

실습 1

- 팰린드롬 (palindrom)
 - 앞에서부터 읽으나 뒤에서부터 읽으나 같게 읽히는 문자열
 - 입력 : 문자열 S
 - 출력 : S에 문자열을 추가해서 팰린드롬으로 만들 수 있는 가장 짧은 팰린드롬의 길이

예제 입력	예제 출력
ABAB	5
ABABAA	10
ABCADSABDS	19

문자열을 입력해주세요 : ababaa
팰린드롬에 필요한 문자의 갯수는 10개 입니다.

실습 1

- 주어진 문제를 Brute-Force 알고리즘을 이용해, 출력을 구하는 palindrom_bf 함수를 작성해 구하시오. 시간 복잡도 $O(n^2)$

실습 2

- 영업사원 A는 판매를 위해 N개 업체와 미팅을 해야하는데, 각 회의의 시작 시간과 회의 종료 시간이 입력에 주어졌을 때, A영업사원이 최대한 많은 업체와 미팅을 하려고 한다. 이때, 진행할 회의 각각의 시작시간, 종료시간을 출력하시오.
- Hint : Greedy 알고리즘을 이용해, 문제를 풉니다.
 - 가장 많은 회의를 진행하기 위해서는 회의 종료 시간이 빨리 종료되어야 함. (Sort)
 - Greedy 알고리즘은 항상 같은 답을 출력하지는 못함. 다만, 각각의 에지에 가중치가 있을 때에는, 위상정렬 알고리즘을 이용할 수 있음.
Ex) 입력을 시작시간, 종료시간 (회의에 드는 노력) 을 받는다고 하면, greedy로는 정답을 낼 수 없는 경우가 있음.
 - 이런 가중치를 포함해 계산하고 싶을 때에는 위상정렬을 이용할 수 있음.

예제 입력	greedy 정답	위상 정렬
4 1 2 (10) 1 3 (1) 2 4 (100) 3 4 (10)	1 2 2 4 (110)	1 3 3 4 (11)

실습 2

예제 입력	예제 출력
2 0 2 2 2	0 2 2 2
11 1 4 3 5 0 6 5 7 3 8 5 9 6 10 8 11 8 12 2 13 12 14	1 4 5 7 8 11 12 14

실습 3

- 연속된 수열에서 가장 큰 합 구하기

- N개의 정수로 이루어진 임의의 수열이 주어진다. 이 중 연속된 몇 개의 수를 선택해서 구할 수 있는 합 중 가장 큰 합을 구하려 한다.
- 입력 : N
N개의 정수
- 출력 : 연속된 부분의 가장 큰 합

예제 입력	예제 출력
10 2 1 -4 3 4 -4 6 5 -5 1	14
5 -1 -2 -3 -4 -5	-1
10 10 -4 3 1 5 6 -35 12 21 -1	33

N : 10
수열을 입력해주세요 :
2 1 -4 3 4 -4 6 5 -5 1
정답은 14입니다.

실습 4

실습 4번 문제는 실습 파일 제출 후, 코드 실행 시간에 따른 추가적인 평가가 있을 예정입니다

- 나라 방문

1부터 N까지 번호가 매겨져 있는 나라들이 있고, 나라들 사이에는 길이 있다. 이제 한 대학생이 어느 한 나라에서 출발해 N개의 나라를 모두 거쳐 원래의 나라로 돌아오는 순회 여행을 계획하려 한다. 단, 한 번 갔던 나라로는 다시 갈 수 없다. 이때, 각 나라 간의 입국심사에 쓰이는 노력(비용)의 양이 다르다. (이는 대칭적이지 않음)

이때, 가장 적은 노력(비용)을 들여 순회 여행을 하려 한다. 가장 적은 비용을 출력해라. (수행시간 제한이 있음. 제한 시간 안에 완료가 돼야 함.)

입력 : N, N x N 의 노력배열

출력 : 순회 여행을 할 수 있는 최소 노력

예제 입력	예제 출력
4 0 10 15 20 5 0 9 10 6 13 0 12 8 8 9 0	35

N : 4
각 나라별 필요 노력을 입력해주세요
0 10 15 20
5 0 9 10
6 13 0 12
8 8 9 0
정답은 35 입니다.

실습 4

- Hint :

- DFS, DP를 이용해 문제를 풉니다.
- DFS로 각 여행지를 체크하는데, DP를 이용해 노력의 양을 저장합니다. (다익스트라 기법과 비슷합니다.)
- DP 에는 2차원 배열로 DP[현재위치][방문할위치] 의 노력의 최소값을 저장합니다.
- 정답이 정상적으로 나오지만, 시간이 오래 걸린다면, DFS에서 방문조건을 비트마스킹 기법을 이용해 나타냅니다.

- 다익스트라 (Dijkstra) 알고리즘

- 현재까지 알고 있던 최단 경로를 계속해서 갱신하는 알고리즘
- 문제4 에서는 DFS를 이용해야 하지만, 다익스트라는 DFS는 아님

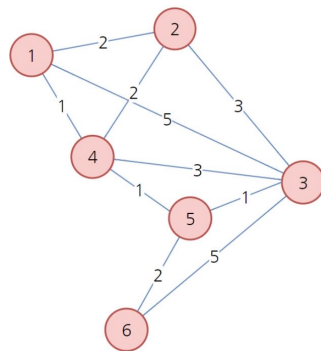
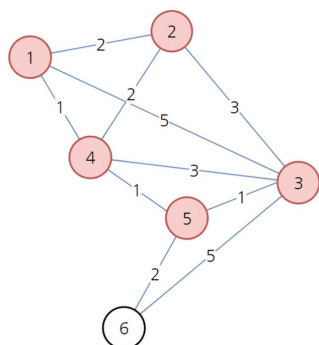
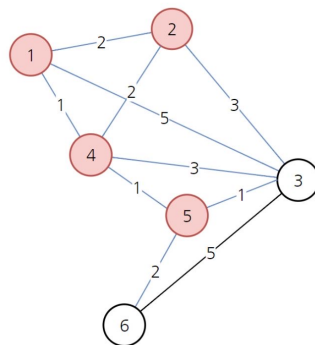
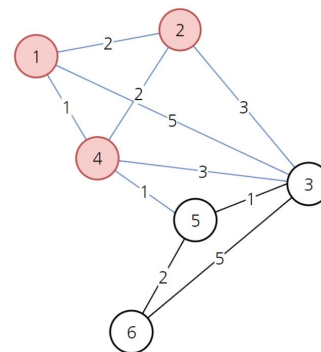
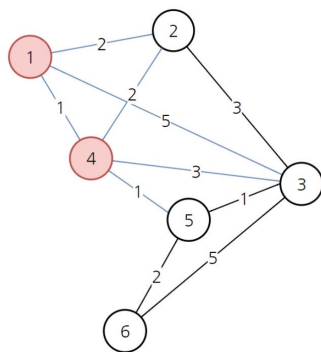
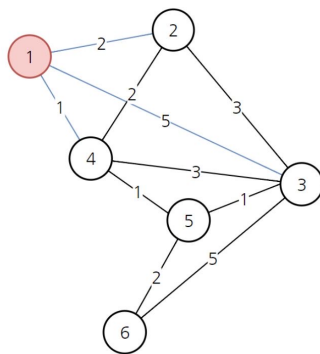
0	2	5	1	무한	무한
2	0	3	2	무한	무한
5	3	0	3	1	5
1	2	3	0	1	무한
무한	무한	5	1	0	2
무한	무한	5	무한	2	0

실습 4

1. 시작 노드 지정
2. 시작 노드에서 갈 수 있는 노드의 비용을 저장
3. 방문하지 않은 노드 중 가장 작은 비용의 노드를 선택
4. 3에서 포함된 노드를 경유하는 경우를 고려하여 비용 갱신
5. 모든 노드를 방문할 때까지 3-4 반복

• Hint : 다익스트라 (Dijkstra) 알고리즘

• 현재까지 알고 있던 최단 경로를 계속해서 갱신하는 알고리즘



0	2	5	1	무한	무한
0	2	4	1	2	무한
0	2	4	1	2	무한
0	2	3	1	2	4
0	2	3	1	2	4
0	2	3	1	2	4

실습 4, Hint : Bit-Masking

- 정수의 이진 표현을 자료구조로 쓰는 기법
- 더 빠른 수행 시간, 더 간결한 코드, 더 적은 메모리 사용
- 집합을 쉽게 표현할 수 있음

실습 4, Hint : Bit-Masking

비트 연산자

$A \& B$	A의 모든 비트와 B의 모든 비트를 AND 연산한다. 둘 다 1이라면 1 아니면 0	$A = 100(2), B = 111(2)$ $A \& B = 100(2)$
$A B$	A의 모든 비트와 B의 모든 비트를 OR 연산한다. 둘 다 0이라면 0, 아니면 1	$A = 010(2), B = 101(2)$ $A B = 111(2)$
$A \wedge B$	A의 모든 비트와 B의 모든 비트를 XOR 연산한다. 둘이 다르다면 1, 아니면 0	$A = 011(2), B = 101(2)$ $A \wedge B = 110(2)$
$\sim A$	A의 모든 비트에 NOT 연산한다 0이면 1, 1이면 0	$A = 011(2)$ $\sim A = 100(2)$
$A \ll B$	A를 B비트만큼 왼쪽으로 시프트	$A = 001(2)$ $A \ll 2 = 100(2)$
$A \gg B$	A를 B비트만큼 오른쪽으로 시프트	$A = 100(2)$ $A \gg 2 = 001(2)$

실습 4, Hint : Bit-Masking

집합의 표현

- 각각의 원소를 0부터 N-1까지 번호를 부여하고, 번호에 해당하는 비트가 1이면 포함, 0이면 불포함

Ex) $A = \{1, 2, 3, 4, 5, 6\}$ 일때,

$$A = 111111_{(2)}, \{2, 4\} = 010100_{(2)}, \{1, 2, 6\} = 110001_{(2)}$$

이런 특성으로 합집합, 교집합, 차집합을 표현할 수 있음

1. 합집합 : $A \mid B$
2. 교집합 : $A \& B$
3. 차집합 : $A \& \sim B$

실습 4, Hint : Bit-Masking 예시

```
switch_states = [True, False, False, True, True, False, False, False, True, True]

# 집합을 이진으로 표현함
switch_states_with_bit = 0b1001100011

# 인덱스 2를 True로 바꿈
n = len(switch_states) - 1 - 2
print(bin(switch_states_with_bit | (1 << n)))

# 인덱스 3을 False로 바꿈
n = len(switch_states) - 1 - 3
print(bin(switch_states_with_bit & ~(1 << n)))

# 원소 토글 (켜져있으면 끄고 꺼져있으면 켜기)
n = len(switch_states) - 1 - 6
print(bin(switch_states_with_bit ^ (1 << n)))
```

```
0b1011100011
0b1000100011
0b1001101011
```

과제

- 실습 30이 Greedy algorithm과 Dynamic programming 중 어디에 해당된다고 할 수 있는가? 이유를 설명하시오.
- 앞에서 다루지 않은 문제 중에 Dynamic programming을 사용했을 때 더 효율적으로 풀 수 있는 문제를 2개 들고, 알고리즘을 설명하시오. brute force 방식에 비하여 DP를 사용하는 경우 어느 정도 빨라지는지도 설명하시오.