图节点的逻辑结构

```
class Vertex{
  int id;
  Vertex[] neighbors;
};
```

图遍历算法

```
Graph graph;
bool []visited;

// 图遍历框架
void traverse(Graph graph, int s){
  if(visited[s]) {
    return ;
  }

  // 经过节点s
  visited[s] = true;
  for(TreeNode neighbor : graph.neighbors(s)){
    traverse(neighbor);
  }

  // 离开节点s
  visited[s] = false;
}
```

所有节点查找

```
class Solution {
public:
  std::vector<std::vector<int>> allPathsSourceTarget(
      std::vector<std::vector<int>>& graph) {
    std::vector<int> path;
    traverse(graph, 0, path);

    return res;
  }

private:
  void traverse(std::vector<std::vector<int>> graph,
                int                              index,
                std::vector<int>&                path) {
    path.push_back(index);
    int len = graph.size();
```

```
      if (len - 1 == index) {
        res.push_back(path);
        path.pop_back();
        return;
      }

      // 递归每个相邻接点
      for (auto item : graph[index]) {
        traverse(graph, item, path);
      }

      path.pop_back();
    }

    std::vector<std::vector<int>> res;
  };
```