```
import heapq

def prim_mst(n, edges):
    graph = {i: [] for i in range(n)}
    for u, v, w in edges:
        graph[u].append((v, w))
        graph[v].append((u, w))
    visited = [False] * n
    min_heap = [(0, 0)]
    total_weight = 0
    mst_edges = []
    while min_heap:
        w, u = heapq.heappop(min_heap)
        if visited[u]:
            continue
        visited[u] = True
        total_weight += w
        mst_edges.append((u, w))
        for v, weight in graph[u]:
            if not visited[v]:
                heapq.heappush(min_heap, (weight, v))
    if not all(visited):
        return None, None
    return mst_edges[1:], total_weight

def main():
    n = int(input("Enter number of vertices: "))
    m = int(input("Enter number of edges: "))
    edges = []
    print("Enter each edge in format: u v weight")
    for _ in range(m):
        u, v, w = map(int, input().split())
        edges.append((u, v, w))
    mst_edges, cost = prim_mst(n, edges)
    if mst_edges is None:
        print("MST not possible — graph not connected")
    else:
        print("Edges in MST (vertex, weight):")
        for e in mst_edges:
            print(e)
        print("Total cost of MST:", cost)

if __name__ == "__main__":
    main()
```