**Bipartite Graphs and Maximum Matching**

**Objective:**

The task is to design an algorithm to efficiently assign workers (nodes in one set) to tasks (nodes in another set) by leveraging **bipartite graphs**. The algorithm should ensure maximum matching between workers and tasks, where no two edges share a common node.

**Key Concepts:**

1. **Bipartite Graphs**:
   * A graph is bipartite if vertices can be divided into two disjoint sets (workers and tasks) with edges connecting vertices from one set to the other.
   * There are no edges within the same set.
   * A graph is bipartite only if it contains no odd-length cycles.
2. **Matching**:
   * A subset of edges such that no two edges share a vertex.
   * The goal is to maximize the number of such pairings between workers and tasks.
3. **Augmenting Path**:
   * A path starting and ending at free vertices (not part of any edge in the current matching).
   * Alternates between unmatched and matched edges.
   * Used to iteratively improve the matching until no augmenting paths remain.

**Program Phases:**

1. **Phase 1**: Check if the input graph is bipartite.
   * If not bipartite, output: The graph is not bipartite.
   * If bipartite, proceed to Phase 2.
2. **Phase 2**: Compute the maximum matching.
   * Utilize the augmenting path approach to achieve maximum matching.
   * Output:

arduino

CopyEdit

The graph is bipartite.

Maximum matching size: X

**Input Format:**

The program processes input from text files (graph{i}.txt) in adjacency list format:

* **First line**: Number of vertices and edges.
* **Subsequent lines**: Pairs of integers representing edges.

**Example**:

CopyEdit

4 5

1 2

1 3

2 4

3 4

1 4

**Output:**

* For non-bipartite graphs: The graph is not bipartite.
* For bipartite graphs:

arduino

CopyEdit

The graph is bipartite.

Maximum matching size: X

**Program Flow:**

1. Prompt the user to enter a graph name (e.g., graph1).
2. Process the file and check if it’s bipartite.
3. If bipartite, compute the maximum matching size.
4. Continue until the user inputs exit to terminate the program.