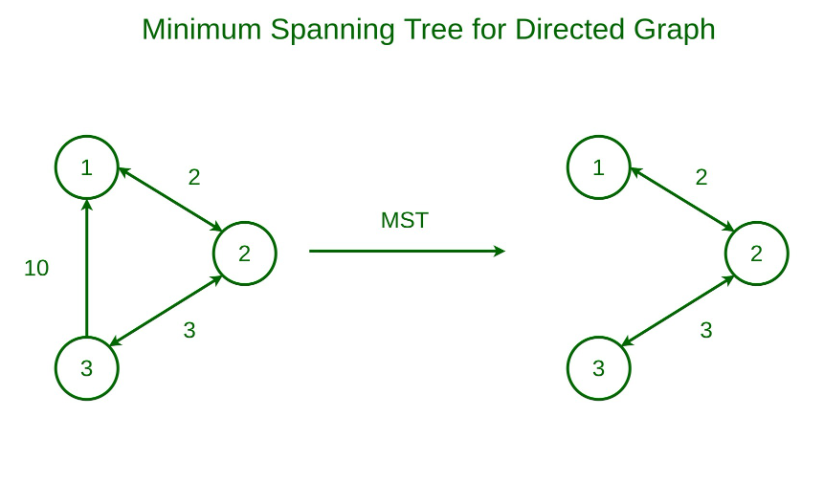
MST ( 최소 스패닝 트리 )

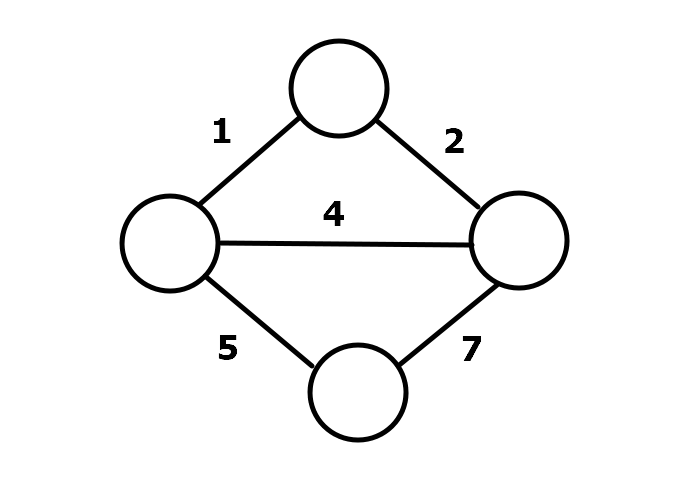
**#1197**

노드의 개수 N과 링크의 개수 M이 주어집니다.

M개 만큼 연결되어 있는 노드 A,B와 가중치 C가 주어집니다.

모든 노드를 최소 비용(가중치의 합)으로 연결했을 때의 비용을 계산하세요.





**입력**

**3 3**

**1 2 1**

**2 3 2**

**1 3 3**

**출력**

**3**

**정답코드**

**#1197(프림)**

N, M = map(int,input().split())

graph = [[] **for** \_ **in** range(N+1)] *#노드의 개수 + 1*

visited = [ 0 **for** \_ **in** range(N+1)]

*# 1번부터 3번까지 노드가 주어지니*

*# 0 1 2 3*

**for** \_ **in** range(M):

A,B,C = map(int,input().split())

graph[A].append([C,B])

graph[B].append([C,A])

*# 다익스트라!*

answer = 0

cnt = 0

q = [[0,1]] *# 1에서 출발할거다! 가중치 없이!*

*# [[1,0],[2,1],[3,1]]*

**while** q: *# q가 아무것도 없어질 때까지!*

**if** cnt == N:

**break**

weight,node = heapq.heappop(q) *# 최소비용을 꺼내줄겁니다!*

*# 방문처리!*

**if** visited[node] == 0:

visited[node] = 1

answer += weight

cnt += 1

**for** nxt **in** graph[node]:

heapq.heappush(q,nxt) *# 담아주기*

print(answer)

**#1197(크루스칼)**

*# 크루스칼*

*# 1. 모든 링크를 한번에 가져온다.*

*# 2. 링크를 연결하면서 같은 집합으로 만들어준다.*

*# 3. 만약에 이미 같은 집합이라면 연결하지 않는다.*

*# union 최적화!*

**def** \_find(x):

**while** par[x] != x : *# 루트가 아니라면*

x = par[x]

**return** x *# 너가 부모라면 리턴해줘!*

**def** \_union(a,b):

a = \_find(a)

b = \_find(b)

**if** a==b :

**return**

**if** rank[a] < rank[b]:

par[a] = b

**elif** rank[b] < rank[a]:

par[b] = a

**else**:

par[a] = b

rank[b] += 1

N,M = map(int,input().split())

link = [list(map(int,input().split())) **for** \_ **in** range(M)]

link.sort(key=**lambda** x:x[2]) *# 가중치 기준으로 정렬해줘!*

*# [A,B,C]*

par = [i **for** i **in** range(1\_000\_001)]

rank = [0 **for** \_ **in** range(1\_000\_001)]

ans = 0

**for** i **in** range(M):

A = link[i][0]

B = link[i][1]

weight = link[i][2]

A = \_find(A)

B = \_find(B)

**if** A == B:

**continue**

**else**:

\_union(A,B)

ans += weight

print(ans)