1.Randomized Heuristic

This algorithm randomly assigns each vertex to one of two partitions (sets X and Y) with equal probability. The expected cut weight improves over multiple iterations, and the best result is selected. It is simple but provides a good baseline.

2. Greedy Heuristic

In each step, the algorithm places the next vertex into the partition that maximizes the current increase in cut weight. It evaluates the potential contribution of each unplaced vertex and chooses the one with the best immediate gain.

3. Semi-Greedy Heuristic

This is a probabilistic version of the greedy algorithm. It builds a **Restricted Candidate List (RCL)** of the best candidates based on a threshold defined by:

```
μ = w_min + ε * (w_max - w_min)
```

Where $\varepsilon \in [0, 1]$ controls greediness. A candidate is then randomly chosen from this RCL. This adds diversity and helps escape local optima.

4. Local Search

Starting from an initial solution (e.g., from Greedy or Randomized or Semi-greedy), this algorithm iteratively swaps vertices between the partitions if doing so improves the cut weight. It terminates when no single-vertex move improves the solution.

5. GRASP (Greedy Randomized Adaptive Search Procedure)

This is a heuristic that repeats the following:

Construction phase: Build an initial solution using the semi-greedy heuristic.

Local Search phase: Improve the solution via local optimization.

It repeats for multiple iterations and keeps the best solution found. GRASP balances exploration and exploitation effectively.

Which one is comparatively better and consistent ?

GRASP is the more consistent and the better among the other approaches. In our code, we have used semi-greedy method to get the primary cut. Which in itself is a better approach than randomize and greedy. Then local search also improves that primary cut to it’s max potential. So, we get the maximum cut among all the other approaches.