Reg.No : 24BCE0554
Name   : Partha Pratim Gogoi
Topic : Assembly Line Scheduling

Algorithm

```
1  assemblyLines = 2
2  calcMinAssemblyTime(                                                [T.C: O(n)]
             noOfstations,
             processingTime[][],
             transferTime[][],
             entryTime[],
             minTime[][],
             previousLine[][]
   ):
   2.1       minTime[0][0] = entryTime[0] + processingTime[0][0]
   2.2       minTIme[1][0] = entryTime[1] + processingTime[1][0]
   2.3       for station 1→noOfstations                               [T.C: O(n)]
      2.3.1     time_stayL0 =
                             minTime[0][station-1]
                             + processingTime[0][station]
      2.3.2     time_switchFromL1 =
                             minTime[0][station-1]
                             + processingTime[0][station]
                             + transferTime[1][station-1]
      2.3.3     if time_stayL0 <= time_switchFromL1:
      2.3.4         minTime[0][station] = time_stayL0
      2.3.5         previousLine[0][station] = 0
      2.3.6     else
      2.3.7         minTime[0][station] = time_switchFromL1
      2.3.8         previousLine[0][station] = 1
      2.3.9     int time_stayL1 =
                             minTime[1][station-1]
                             + processingTime[1][station]
      2.3.10    int time_switchFromL0 =
                             minTime[0][station-1]
                             + processingTime[1][station]
                             + transferTime[0][station-1]
      2.3.11    if time_stayL1 <= time_switchL0:
      2.3.12        minTime[1][station] = time_stayL1
      2.3.13        previousLine[1][station] = 1
      2.3.14    else
      2.3.15        minTime[1][station] = time_switchFromL0
      2.3.16        previousLine[1][station] = 0
3  getOptimalExitLine(                                                 [T.C: O(1)]
             noOfstations,
             minTime[][],
             exitTime[], minTotalAssemblyTime
   ):
    3.1     totalTime_exitFromL0 =
                        minTime[0][noOfstations-1] + exitTime[0]
    3.2     totalTime_exitFromL1 =
                        minTime[1][noOfstations-1] + exitTime[1]
    3.3     if totalTime_exitFromL0 <= totalTime_exitFromL1:
    3.4         minTotalAssemblyTime = totalTime_exitFromL0
    3.5         return 0
    3.6     else
    3.7         minTotalAssemblyTime = totalTime_exitFromL1
    3.8         return 1
  4 calcMinAssemblyTime()
  5 optimalExitLine = getOptimalExitLine()
```

Time Complexity

```
    Total Time Complexity    = O(n) + O(1)
                             = O(n)
```

Source Code

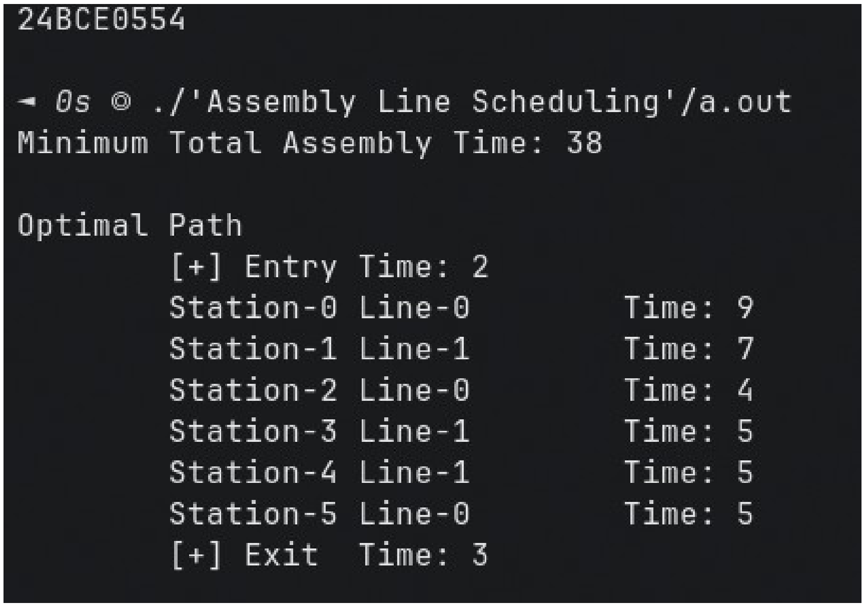
```
#include <iostream>
#include <vector>
using namespace std;

const int ASSEMBLY_LINES = 2;


/////////////////////////////////////////
///  Calculate Minimum Assembly Time  ///
/////////////////////////////////////////
/// --- Excluding exiting the assembly line
void calculateMinAssemblyTime(
            int noOfstations,
            vector<vector<int>> processingTime,
            vector<vector<int>> transferTime,
            vector<int> entryTime,
            vector<vector<int>>& minTime,
            vector<vector<int>>& previousLine
      ) {
      // Starting from:- Station-0
      minTime[0][0] = entryTime[0] + processingTime[0][0];
      minTime[1][0] = entryTime[1] + processingTime[1][0];

      // Continuing from Station-1 (whichever line is picked)
      for (int station=1; station < noOfstations; station++) {
            // -------------------------------------- //
            // --- Next Station taken from Line-0 --- //
            // -------------------------------------- //
            // --- 1. Time:- Previous station also on Line-0
            int time_stayL0 =
                  minTime[0][station-1]
                  + processingTime[0][station];

            // --- 2. Time:- Switch from Line-1
            int time_switchFromL1 =
                  minTime[1][station-1]
                  + processingTime[0][station]
                  + transferTime[1][station-1];

            // --- 3. Compare assembly time and select least time
            if (time_stayL0 <= time_switchFromL1) {
                  minTime[0][station] = time_stayL0;
                  previousLine[0][station] = 0;
            } else {
                  minTime[0][station] = time_switchFromL1;
                  previousLine[0][station] = 1;
            }

            // ------------------------------------- //
            // --- Next Station taken from Line-1 --- //
            // ------------------------------------- //
            // --- 1. Time:- Previous station also on Line-1
            int time_stayL1 =
                  minTime[1][station-1]
                  + processingTime[1][station];

            // --- 2. Time:- Switch from Line-0
            int time_switchFromL0 =
                  minTime[0][station-1]
                  + processingTime[1][station]
                  + transferTime[0][station-1];

            // --- 3. Compare assembly time and select least time
            if (time_stayL1 <= time_switchFromL0) {
                  minTime[1][station] = time_stayL1;
                  previousLine[1][station] = 1;
            } else {
                  minTime[1][station] = time_switchFromL0;
                  previousLine[1][station] = 0;
            }
      }
}

//////////////////////////////////
///  Choose Optimal Exit Line  ///
//////////////////////////////////
int getOptimalExitLine(
            int noOfstations,
            vector<vector<int>>& minTime,
            vector<int> exitTime,
            int& minTotalAssemblyTime
      ) {
      int totalTime_exitFromL0 =
            minTime[0][noOfstations-1] + exitTime[0];
      int totalTime_exitFromL1 =
            minTime[1][noOfstations-1] + exitTime[1];

      if (totalTime_exitFromL0 <= totalTime_exitFromL1) {
            minTotalAssemblyTime = totalTime_exitFromL0;
            return 0;
      } else {
            minTotalAssemblyTime = totalTime_exitFromL1;
            return 1;
      }
```

```
}

////////////////////////////
///  Build Optimal Path  ///
////////////////////////////
vector<int> getOptimalPath(
            int noOfstations,
            int exitLineUsed,
            vector<vector<int>> previousLine
      ) {
      vector<int> chosenLine(noOfstations);
      chosenLine[noOfstations-1] = exitLineUsed;

      for (int station = noOfstations-1; station>0; station--) {
            chosenLine[station-1] =
                  previousLine[chosenLine[station]][station];
      }

      return chosenLine;
}

/////////////////////////////////
///  Print Assembly Schedule  ///
/////////////////////////////////
void printAssemblySchedule(
            int minTotalAssemblyTime,
            vector<int> chosenLine,
            vector<vector<int>> minTime,
            vector<int> entryTime,
            vector<int> exitTime
      ) {
      cout << "Minimum Total Assembly Time: " << minTotalAssemblyTime << endl << endl;

      cout << "Optimal Path" << endl;
      cout << "\t[+] Entry Time: " << entryTime[chosenLine[0]] << endl;
      cout << "\tStation-0 Line-" << chosenLine[0]
            << "\tTime: " << minTime[chosenLine[0]][0] << endl;
      for (int station=1; station < chosenLine.size(); station++) {
            cout << "\tStation-" << station << " ";
            cout << "Line-" << chosenLine[station] << "\t";

            int stationTime =
                  minTime[chosenLine[station]][station]
                  - minTime[chosenLine[station-1]][station-1];
            cout << "Time: " << stationTime << endl;
      }
      cout << "\t[+] Exit  Time: " << exitTime[chosenLine[chosenLine.size()-1]] <<
endl;
}


/////////////////////
///  Driver Code  ///
/////////////////////
int main() {
      // --------------------------------- //
      // --- 1. Setup of Assembly Line --- //
      // --------------------------------- //
      int noOfstations = 6;
```

```
vector<vector<int>> processingTime = {
      {7,9,3,4,8,4},
      {8,5,8,4,5,7}
};
vector<vector<int>> transferTime = {
      {2,3,1,3,4},
      {2,1,2,2,1}
};
vector<int> entryTime = {
      2,
      4
};
vector<int> exitTime = {
      3,
      2
};


// ---------------------------------- //
// --- 2. Initializing time arrays --- //
// ---------------------------------- //
vector<vector<int>> minTime(
                  ASSEMBLY_LINES,
                  vector<int>(noOfstations));
vector<vector<int>> previousLine(
                  ASSEMBLY_LINES,
                  vector<int>(noOfstations));


// ---------------------------------------- //
// --- 3. Get Minimum Assembly Time Path --- //
// ---------------------------------------- //
calculateMinAssemblyTime(
      noOfstations,
      processingTime,
      transferTime,
      entryTime,
      minTime,
      previousLine
);
int minTotalAssemblyTime;
int optimalExitLine =
      getOptimalExitLine(
            noOfstations,
            minTime,
            exitTime,
            minTotalAssemblyTime
      );
vector<int> optimalPath =
      getOptimalPath(
            noOfstations,
            optimalExitLine,
            previousLine
      );


// -------------------------------- //
// --- 4. Print out Optimal Path --- //
```

```
	// ------------------------------- //
	printAssemblySchedule(
		minTotalAssemblyTime,
		optimalPath,
		minTime,
		entryTime,
		exitTime
	);

	return 0;
}
```

Sample Output



```
24BCE0554

◄ 0s ◎ ./'Assembly Line Scheduling'/a.out
Minimum Total Assembly Time: 38

Optimal Path
        [+] Entry Time: 2
        Station-0 Line-0        Time: 9
        Station-1 Line-1        Time: 7
        Station-2 Line-0        Time: 4
        Station-3 Line-1        Time: 5
        Station-4 Line-1        Time: 5
        Station-5 Line-0        Time: 5
        [+] Exit  Time: 3
```