프 이론에 등장하는 문제이다. 그러면서 오늘 따끈따끈하게 배운 내용이기도 하다. 그래프의 클릭(Clique)이란 부분그래프이면서 그래프의 임의의 두 노드가 서로 연결된 것으로 정의된다. 즉, 완전 그래프인 부분그래프를 클릭이라 한다.

input: 그래프 G (노드 V , 간선 E)
부분그래프 크기 r

Question: 그래프 G 에 크기 r 인 완전 부분 그래프 (클릭)인 K_r 이 존재하는가?

output: Yes / No

ex) $r=3$, $G =$  \Rightarrow 노드 1, 2, 3 : 클릭

(출처 : https://ko.wikipedia.org/wiki/%ED%81%B4%EB%A6%AD_%EB%AC%B8%EC%A0%9C)

(출처 : https://ko.wikipedia.org/wiki/%ED%81%B4%EB%A6%AD_%EB%AC%B8%EC%A0%9C)

2. Satisfiability(SAT, 만족 문제)

만족 문제 또한 오늘 배운 내용이다. 이는 주어진 부울 공식을 만족하는 해석이 존재하는지 확인하는 문제이다.

input : 논리식, 변수 갯수
 N -SAT 이면 리터럴 개수 N 이하의 포함

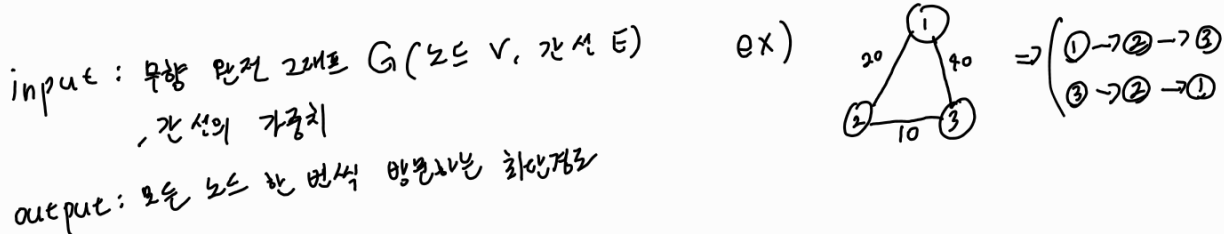
output: Yes / No

(출처 : <https://www.geeksforgeeks.org/proof-that-sat-is-np-complete/>)

3. TSP(Traveling Salesman Problem, 외판원 문제)

교수님께서 예시로 올려주신 문제이다. 이는 내가 유일하게 수업 이전에 알고있던 문제이기도 하다. 외판원이 있다고 가정하겠다. 이 외판원이 어느 도시를 방문했는데 이 도시에는 여러 집이 있다. 이 집을 전부 한 번씩만 방문해야 하는데, 이 외판원은 빨리 일을 끝내고 퇴근하고 싶어한다. 이 때, 외판원이 모든 집을 각각 한 번씩만 방문하고(방문했던 곳은 다시 방문하지 않는다.) 이때 가장 짧은 경로를 구하는 것이 이 문제의 요지이다.

조금 더 찾아보니 외판원 문제는 NP-난해라는 것이 증명되었다고 한다. 그럼 이 문제는 NP-완전가 아닌 것인가? 외판원 문제를 결정문제 X값이 주어졌을 때 X보다 비용이 적게 드는 회로가 있는가?로 변환하면 NP-완전이 된다고 한다.



외판원 문제를 실생활에 응용한 사례도 찾아볼 수 있었다. 이 문제를 다른데도 아니고 '원전'에 접목했더니 효율이 40%나 올랐다는 사례였다. 요약하면 원전에서 이동해야 할 삽입체의 개수는 33개, 삽입체가 장착돼야 하는 연료도 33개, 기중기는 이 삽입체를 놓은 뒤 다시 제자리로 돌아와야 하는데, 모든 지점을 들렀다가 원점으로 돌아온다는 외판원 문제와 닮아있었다. 이 때 발생할 수 있는 경우의 수는 $33!$ 로 상상할 수도 없는 큰 수가 나오는데, 외판원 문제를 접목시키니 최소거리가 나오는 이동경로를 구해 적용한 결과 원전 1기당 이동시간이 10분정도 감소하였다고 한다. 이를 비용으로 따지면 수천만 원에 달하는 수준이라니, 실생활에 접목된 NP-완전문제가 이렇게 활용될 수도 있다는 게 신기할 따름이다.

외람된 이야기지만, 친구들끼리 진행하는 알고리즘 스터디에서 이번주에 서로 풀어오도록 제시된 문제가 외판원 문제이다. 요근래 엄청 많이 접하는 것 같아 신기하다. 외판원 문제에 관한 코딩테스트를 하면서 이 문제에 대해 더 깊게 이해할 수 있으면 좋겠다.

응용 문제

외판원 순회

★

1 골드 I

시간 제한	메모리 제한	제출	정답	받은 사항	정답 비율
1 초	128 MB	37925	9728	5847	27.993%

문제

외판원 순회 문제는 영어로 Traveling Salesman problem (TSP) 라고 불리는 문제로 computer science 분야에서 가장 중요하게 취급되는 문제 중 하나이다. 여러 가지 변종 문제가 있으나, 여기서는 가장 일반적인 형태의 문제를 살펴보자.

1번부터 N번까지 번호가 매겨져 있는 도시들이 있고, 도시들 사이에는 길이 있다. (길이 없을 수도 있다) 이제 한 외판원이 어느 한 도시에서 출발해 N개의 도시를 모두 거쳐 다시 원래의 도시로 돌아오는 순회 여행 경로를 계획하려고 한다. 단, 한 번 갔던 도시로는 다시 갈 수 없다. (맨 마지막에 여행을 출발했던 도시로 돌아오는 것은 예외) 이런 여행 경로는 여러 가지가 있을 수 있는데, 가장 적은 비용을 들이는 여행 계획을 세우고자 한다.

각 도시간에 이동하는데 드는 비용은 행렬 $W[i][j]$ 형태로 주어진다. $W[i][j]$ 는 도시 i에서 도시 j로 가기 위한 비용을 나타낸다. 비용은 대칭적이지 않다. 즉, $W[i][j]$ 는 $W[j][i]$ 와 다를 수 있다. 모든 도시간의 비용은 양의 정수이다. $W[i][i]$ 는 항상 0이다. 경우에 따라서 도시 i에서 도시 j로 갈 수 없는 경우도 있으며 그럴 경우 $W[i][j]=0$ 이라고 하자.

N과 비용 행렬이 주어졌을 때, 가장 적은 비용을 들이는 외판원의 순회 여행 경로를 구하는 프로그램을 작성하시오.

입력

첫째 줄에 도시의 수 N이 주어진다. ($2 \leq N \leq 16$) 다음 N개의 줄에는 비용 행렬이 주어진다. 각 행렬의 성분은 1,000,000 이하의 양의 정수이며, 갈 수 없는 경우는 0이 주어진다. $W[i][i]$ 는 도시 i에서 j로 가기 위한 비용을 나타낸다.

항상 순회할 수 있는 경우만 입력으로 주어진다.

출력

첫째 줄에 외판원의 순회에 필요한 최소 비용을 출력한다.

예제 입력 1 복사

예제 출력 1 복사

```
4
0 10 15 20
5 0 9 10
6 13 0 12
8 8 9 0
```

```
35
```

(출처: <https://zeddios.tistory.com/176>)

(출처: https://ko.wikipedia.org/wiki/%EC%99%B8%ED%8C%90%EC%9B%90_%EB%AC%B8%EC%A0%9C)

(사례: <https://www.mk.co.kr/news/business/9102763>)

(문제: <https://www.acmicpc.net/problem/2098>)

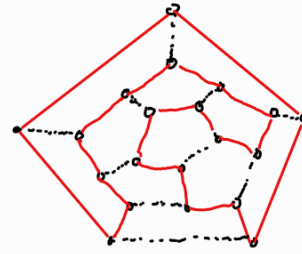
4. Hamiltonian Path(해밀턴 경로)

해밀턴 경로란, 무방향 완전그래프의 모든 정점을 한 번씩 지나는 경로이다. 어? 어디서 본 것 같다. 바로 외판원 문제에서 간선의 가중치를 1로 주었을 때가 해밀턴 경로이다. (추가적으로, 해밀턴 회로는 해밀턴 경로 중, 시작점 = 마지막점일 때를 말한다.)

input : 무방향 완전그래프 G (노드 V , 간선 E)

output : 모든 정점을 한 번씩 지나는 경로

ex)



(출처: <https://torbjorn.tistory.com/240>)

5. 그래프 색칠 문제

그래프의 노드에 색을 칠해야 하는데, 인접한 노드는 서로 색이 달라야한다거나, 간선의 색이 달라야 하는 조건이 붙는 등, 여러가지 제약이 가해진다. 이 문제에서 찾는 것은 여러가지가 있으며, 예는 다음과 같다.

- 모든 노드를 칠하는데 필요한 최소한의 색상의 수
- 최소한의 색상으로 칠하는 경우의 수
- 정해진 수의 색상만을 이용할 때 칠해지지 않는 노드의 수

input : 그래프 G (노드 V , 간선 E), 조건에 따른 필요값

output : 조건에 부합하는 결과

(출처: <https://m.blog.naver.com/PostView.naver?isHttpsRedirect=true&blogId=pyjune&logNo=220786789271>)

6. Independent set(독립집합 문제)

이 문제 또한 코딩테스트에서 봤던 문제인데 NP-완전 문제의 예시로 알려져 있는것을 보고 왠지 모르게 신기했다. 이 문제도 직접 풀어봐야겠다.

응용 문제

트리의 독립집합 문제풀이 ☆

시간 제한	메모리 제한	채점	정답	맞힌 사람	정답 비율
2 초	128 MB	5054	2450	1834	48.442%

문제

그래프 $G(V, E)$ 에서 정점의 부분 집합 S 에 속한 모든 정점쌍이 서로 인접하지 않으면 (정점쌍을 잇는 간선이 없으면) S 를 독립 집합(independent set)이라고 한다. 독립 집합의 크기는 정점에 가중치가 주어지지 않을 경우는 독립 집합에 속한 정점의 수를 말하고, 정점에 가중치가 주어지지 않으면 독립 집합에 속한 정점의 가중치의 합으로 정의한다. 독립 집합이 공집합일 때 그 크기는 0이라고 하자. 크기가 최대인 독립 집합을 최대 독립 집합이라고 한다.

문제는 일반적인 그래프가 아니라 트리(연결되어 있고 사이클이 없는 그래프)와 각 정점의 가중치가 양의 정수로 주어지 있을 때, 최대 독립 집합을 구하는 것이다.

입력

첫째 줄에 트리의 정점의 수 n 이 주어진다. n 은 10,000이하인 양의 정수이다. 1부터 n 사이의 정수가 트리의 정점이라고 가정한다. 둘째 줄에는 n 개의 정수 w_1, w_2, \dots, w_n 이 주어지는데, w_i 는 정점 i 의 가중치이다($1 \leq i \leq n$). 셋째 줄부터 마지막 줄까지는 간선의 리스트가 주어지는데, 한 줄에 하나의 간선을 나타낸다. 간선은 정점의 쌍으로 주어진다. 입력되는 정수들 사이에는 빈 칸이 하나 있다. 가중치들의 값은 10,000을 넘지 않는 자연수이다.

출력

첫째 줄에 최대 독립집합의 크기를 출력한다. 둘째 줄에는 최대 독립집합에 속하는 정점을 오름차순으로 출력한다. 최대 독립 집합이 하나 이상일 경우에는 하나만 출력하면 된다.

예제 입력 1 복사

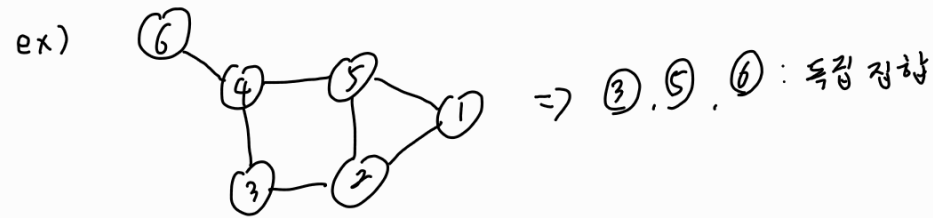
예제 출력 1 복사

```
7
10 30 40 10 20 20 70
1 2
2 3
4 3
4 5
6 2
6 7
```

```
140
1 3 5 7
```

독립집합 문제란, 무향 그래프가 주어졌을 때 서로를 직접 연결하고 있는 모서리가 하나도 없는 꼭지점들의 집합이 존재하는지 구하는 문제이다.

input : 무향 그래프 $G(\text{노드 } V, \text{가선 } E)$, 독립집합 요소의 개수 k
 Output : G 에 원소가 k 개 이상인 독립집합이 존재하면 True, 아니면 False



(출처: <https://kylog.tistory.com/47>)
 (문제: <https://www.acmicpc.net/problem/2213>)

7. 0-1 Knapsack problem(0-1 배낭문제)

배웠었던 냅색과는 조금 다른 0-1 냅색 문제에 대해 알아보고자 한다.

기존 냅색문제와는 다르게, 짐을 쪼갤 수 없으며 한정된 공간을 지닌 배낭에 최대한의 코스트의 짐을 넣어 최대한의 가치를 도출할 수 있는지 알아내는 문제이다.

input : 배낭 공간 S , 짐들의 무게, 코스트
 output : 최대한의 가치를 내는 짐들의 조합

(출처: <https://going-to-end.tistory.com/entry/%EC%B5%9C%EC%A0%81%ED%99%94-%EC%95%8C%EA%B3%A0%EB%A6%AC%EC%A6%98>)

8. Subset sum problem(부분집합 문제)

유한 개의 정수로 이루어진 집합이 있을 때, 이 집합의 부분집합 중에서 그 집합의 원소를 다 더한 값이 0이 되는 경우가 있는지를 알아내는 문제이다.

input : 집합 S
 output : Yes/No

ex) $S = \{-5, -3, 1, 2, 8\}$
 $\therefore -3 + 1 + 2 = 0$
 $\Rightarrow \text{Yes (True)}$

(출처: <https://blog.naver.com/dltjdrif12/50149117964>)

9. Set cover(집합덮개 문제)

집합덮개 문제는 전산학과 복잡도 이론에서 다루는 오래된 문제로, 어떠한 전체집합과 그 집합의 부분집합들이 주어졌을 때, 부분집합들 중에서 가능한 한 적은 집합을 골라 그 집합들의 합집합이 원래 집합이 되도록, 즉 그 집합들이 원래 전체집합을 덮도록 집합을 선택하는 문제이다. 이 때 집합을 가능한 한 적게 골라내는 것이 목표이다. 이 문제의 결정문제는 수업시간에 들었던 리처드 카프가 NP-완전임을 증명했던 최초의 21문제중 하나라고 한다.

input : 전체집합 U , 부분집합의 모임 S

output : 가능한 한 적은 부분집합의 조합으로 U 만드는 case

ex) 결정문제 : (U, S) 쌍과 정수 k 가 입력될 때, k 개 이하인 집합덮개가 존재하는가?

10. Vertex cover problem(정점커버 문제)

각 간선의 양 끝점들 중에서 적어도 하나의 끝점을 포함하는 점들의 집합들 중에서 최소 크기의 집합을 찾는 문제이다. 극대 매칭(Maximal Matching) 방법을 이용하여 문제를 해결할 수 있다고 한다.

극대 매칭 방법 : 한 간선을 선택한 다음, 해당 간선의 양 끝점과 연결된 간선을 모두 제거한다. 그 이후, 간선의 양 끝점이 이미 커버된 간선의 끝점이 아닐 때에만 선택한다.

input : 무향 그래프 G (노드 V , 간선 E)
자연수 k

output : G 에 최소 k 개 이하의 정점커버가 존재하면 True
or 아니면 False

