Problem Statement

In Olympics 2020, there were athletes from n countries participating in m games. Let C = c1, c2, · · · , cn denote the n countries, and let H = h1, h2, · · · , hm denote the m games. For simplicity, assume that every country sent at most one athlete to each game, and that every game has at least three countries participating in it. For i = 1, 2, · · · , m, let Ai C denote the set of countries who sent athletes to participated in the game hi. Every game produced three medals: the gold medal, the silver medal, and the bronze medal. For j = 1, 2, · · · , n, let xj denote the total number of medals won by country cj after all games ended. Clearly, the vector (x1, x2, · · · , xn) (which is basically the Medal List of Countries) needs to satisfy certain constraints and therefore cannot be arbitrary. For example, the summation of x1, x2, · · · , xn has to be 3m. The question we consider is this: Given any vector X = (x1, x2, · · · , xn), is X a possible outcome of Olympics 2020? Design an efficient algorithm to determine if the answer if "yes" or "no", prove its correctness, and analyze its time complexity.

Main Idea

A good approach for solving this Olympic medal distribution problem is to use a network flow. With this scenario, games and countries would be nodes and the medals would flow between them based on game results. Each game node would be connected to the participating countries and there could be up to 3 of these nodes who would win a medal, since there are a maximum of 3 medals that can be given. Then each country would connect to node T based on how many medals it won. Lastly, using the Edmonds-Karp algorithm, we can compute the maximum flow from node S to node T, determining if there is a valid distribution of medals among the countries. There is a valid distribution if the total flow is equal to 3*m.

Pseudocode

Algorithm 1 OlympicMedals

function OLYMPIC MEDAL DISTRIBUTIONS(str)

Initialize global variables:

- C = c1, c2, ... , cn; set of n countries
- H = h1, h2, ... , hi; set of m games
- A = list of m sets; Ai is the set of countries who sent athletes to participate in game hi
- X = [x1, x2, ... , xn]; list of medal counts
- G = directed graph, with a node for generating medals (S) and another node for receiving medals (T)

For i in range(length(H)):

- currGame = Create node for game hi
- Add an edge between currGame and node S
- For country j in A[i]:
  - If country j is not already a node in G:
    - Create new node for country j
  - Add edge between hi and country j

For county j in C:

- xj = medals won by country j
- Add edge from country j node to node T

Compute maximum flow from node S to node T (Edmonds-Karp algorithm)

If maxFlow == 3*m (valid medal distribution)

- Return ("Yes")

Else:

- Return ("No")

end function

Proof of Correctness

To prove correctness of this algorithm, first we start with the base case when m=1. Assuming that at least 3 countries participate and the three medals can be awarded, there will be a network where node S is connected to one game and this game node is connected to 3 or more countries. If each medal is assigned to a unique country, then the total flow would be calculated as 3*m = 3*1 = 3. Thus, the algorithm would produce a valid medal distribution.

Next, assume that we already know that the algorithm works for the scenario where m = k. When there are m = k + 1 games, we can build the flow network similar to when m = k where each game node is connected to node S and the participating countries and the country nodes are connected to node T based on the number of medals they have won. With k + 1, you would add an additional game node and respective edges in accordance with the specific requirements outlined in the problem. Then, if a valid medal distribution exists, the Edmonds-Karp algorithm would be able to find the maximum flow equal to 3*(k+1). Thus, by induction this algorithm should be correct.

Time Complexity

Building the graph G will involve creating 2 nodes (S and T) and additional nodes for each game and each country. This step has O(2 + n + m), which can be reduced to O(n + m). Additionally, you also need to create edges between the differing nodes. There are three scenarios for creating edges: from node S to each game, from each game to the countries that participated in it, and from each country to node T. This means this step would have a time complexity of O(n + m + y), where y = the number of game-country combinations.

Next, the maximum flow algorithm when using Edmonds-Karp has a known time complexity of O(number of nodes * (number of edges) squared). We can use the values we calculated from building graph G. These two steps can combine their time complexities to have a total of O((n + m)*((n + m + y)*(n + m + y))). This ends up being the total time complexity of the overall algorithm.