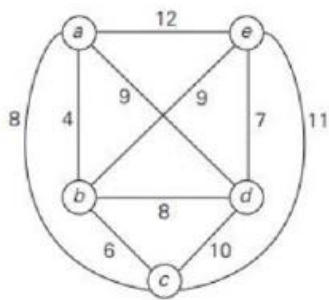


과제#3 - TSP 문제의 근사알고리즘 "ApproxTSP1"

- ① 그래프 G 에 대해 알고리즘 $\text{ApproxTSP1}(G)$ 를 이용하여 근사해를 구하는 과정

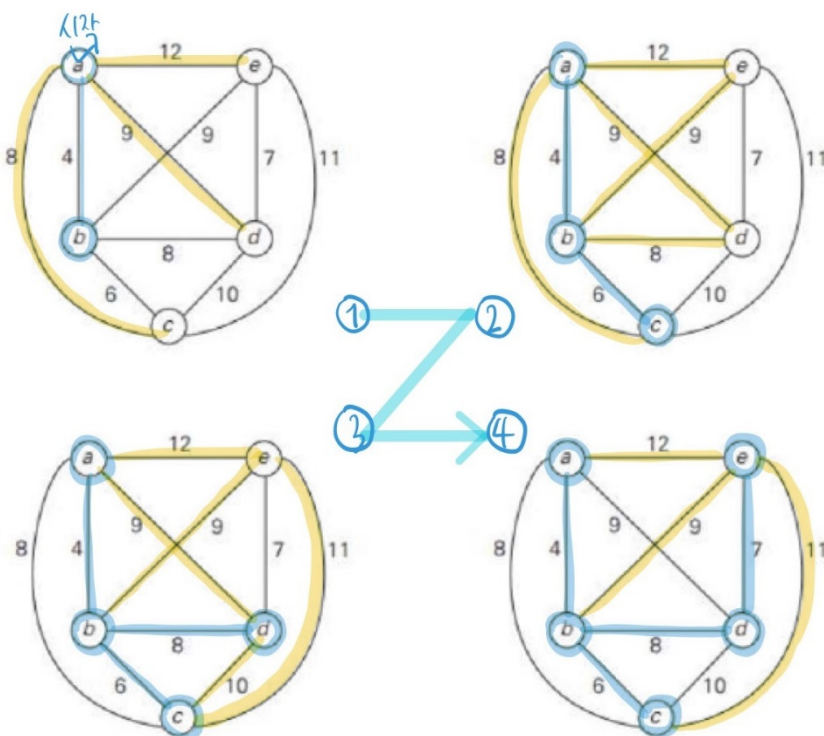
입력



알고리즘 $\text{ApproxTSP1}(G)$

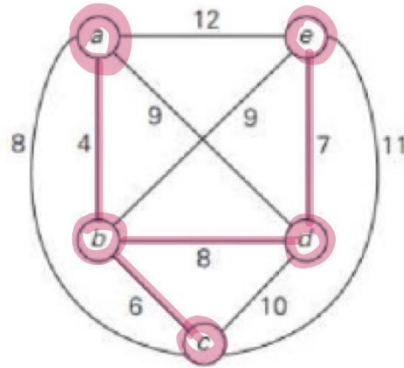
1단계) 프림 알고리즘이나 크루스칼 알고리즘을 사용해서 최소 스패닝 트리 T 를 구한다.

다음은 프림 알고리즘을 이용하여 최소 스패닝 트리 T 를 구하는 과정이다.

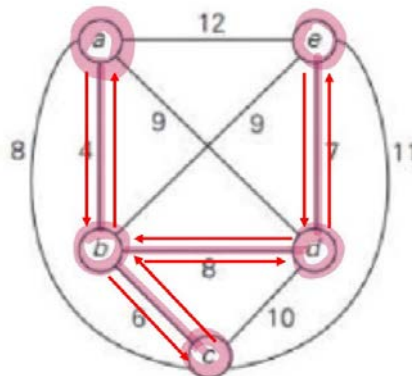


정점 a 를 시작점으로 하여 각 과정이 $1 \rightarrow 2 \rightarrow 3 \rightarrow 4$ 로 진행됨에 따라, 트리에 정점 a, b, c, d, e 를 차례로 추가하게 된다. 연결 가능한 간선은 노란색으로 표시하였고, 그 중에서 가장 작은 가중치를 가져 선택된 노드와 엣지는 파란색으로 표시하였다.

따라서 프림 알고리즘을 이용하여 구한 최소 스패닝 트리 T 는 다음과 같다.



2단계) T 에서 임의의 정점 a 를 시작점으로 선택하였고, a 가 루트인 T 를 중순위로 운행한다. 중순위로 운행하면서 방문하는 정점들을 경로 Γ' 에 추가한다. 여기서는 Γ' 가 사이클이 되도록 a 를 마지막에 추가하였다.



경로 Γ' 를 순서대로 나열하면 다음과 같다.

$a \ b \ c \ b \ d \ e \ d \ b \ a$

3단계) Γ' 에서 각 정점이 처음으로 나오는 경로만 남겨두고 그 외에는 모두 제거한 경로를 Γ 라한다.

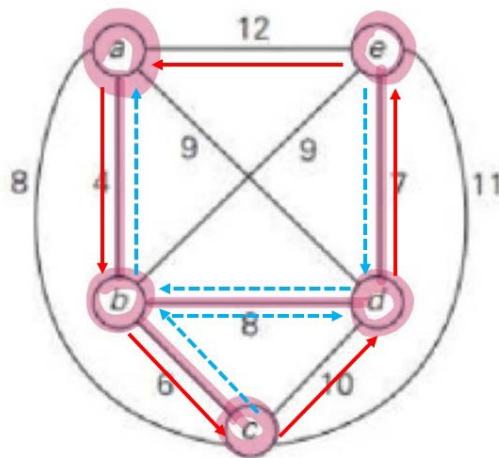
Γ 의 경로는 다음과 같다.

$a \ b \ c \ b \ d \ e \ a$

출력

해당 알고리즘을 통해 구한 근사해 Γ 은 $a \rightarrow b \rightarrow c \rightarrow d \rightarrow e$ 의 경로 (빨간색 화살표로 표시)를 가지

게 된다.



따라서, 그래프 G 에 대하여 알고리즘 $\text{ApproxTSP1}(G)$ 의 결과로 얻은 사이클에 있는 가중치의 합 $w(\Gamma)$ 는 $4+6+10+7+12 = 39$ 이다.

② 알고리즘 ApproxTSP1 이 2-근사 알고리즘임을 확인

#TSP 문제를 완전탐색을 이용하여 해결하는 방법

```
def calcPath(currentdist) :  
  
    if len(path) == n :  
        return currentdist + weight[path[len(path)-1]][path[0]]  
  
    shortestdist = float('inf')  
  
    for next in range (n) :  
  
        if visited [next] :  
            continue  
  
        currentindex = path[len(path)-1]  
        path.append(next)  
        visited[next] = True  
        dist = calcPath(currentdist + weight[currentindex][next])  
        shortestdist= min (shortestdist, dist)  
  
        visited[next] = False  
        path.remove(next)  
  
    return shortestdist
```

$n = \text{int}(\text{input}())$ #노드의 수 입력

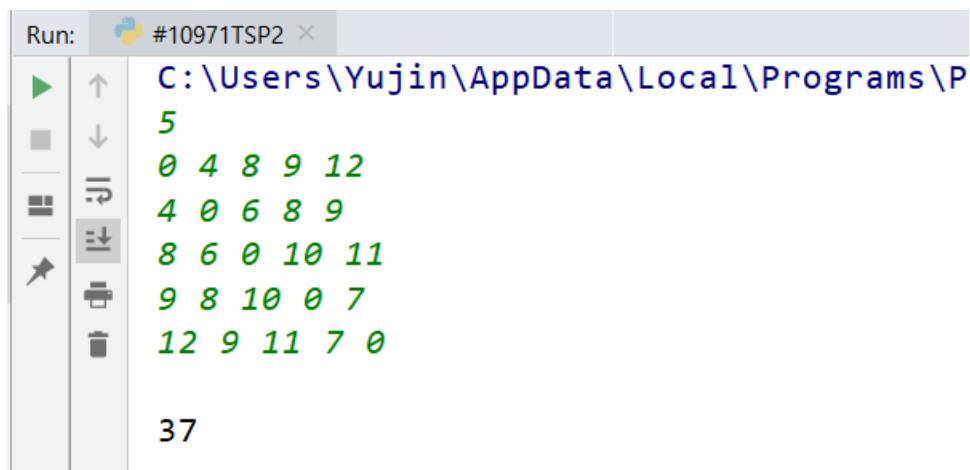
```
weight = [ [] for row in range(n)]    #경로의 가중치를 저장하는 2 차원 배열
visited = []    #도시의 방문 여부를 체크하기 위한 1 차원 배열
path = []    #방문한 실제 경로를 저장하기 위한 1 차원 배열

#간선의 가중치 값 입력
for i in range (n) :
    weight[i] = [int(k) for k in (input()).split()]
    visited.append(False)

#0 번(a)을 첫번째 시작점으로 한다.
visited[0] = True
path.append(0)

print(calcPath(0))
```

위와 같이 완전탐색을 수행하는 코드를 구현하여 TSP 문제를 푼다면,



The screenshot shows a Python IDE window titled "#10971TSP2". The main area displays a 5x5 weight matrix with values in green text:

0	4	8	9	12
4	0	6	8	9
8	6	0	10	11
9	8	10	0	7
12	9	11	7	0

Below the matrix, the number "37" is printed, representing the optimal solution value.

출력 결과인 37이 그래프 G에 대한 최적해(opt) 임을 구할 수 있다.

이를 통해 앞에서 구한 $w(\Gamma)$ 값인 39가 최적해의 2배보다 같거나 작음을 확인할 수 있다. 즉, $w(\Gamma) \leq 2 \times opt$ 를 만족하므로 앞에서 사용한 알고리즘 $ApproxTSP1(G)$ 이 2-근사 알고리즘임을 확인할 수 있다.