"""N = int(input())A = list(map(int, input().split()))N, M = map(int, input().split())"""from heapq import heappush, heappopfrom itertools import accumulateINF = 10 \*\* 9def dijkstra(s, n, adj): # (始点, ノード数) dist = [INF] \* n hq = [(0, s)] # (distance, node) dist[s] = 0 seen = [False] \* n # ノードが確定済みかどうか while hq: v = heappop(hq)[1] # ノードを pop する seen[v] = True for to, cost in adj[v]: # ノード v に隣接しているノードに対して if to >= n: continue if seen[to] == False and dist[v] + cost < dist[to]: dist[to] = dist[v] + cost heappush(hq, (dist[to], to)) ss = 0 for i in range(n): if dist[i] != INF: for v, c in adj[i]: if v >= n: ss += dist[i] + c return dist, seen, ssdef maina(): N, M = map(int, input().split()) print("Yes" if N\*6 <= M else "No")def mainb(): from math import factorial N = int(input()) cnt = 0 for i in range(20)[::-1]: if factorial(i) <= N: N %= factorial(i) cnt += N // factorial(i) print(cnt)def mainc(): N, K = map(int, input().split()) A = list(map(int, input().split())) S = [] for i in enumerate(A): S.append((A[i], i)) S = sorted(S, key=lambda x: x[0]) g = N//K N %= K ans = [g] \* N for i in range(N): v, k = S[0], S[1] ans[k] += 1 print(\*ans, sep="\n")def maind(): N, M = map(int, input().split()) G = [[]for i in range(N)] for \_ in range(M): a, b, c = map(int, input().split()) G[a-1].append([b-1, c]) ans = 0 neighbor = [0]\*N for i in range(N): for j, c in G[i]: if max(i, j) <= N-2: neighbor[max(i, j) + 1] += c ne = list(accumulate(neighbor)) print((list(ne))) p = sum(ne) while N >= 1: for s in range(N): l, seen, c = dijkstra(s, N, G) ans += sum(l) - INF\*(N - sum(seen)) N -= 1 print(ans) ans += p print(ans)if \_\_name\_\_ == "\_\_main\_\_": mori = True # maina() # maina() # maina() maind()