# ADSA PROJECT REPORT
# PROJECT NUMBER - 3
# GRAPH BASED PANCAKE PROBLEM

**TEAM MEMBERS:**
Sajan Kumar- AP17110010003
Sai Krishna Rohith Kattamuri - AP17110010024
Fahad Kamraan- AP17110010045
Sai Rishvanth K- AP17110010012
Sri Ritika K- AP17110010011

## <u>Problem:-</u>

Let us consider the simple example of ordering a pancake recipe. To make pancakes, you need a specific set of ingredients, such as eggs, milk, flour or pancake mix, oil, syrup, etc. This information, along with the quantity and portions, can be easily represented in a graph. But it is equally important to know the precise order of using these ingredients. This is where you can implement topological ordering. Other examples include making precedence charts for optimizing database queries and schedules for software projects. Here is an overview of the process for your reference:

- Call the DFS algorithm for the graph data structure to compute the finish times for the
- vertices
- Store the vertices in a list with a descending finish time order
- Execute the topological sort to return the ordered list

## <u>Solution:-</u>

Figure 1 shows the process of making pancakes represented by a graph ( directed acyclic graph). The initial problem  posed is which node to select as our first node. In a topological sort we select the first node based on the indegree of the node. We select a node as the first node if the indegree is zero. As you can see,  In Figure 1, there are 4 nodes with 0 indegree.

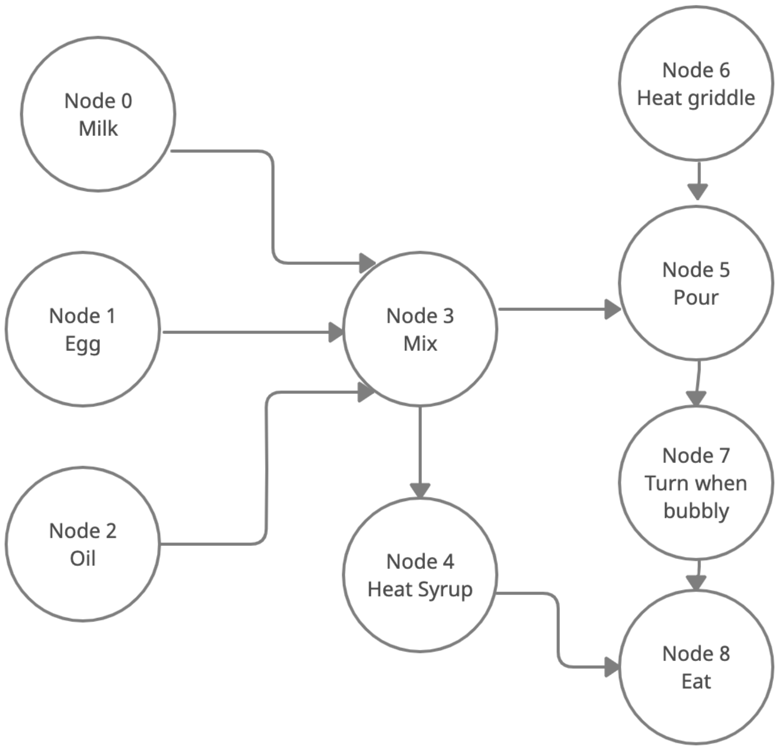


*Figure 1. Pan cake recipe as graph*

We then use the DFS algorithm to visit every node starting from a node with indegree 0. For this graph problem we are starting from the node with label 0 (Node 0 in the figures). We then visit all the nodes connected to it. If the node does not have any outdegree node then we backtrack and select the alternative node connected with previous nodes until every node is visited.

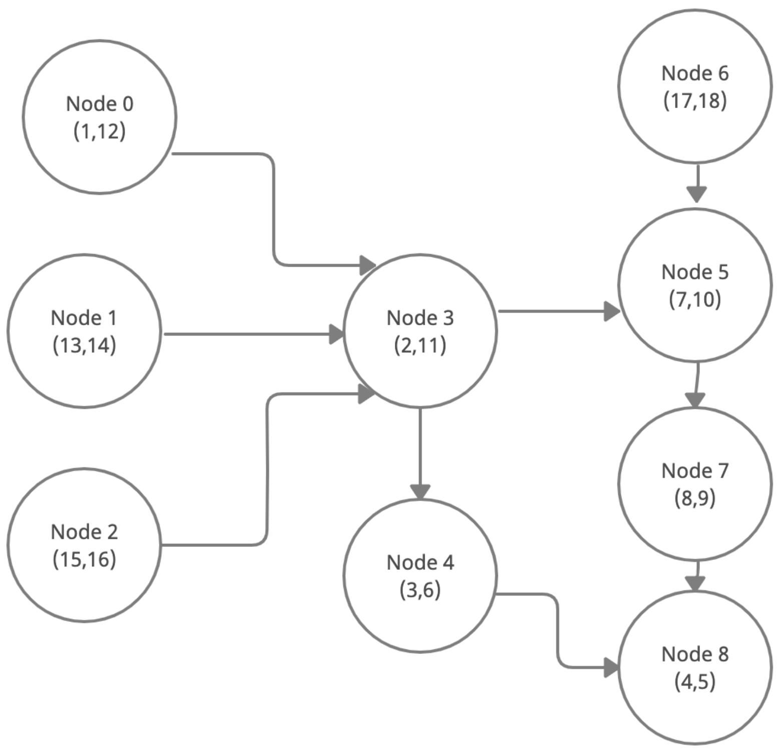


*Figure 2. Result of DFS(Depth First Search) on Pancake problem*

Figure 2 shows the arrival and finish times of each node in the pancake graph after DFS.

For deciding the process of making pancakes we head to **topological sorting** to get an order in which ingredients need to be mixed to make a proper pancake. A topological sort takes a directed acyclic graph and produces a linear ordering of all its vertices such that if the graph $G$ contains an edge $(v,w)$ then the vertex $v$ comes before the vertex $w$ in the ordering.

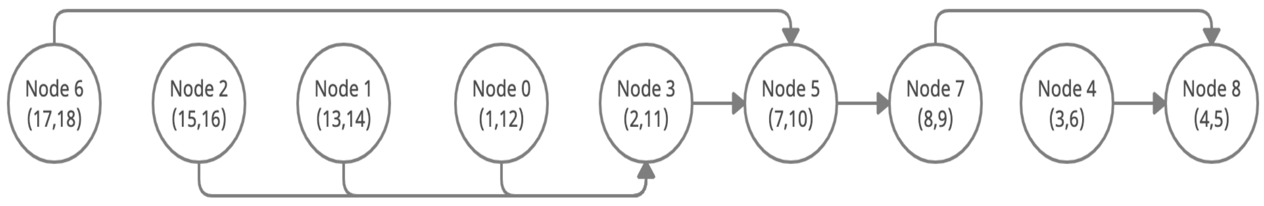


*Figure 3. Result of Topological Sort in a DAG(Directed Acyclic Graph)*

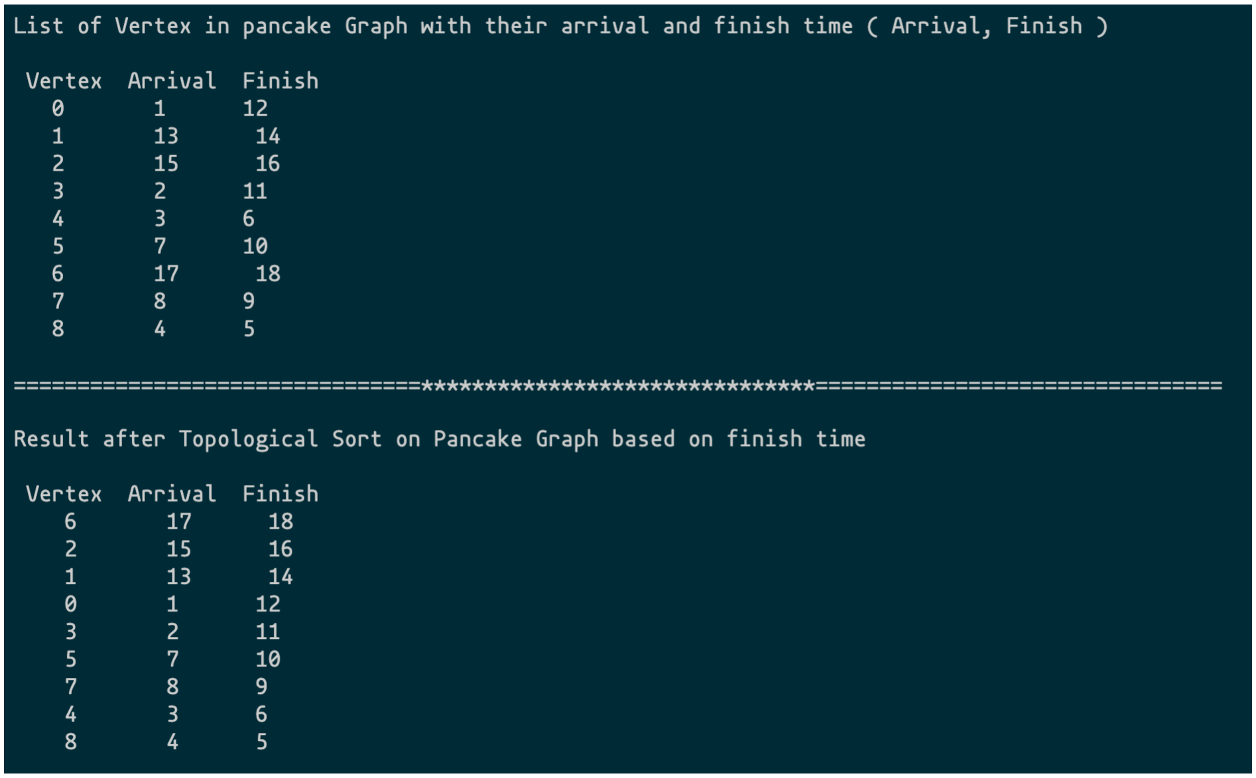
## Code:

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

struct Edge {
   int src, dest;
};
class Graph{
   public:
   vector<vector<int> > adjacentList;
   Graph(vector<Edge> const &edgeList, int N){
       adjacentList.resize(N);

       for (auto &edge: edgeList) {
           adjacentList[edge.src].push_back(edge.dest);
       }
   }
};



// DFS Method to determine arrival and finish time of each node while discovering all
nodes
void DFS(Graph const &graph, int v, vector<bool> &visited, vector<int> &start,
vector<int> &finish, int &time){
   start[v] = ++time; // Setting the starting time of vertex v
   visited[v] = true; //Marked node v as visited

   for (int i: graph.adjacentList[v]){
       if (!visited[i]){
           DFS(graph, i, visited, start, finish, time);
       }
   }

   finish[v] = ++time; // set vertex v finish time

}

bool sortByFinishTime(const vector<int> &v1, const vector<int> &v2){
   return v1[2] > v2[2];
}

void topologicalSort(vector<vector<int> > &finishOrder, int rows, int cols){
   sort(finishOrder.begin(), finishOrder.end(), sortByFinishTime);
   cout << "Result after Topological Sort on Pancake Graph based on finish time" <<
endl;
   cout << endl;
   cout << " Vertex " << " Arrival " << " Finish " << endl;
   for (int i=0; i<rows; i++)
   {
       cout << "    " << finishOrder[i][0] << "       " << finishOrder[i][1] << "
" << finishOrder[i][2] << endl;
   }
}

int main(){
   vector<Edge> edgeList = {
       {0, 3}, {1, 3}, {2, 3}, {3, 4}, {3, 5}, {4, 8}, {5, 7} , {6, 5}, {7, 8}
   };
   int N = 9;
   Graph graph(edgeList, N);
   vector<int> start(N);
   vector<int> finish(N);
   vector<bool> visited(N);
   int time = 0;

   // Performing DFS traversal to calculate finish time of all unvisited nodes.
   for (int i = 0; i < N; i++){
       if (!visited[i]){
           DFS(graph, i, visited, start, finish, time);
       }
   }
   vector<vector<int> > finishOrder(N);
   cout << endl;
   cout << "List of Vertex in pancake Graph with their arrival and finish time (
Arrival, Finish )" << endl;
   cout << endl;
   cout << " Vertex " << " Arrival " << " Finish " << endl;
   for (int i = 0; i < N; i++)
   {
       cout << "   "<< i << "        " << start[i] << "       " << finish[i]  << endl;
       finishOrder[i] = vector<int>(3);
       finishOrder[i][0] = i;
```

```cpp
        finishOrder[i][1] = start[i];
        finishOrder[i][2] = finish[i];

    }
    cout << endl;
    cout <<
"================================*******************************======================
==========" << endl;
    cout << endl;
    topologicalSort(finishOrder, N, 3);
    return 0;
}
```

Link to Github Repo

## Output:-



```
List of Vertex in pancake Graph with their arrival and finish time ( Arrival, Finish )

 Vertex  Arrival  Finish
   0       1      12
   1       13      14
   2       15      16
   3       2      11
   4       3      6
   5       7      10
   6       17      18
   7       8      9
   8       4      5

================================*******************************================================

Result after Topological Sort on Pancake Graph based on finish time

 Vertex  Arrival  Finish
    6       17      18
    2       15      16
    1       13      14
    0       1      12
    3       2      11
    5       7      10
    7       8      9
    4       3      6
    8       4      5
```

The final output which is shown in the above screenshot, shows the output for the following two subproblems:

1. DFS for determining arrival and finish time
2. Sorting the stack/vector containing the finish times.

Now, we know the exact order in which we could perform the steps in order to make a delicious pancake.
Inferring from the above output and as also shown in Figure 3, following are the steps which need to be followed to get the perfect pancake:-

1. Heat griddle
2. Pour Oil
3. Put in eggs
4. Pour Milk
5. Mix
6. Pour the mixture into the griddle
7. Turn the pancake when bubbly
8. Heat syrup
9. Eat and make it again if you like it.