CS 323_33 Programming Language: C++

Project #9 Dependency Scheduling

Andres Quintero

Due Date:

Soft copy: 4/19/2020

Hard copy: 4/19/2020

Main*

```
Step 0: inFile1, inFile1, outFile1, outFile2 ←open
        numProcs  ← from argv[3]
         if (numProcs > numNodes)
               numProcs ← numNodes // means unlimited processors, why?

Step 1: initialization (…) // see algorithm below.

Step 2: loadOpen(…) // see algorithm below.

Step 3: printList(Open, outFile2) // debug print

Step 4 loadProcAry(…) // see algorithm below.

Step 5: hasCycle ← checkCycle (…) // on your own, see the description in the above.

          if hasCycle == true
                      output error message to console: "there is cycle in the graph!!!"
                      and exit the program
step 6: printScheduleTable (outFile1) // print intermediate schedule table to outFile1

step 7: currentTime++

step 8: updateProcTime (…) // on your own, see the description in the above.

step 9: deleteFinishedNodes (…)

step 10: repeat step 2 to step 11 until graphIsEmpty (…)

step 11: printScheduleTable (outFile1) // The final schedule table to outFile1

step 12: close all files
```

Source code:

```
#include <iostream>
#include <string>
#include <fstream>

using namespace std;


class Scheduling {

    class Node {
    public:
    int jobId;
    int jobTime;
    int dependentCount;
    Node* next;
    Node(){

    }

    Node(int id, int time, int dCount){
        jobId = id;
        jobTime = time;
        dependentCount = dCount;
        next = NULL;
    }

    void printNode(ofstream& outFile){
        if(this->next == NULL){
            outFile << "(" << "jobId:" << jobId << ", " << "dependetCount: " << dependentCount <<
", next.jobId:" << "NULL" << ") -> NULL";
        } else {
            outFile << "(" << "jobId:" << jobId << ", " << "dependetCount: " << dependentCount <<
", next.jobId:" << next->jobId << ") ->";
        }
    }
};

class Job {
    public:
    int jobTime;
    int onWhichProc;
    int onOpen;
    int parentCount;
    int depedentCount;

    Job(){ // set all to zero?
        jobTime = 0;
        onWhichProc = 0;
        onOpen = 0;
        parentCount = 0;
        depedentCount = 0;
    }

};

class Proc {
    public:
    int doWhichJob = -1;
    int timeRemain = 0; // start at zero AKA available
};


 public:
    int numNodes;
    int numProcs;
    int procUsed;

    Job* jobAry;
    Proc* procAry;
    Node* Open;

    int** adjMatrix;
    int* parentCountAry;
    int* dependentCountAry;
    int* onGraphAry;
    int totalJobTimes;
    int** scheduleTable;

    int currentTime; // This could be outside the code however it will make passing the varible
around an pain so KEEP IN CLASS!

    void initialization(ifstream& inputFile1, ifstream& inputFile2, int numberOfProcs){
        // 0
        procUsed = 0;
        currentTime = 0; // Maybe be class variable? Does it get passed around?
        Open = new Node(0,0,0); // headList

        // 1
        numProcs = numberOfProcs; // from argument



        // 2
        inputFile1 >> numNodes;
        if (numProcs > numNodes){
            numProcs = numNodes; //  because each node can have thier own processors
        }
        // cout << "numNodes: " << numNodes << " " << numProcs << endl;

        // 3
        // adjMatrix init and then set to zero
        adjMatrix = new int*[numNodes+1];
        for(int i = 0; i < numNodes+1; i++){
            adjMatrix[i] = new int[numNodes+1];
        }
        // to zeros
        for(int i  = 0; i < numNodes+1; i++){
            for(int j = 0; j < numNodes+1; j++){
                adjMatrix[i][j] = 0;
            }
        }

        parentCountAry = new int[numNodes+1];
        for(int i = 1 ; i < numNodes+1; i++){parentCountAry[i] = 0;}

        dependentCountAry = new int[numNodes+1];
        for(int i = 1 ; i < numNodes+1; i++){dependentCountAry[i] = 0;}

        onGraphAry = new int[numNodes+1];
        for(int i = 1; i < numNodes + 1; i++){
            onGraphAry[i] = 1;
        }



        jobAry = new Job[numNodes+1];
        // for(int i = 1; i < numNodes+1; i++){
        //     jobAry[i] = Job();
        // }
        procAry = new Proc[numProcs+1];
        // for(int i = 1; i < numProcs+1; i++){
        //     procAry[i] = Proc();
        // }
        // 4
        loadMatrix(inputFile1);

        // 5
        computeParentCount();

        // 6
        computeDependentCount();


        // 7
        totalJobTimes = constructJobAry(inputFile2);
        // cout << "Total job times: " << totalJobTimes << endl;

        // need totalJobTimes first
        scheduleTable = new int*[numProcs+1];
        for(int i = 0; i < numProcs+1; i++){
            scheduleTable[i] = new int[totalJobTimes+1];
        }

        // for(int i  = 0; i < numProcs+1; i++){
        //     for(int j = 0; j < totalJobTimes+1; j++){
        //         scheduleTable[i][j] = 0;
        //     }
        // }

    }

    void loadMatrix(ifstream& inputFile1){
        int parent;
        int dependent;
        while(!inputFile1.eof()){
            inputFile1 >> parent;
            inputFile1 >> dependent;
            adjMatrix[parent][dependent] = 1;
        }
    }

    int constructJobAry(ifstream& inputFile2){
        int totalTime = 0;
        int emptyRead;
        inputFile2 >> emptyRead; // clear the header information
        int nodeID;
        int jobTime;
        while(!inputFile2.eof()){


            inputFile2 >> nodeID;
            inputFile2 >> jobTime;

            // cout << "nodeID : " << nodeID << " jobtime: " << jobTime << endl;
            totalTime += jobTime;

            // 2
            jobAry[nodeID].jobTime = jobTime;
            jobAry[nodeID].onWhichProc = -1;
            jobAry[nodeID].onOpen = 0;
            jobAry[nodeID].parentCount = parentCountAry[nodeID];
            jobAry[nodeID].depedentCount = dependentCountAry[nodeID];
        }
        return totalTime;
    }

    void computeParentCount(){
        for(int nodeId = 1; nodeId < numNodes+1; nodeId++){
            int sum = 0;
            for(int i = 1; i < numNodes+1; i++){
                sum += adjMatrix[i][nodeId];
            }
```

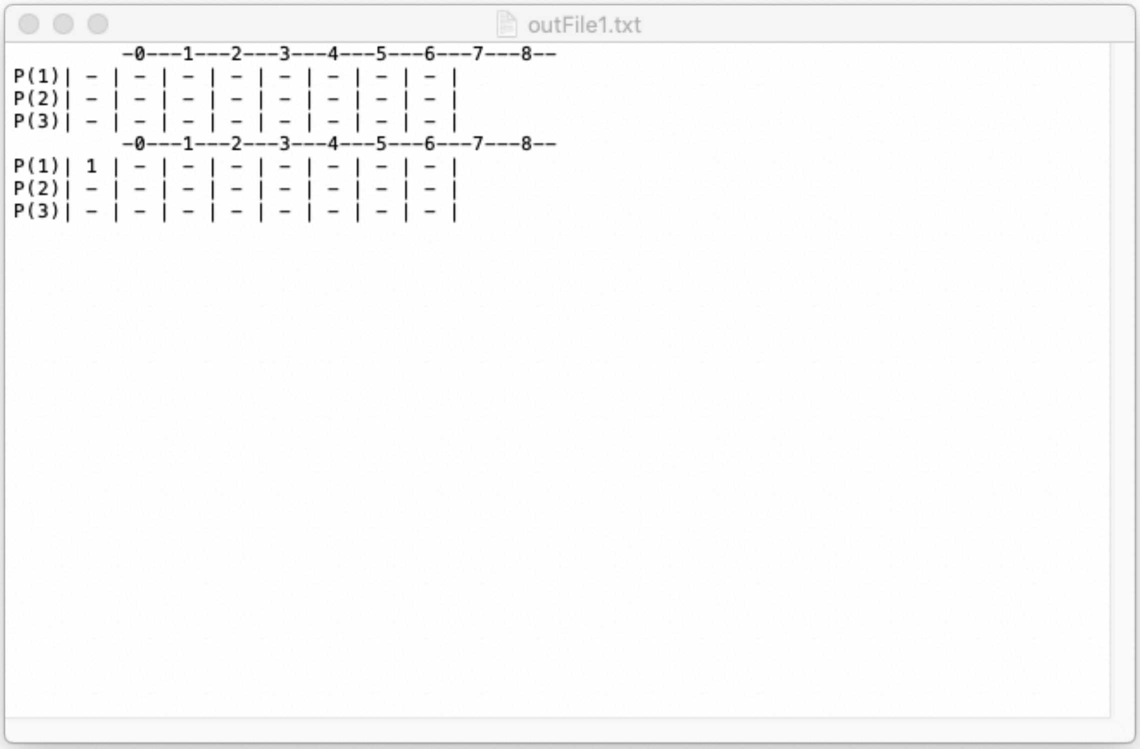
```
            parentCountAry[nodeId] = sum;
            jobAry[nodeId].parentCount = parentCountAry[nodeId];
        }

    }

    void computeDependentCount(){
        for(int nodeId = 1; nodeId < numNodes+1; nodeId++){
            int sum = 0;
            for(int j = 1; j < numNodes+1; j++){
                sum += adjMatrix[nodeId][j];
            }
            dependentCountAry[nodeId] = sum;
            jobAry[nodeId].depedentCount = dependentCountAry[nodeId];
        }
    }

    int findOrphan(){
        for(int i = 1; i < numNodes+1; i++){
            // cout << "node: " << i << endl;
            // cout << "parentCountAry[i]: " << parentCountAry[i] << endl;
            // cout << "jobAry[i].onOpen : " << jobAry[i].onOpen << endl;
            // cout << "jobAry[i].onWhichProc" << jobAry[i].onWhichProc << endl;
            if(parentCountAry[i] <= 0 && jobAry[i].onOpen == 0 && jobAry[i].onWhichProc <= 0){
                // cout << "**************************** found orphan node: " << i << endl;
                return i;
            }
        }
        return -1;
    }


    // Why isnt 9 at the end of the list?!
    Node* findSpot(Node* newNode){
        Node* Spot = Open; //Head
        while(Spot->next != NULL && dependentCountAry[Spot->next->jobId] >=
dependentCountAry[newNode->jobId]){
            Spot = Spot->next;
        }
        return Spot;
    }

    void listInsert(Node* newNode){
        Node* Spot = findSpot(newNode);
        newNode->next = Spot->next;
        Spot->next = newNode;
    }

    void loadOpen(){
        // cout << "calling loadOpen" << endl;


        int orphanNode = findOrphan();
        if(orphanNode == -1){return;}
        int jId;
        int jt;
        // cout << "foudn orphan: " << orphanNode << endl;
        while(orphanNode != -1){
            if (orphanNode > 0) {
                // cout << "** *** foudn orphan: " << orphanNode << endl;
            jId = orphanNode;
            jt = jobAry[jId].jobTime;
            Node* newNode = new Node(jId, jt, dependentCountAry[jId]);
            listInsert(newNode);
            jobAry[jId].onOpen = 1; // bool flag better?
            }
            orphanNode = findOrphan();
        }
    }

    void loadProcAry(int currentTime){
```

```
    int availProc = findProcessor();
    while(availProc > 0 && Open->next != NULL && procUsed < numProcs){
        if( availProc > 0){
            procUsed++;

            Node* newJob = Open->next;
            Open->next = Open->next->next;
            newJob->next = NULL;

            int jobId = newJob->jobId;
            int jobTime = newJob->jobTime;
            procAry[availProc].doWhichJob = jobId;
            procAry[availProc].timeRemain = jobTime;
            putJobOnTable(availProc, currentTime, jobId, jobTime);
        }
        availProc = findProcessor();
    }
}

void printList(ofstream& outFile2){
    outFile2 << "head" << "-> ";
    Node* printSpot = Open->next;
    while(printSpot != NULL){
        printSpot->printNode(outFile2);
        printSpot = printSpot->next;
    }
    outFile2 << endl;
}

void printScheduleTable(ofstream& outFile1){
    // Times
    outFile1 << "\t ";
    for(int i = 0; i <= totalJobTimes; i++){
        outFile1 << "-" << i << "--";
    }
    outFile1 << endl;

    // Each proccessor P(i)| Pi.| |  || |
    for(int proc = 1; proc < numProcs + 1; proc++){
        outFile1 << "P(" << proc << ")|";
        for(int time = 1; time < totalJobTimes+1; time++){
            if(scheduleTable[proc][time] == 0){
                outFile1 << " - |";
            } else {
                outFile1 << " " << time << " |";
            }
        }
        outFile1 << endl;
    }
}


void putJobOnTable(int availProc, int currentTime, int jobId, int jobTime){
    int time = currentTime;
    int endTime = time + jobTime;

    while(time < endTime){
        scheduleTable[availProc][time] = jobId;
        time++;
    }
}

int findProcessor(){
    for(int i = 1; i < numProcs + 1; i++){
        if(procAry[i].timeRemain <= 0){
            return i;
        }
    }
    return -1;
}
```

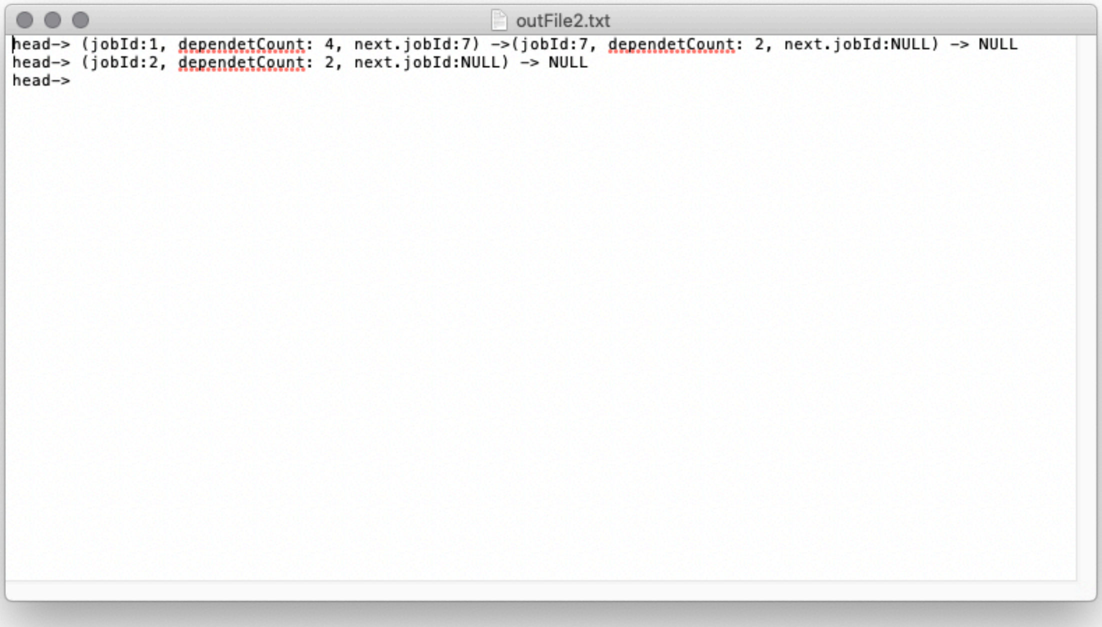
```
bool checkCycle(){
    if(Open->next == NULL && !graphIsEmpty() && checkCond3()){
        return true;
    } else {
        return false;
    }
}

bool graphIsEmpty(){
    for(int i = 1; i < numNodes + 1; i++){
        if(onGraphAry[i] != 0){
            return false;
        }
    }
    return true;
}

bool checkCond3(){
    for(int i = 1; i < numProcs+1; i++){
        if(procAry[i].doWhichJob != -1){
            return false;
        }
    }
    return true;
}

void updateProcTime(){
    for(int i = 1; i < numProcs + 1; i++){
        if(procAry[i].timeRemain != 0){
            // cout << "timeReain " << procAry[i].timeRemain << endl;
            procAry[i].timeRemain--;
        }
    }
}

int findDoneProc(){
    for(int i = 1; i < numProcs + 1; i++){
        if(procAry[i].doWhichJob !=-1 && procAry[i].timeRemain <= 0){
            int j = procAry[i].doWhichJob;
            procAry[i].doWhichJob = -1;
            return j;
        }
    }
    // no more finished procs
    return -1;
}

void deleteEdge(int jobId){
    // cout << "**** from deleteEdge" << endl;
    for(int dependent = 1 ; dependent < numNodes+1 ; dependent++){
        if(adjMatrix[jobId][dependent] > 0){
            parentCountAry[dependent]--;
        }
    }
}

void deleteFinishedNodes(){
    int j = findDoneProc();
    while(j > 0){
        // cout << "J:" << j << endl;
        if(j > 0){
            // cout << "HELLO\n\n\n\n\n" << endl;
            onGraphAry[j] = 0;
            deleteEdge(j);
        }
        j = findDoneProc();
    }
}
```

```
}; // End of Schedule class


int main(int argc, char* argv[]){
    bool hasCycle;

    ifstream inFile1(argv[1]);
    ifstream inFile2(argv[2]);
    int numberOfProc = stoi(argv[3]);
    ofstream outFile1(argv[4]);
    ofstream outFile2(argv[5]);

    Scheduling S;
    S.initialization(inFile1, inFile2, numberOfProc);

    // for (int i = 1; i < S.numNodes+1; i++){
        // cout << "node " << i << " has " << S.parentCountAry[i] << " parents" << endl;
    // }

    while(!S.graphIsEmpty()){
    //     for (int i = 1; i < S.numNodes+1; i++){
    //     cout << "node " << i << " has " << S.parentCountAry[i] << " parents" << endl;
    // }
        // for(int i = 1; i < S.numNodes+1; i++){
        //     cout << S.onGraphAry[i];
        // }
        // cout << endl;
        S.loadOpen();

        // cout << "is open empty? " << (S.Open->next == NULL) << endl;
        S.printList(outFile2);
        S.loadProcAry(S.currentTime);
        hasCycle = S.checkCycle();
        if(hasCycle){
            cout << "Cycle detected!! Exititing Program Now...." << endl;
            exit(1);
        }

        S.printScheduleTable(outFile1);
        S.currentTime++;
        S.updateProcTime();
        S.deleteFinishedNodes();
    }

    S.printScheduleTable(outFile1);

    inFile1.close();
    inFile2.close();
    outFile1.close();
    outFile2.close();
}
```

Data1 with 3 Processors (Cycle)

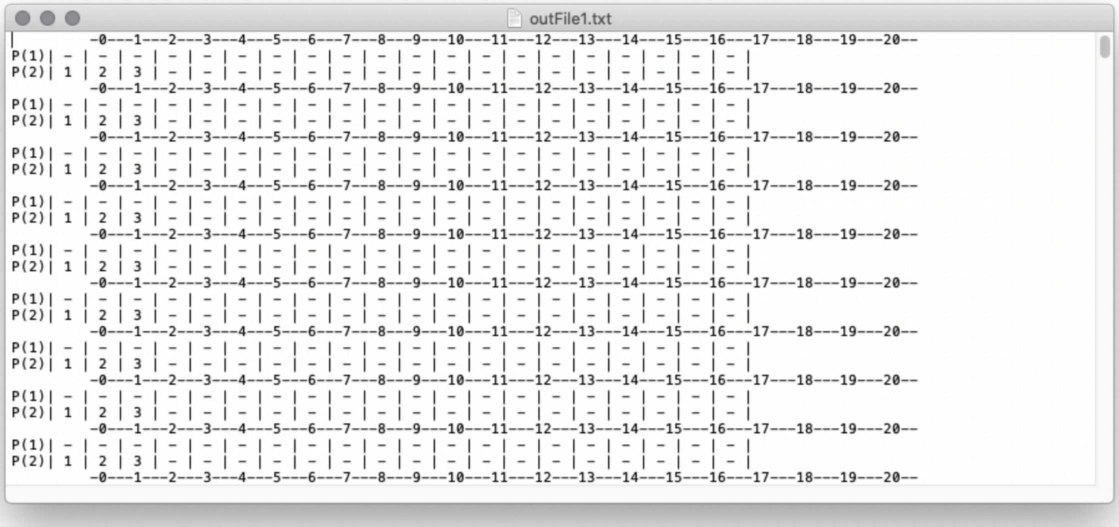


```
         -0---1---2---3---4---5---6---7---8--
P(1)| - | - | - | - | - | - | - | - |
P(2)| - | - | - | - | - | - | - | - |
P(3)| - | - | - | - | - | - | - | - |
         -0---1---2---3---4---5---6---7---8--
P(1)| 1 | - | - | - | - | - | - | - |
P(2)| - | - | - | - | - | - | - | - |
P(3)| - | - | - | - | - | - | - | - |
```



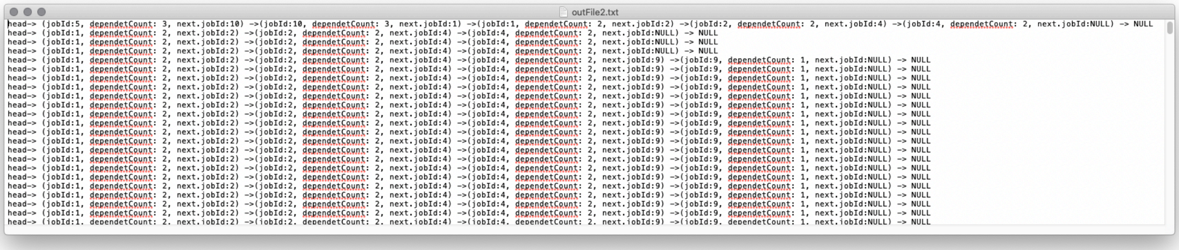
```
head-> (jobId:1, dependetCount: 4, next.jobId:7) ->(jobId:7, dependetCount: 2, next.jobId:NULL) -> NULL
head-> (jobId:2, dependetCount: 2, next.jobId:NULL) -> NULL
head->
```

Data2 with 2 Processors (**Infinite Loop)**



outFile1.txt

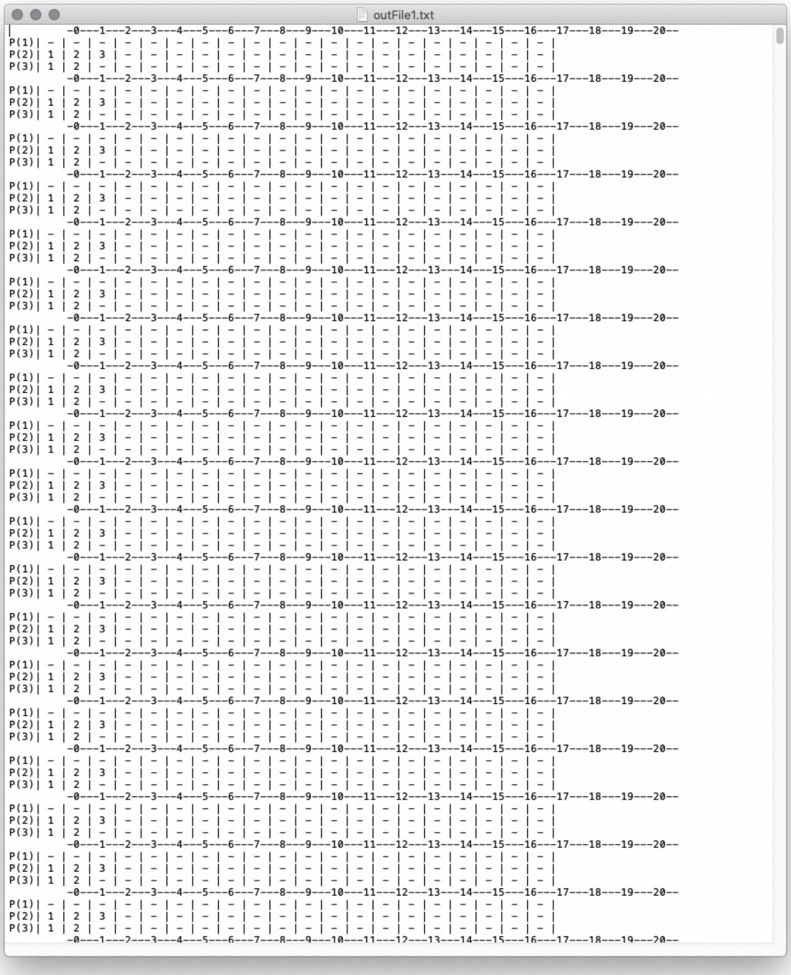
```
         -0---1---2---3---4---5---6---7---8---9---10---11---12---13---14---15---16---17---18---19---20--
P(1)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(2)| 1 | 2 | 3 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
         -0---1---2---3---4---5---6---7---8---9---10---11---12---13---14---15---16---17---18---19---20--
P(1)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(2)| 1 | 2 | 3 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
         -0---1---2---3---4---5---6---7---8---9---10---11---12---13---14---15---16---17---18---19---20--
P(1)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(2)| 1 | 2 | 3 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
         -0---1---2---3---4---5---6---7---8---9---10---11---12---13---14---15---16---17---18---19---20--
P(1)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(2)| 1 | 2 | 3 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
         -0---1---2---3---4---5---6---7---8---9---10---11---12---13---14---15---16---17---18---19---20--
P(1)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(2)| 1 | 2 | 3 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
         -0---1---2---3---4---5---6---7---8---9---10---11---12---13---14---15---16---17---18---19---20--
P(1)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(2)| 1 | 2 | 3 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
         -0---1---2---3---4---5---6---7---8---9---10---11---12---13---14---15---16---17---18---19---20--
P(1)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(2)| 1 | 2 | 3 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
         -0---1---2---3---4---5---6---7---8---9---10---11---12---13---14---15---16---17---18---19---20--
P(1)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(2)| 1 | 2 | 3 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
         -0---1---2---3---4---5---6---7---8---9---10---11---12---13---14---15---16---17---18---19---20--
P(1)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(2)| 1 | 2 | 3 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
         -0---1---2---3---4---5---6---7---8---9---10---11---12---13---14---15---16---17---18---19---20--
```



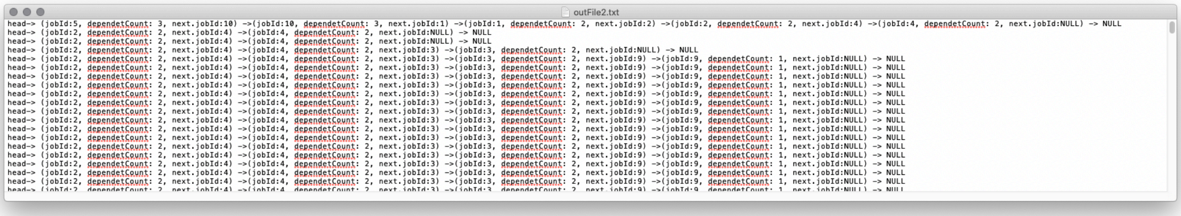
outFile2.txt

```
head-> (jobId:5, dependetCount: 3, next.jobId:10) ->(jobId:10, dependetCount: 3, next.jobId:1) ->(jobId:1, dependetCount: 2, next.jobId:2) ->(jobId:2, dependetCount: 2, next.jobId:4) ->(jobId:4, dependetCount: 2, next.jobId:NULL) -> NULL
head-> (jobId:1, dependetCount: 2, next.jobId:2) ->(jobId:2, dependetCount: 2, next.jobId:4) ->(jobId:4, dependetCount: 2, next.jobId:NULL) -> NULL
head-> (jobId:1, dependetCount: 2, next.jobId:2) ->(jobId:2, dependetCount: 2, next.jobId:4) ->(jobId:4, dependetCount: 2, next.jobId:NULL) -> NULL
head-> (jobId:1, dependetCount: 2, next.jobId:2) ->(jobId:2, dependetCount: 2, next.jobId:4) ->(jobId:4, dependetCount: 2, next.jobId:NULL) -> NULL
head-> (jobId:1, dependetCount: 2, next.jobId:2) ->(jobId:2, dependetCount: 2, next.jobId:4) ->(jobId:4, dependetCount: 2, next.jobId:9) ->(jobId:9, dependetCount: 1, next.jobId:NULL) -> NULL
head-> (jobId:1, dependetCount: 2, next.jobId:2) ->(jobId:2, dependetCount: 2, next.jobId:4) ->(jobId:4, dependetCount: 2, next.jobId:9) ->(jobId:9, dependetCount: 1, next.jobId:NULL) -> NULL
head-> (jobId:1, dependetCount: 2, next.jobId:2) ->(jobId:2, dependetCount: 2, next.jobId:4) ->(jobId:4, dependetCount: 2, next.jobId:9) ->(jobId:9, dependetCount: 1, next.jobId:NULL) -> NULL
head-> (jobId:1, dependetCount: 2, next.jobId:2) ->(jobId:2, dependetCount: 2, next.jobId:4) ->(jobId:4, dependetCount: 2, next.jobId:9) ->(jobId:9, dependetCount: 1, next.jobId:NULL) -> NULL
head-> (jobId:1, dependetCount: 2, next.jobId:2) ->(jobId:2, dependetCount: 2, next.jobId:4) ->(jobId:4, dependetCount: 2, next.jobId:9) ->(jobId:9, dependetCount: 1, next.jobId:NULL) -> NULL
head-> (jobId:1, dependetCount: 2, next.jobId:2) ->(jobId:2, dependetCount: 2, next.jobId:4) ->(jobId:4, dependetCount: 2, next.jobId:9) ->(jobId:9, dependetCount: 1, next.jobId:NULL) -> NULL
head-> (jobId:1, dependetCount: 2, next.jobId:2) ->(jobId:2, dependetCount: 2, next.jobId:4) ->(jobId:4, dependetCount: 2, next.jobId:9) ->(jobId:9, dependetCount: 1, next.jobId:NULL) -> NULL
head-> (jobId:1, dependetCount: 2, next.jobId:2) ->(jobId:2, dependetCount: 2, next.jobId:4) ->(jobId:4, dependetCount: 2, next.jobId:9) ->(jobId:9, dependetCount: 1, next.jobId:NULL) -> NULL
head-> (jobId:1, dependetCount: 2, next.jobId:2) ->(jobId:2, dependetCount: 2, next.jobId:4) ->(jobId:4, dependetCount: 2, next.jobId:9) ->(jobId:9, dependetCount: 1, next.jobId:NULL) -> NULL
head-> (jobId:1, dependetCount: 2, next.jobId:2) ->(jobId:2, dependetCount: 2, next.jobId:4) ->(jobId:4, dependetCount: 2, next.jobId:9) ->(jobId:9, dependetCount: 1, next.jobId:NULL) -> NULL
head-> (jobId:1, dependetCount: 2, next.jobId:2) ->(jobId:2, dependetCount: 2, next.jobId:4) ->(jobId:4, dependetCount: 2, next.jobId:9) ->(jobId:9, dependetCount: 1, next.jobId:NULL) -> NULL
head-> (jobId:1, dependetCount: 2, next.jobId:2) ->(jobId:2, dependetCount: 2, next.jobId:4) ->(jobId:4, dependetCount: 2, next.jobId:9) ->(jobId:9, dependetCount: 1, next.jobId:NULL) -> NULL
head-> (jobId:1, dependetCount: 2, next.jobId:2) ->(jobId:2, dependetCount: 2, next.jobId:4) ->(jobId:4, dependetCount: 2, next.jobId:9) ->(jobId:9, dependetCount: 1, next.jobId:NULL) -> NULL
head-> (jobId:1, dependetCount: 2, next.jobId:2) ->(jobId:2, dependetCount: 2, next.jobId:4) ->(jobId:4, dependetCount: 2, next.jobId:9) ->(jobId:9, dependetCount: 1, next.jobId:NULL) -> NULL
head-> (jobId:1, dependetCount: 2, next.jobId:2) ->(jobId:2, dependetCount: 2, next.jobId:4) ->(jobId:4, dependetCount: 2, next.jobId:9) ->(jobId:9, dependetCount: 1, next.jobId:NULL) -> NULL
head-> (jobId:1, dependetCount: 2, next.jobId:2) ->(jobId:2, dependetCount: 2, next.jobId:4) ->(jobId:4, dependetCount: 2, next.jobId:9) ->(jobId:9, dependetCount: 1, next.jobId:NULL) -> NULL
head-> (jobId:1, dependetCount: 2, next.jobId:2) ->(jobId:2, dependetCount: 2, next.jobId:4) ->(jobId:4, dependetCount: 2, next.jobId:9) ->(jobId:9, dependetCount: 1, next.jobId:NULL) -> NULL
head-> (jobId:1, dependetCount: 2, next.jobId:2) ->(jobId:2, dependetCount: 2, next.jobId:4) ->(jobId:4, dependetCount: 2, next.jobId:9) ->(jobId:9, dependetCount: 1, next.jobId:NULL) -> NULL
head-> (jobId:1, dependetCount: 2, next.jobId:2) ->(jobId:2, dependetCount: 2, next.jobId:4) ->(jobId:4, dependetCount: 2, next.jobId:9) ->(jobId:9, dependetCount: 1, next.jobId:NULL) -> NULL
```

Data2 with 3 Processors (**Infinite Loop**)

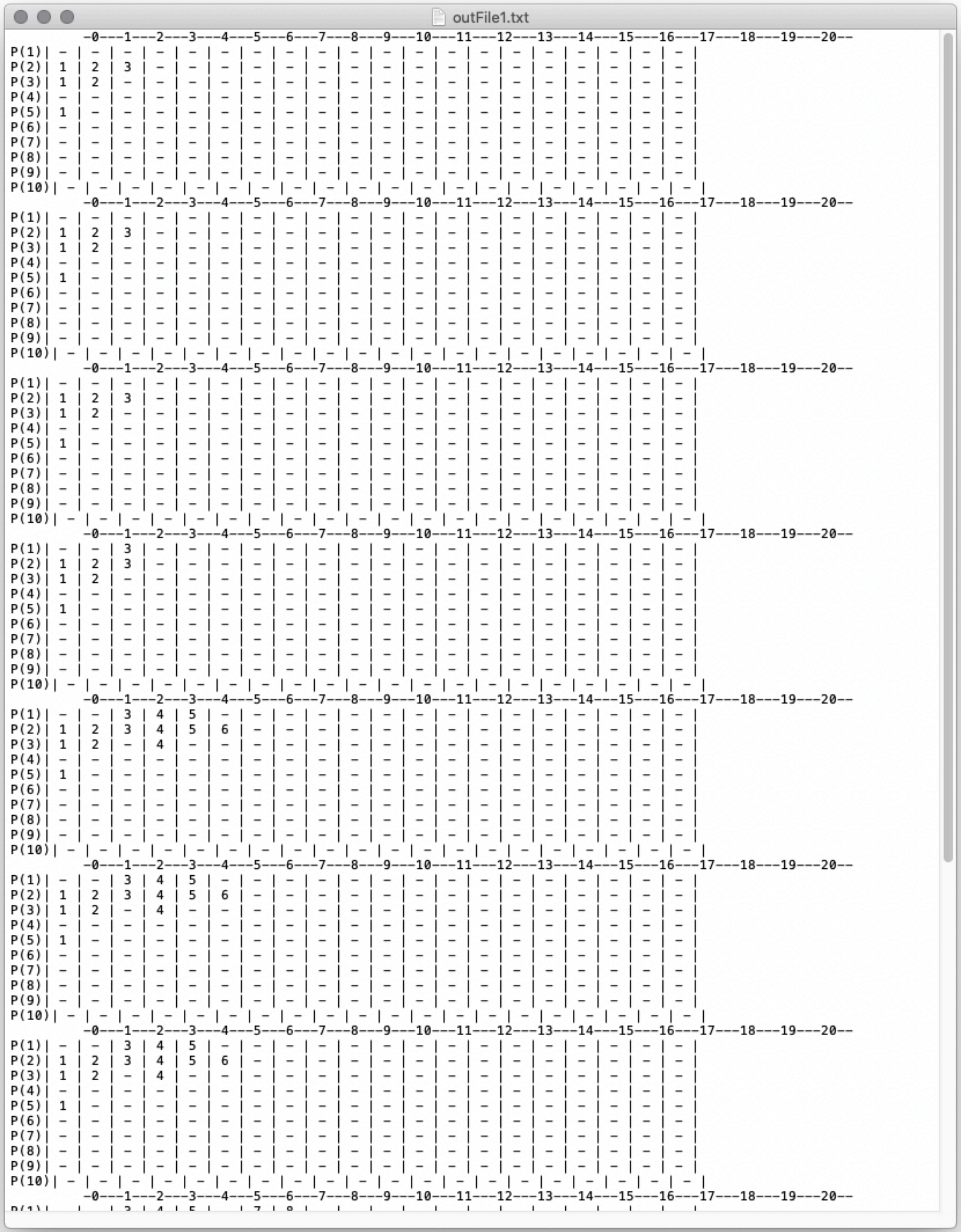


```
         -0---1---2---3---4---5---6---7---8---9---10---11---12---13---14---15---16---17---18---19---20--
P(1)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(2)| 1 | 2 | 3 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(3)| 1 | 2 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
         -0---1---2---3---4---5---6---7---8---9---10---11---12---13---14---15---16---17---18---19---20--
P(1)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(2)| 1 | 2 | 3 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(3)| 1 | 2 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
         -0---1---2---3---4---5---6---7---8---9---10---11---12---13---14---15---16---17---18---19---20--
P(1)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(2)| 1 | 2 | 3 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(3)| 1 | 2 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
         -0---1---2---3---4---5---6---7---8---9---10---11---12---13---14---15---16---17---18---19---20--
P(1)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(2)| 1 | 2 | 3 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(3)| 1 | 2 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
         -0---1---2---3---4---5---6---7---8---9---10---11---12---13---14---15---16---17---18---19---20--
P(1)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(2)| 1 | 2 | 3 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(3)| 1 | 2 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
         -0---1---2---3---4---5---6---7---8---9---10---11---12---13---14---15---16---17---18---19---20--
P(1)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(2)| 1 | 2 | 3 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(3)| 1 | 2 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
         -0---1---2---3---4---5---6---7---8---9---10---11---12---13---14---15---16---17---18---19---20--
P(1)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(2)| 1 | 2 | 3 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(3)| 1 | 2 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
         -0---1---2---3---4---5---6---7---8---9---10---11---12---13---14---15---16---17---18---19---20--
P(1)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(2)| 1 | 2 | 3 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(3)| 1 | 2 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
         -0---1---2---3---4---5---6---7---8---9---10---11---12---13---14---15---16---17---18---19---20--
P(1)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(2)| 1 | 2 | 3 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(3)| 1 | 2 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
         -0---1---2---3---4---5---6---7---8---9---10---11---12---13---14---15---16---17---18---19---20--
P(1)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(2)| 1 | 2 | 3 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(3)| 1 | 2 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
         -0---1---2---3---4---5---6---7---8---9---10---11---12---13---14---15---16---17---18---19---20--
P(1)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(2)| 1 | 2 | 3 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(3)| 1 | 2 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
         -0---1---2---3---4---5---6---7---8---9---10---11---12---13---14---15---16---17---18---19---20--
P(1)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(2)| 1 | 2 | 3 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(3)| 1 | 2 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
         -0---1---2---3---4---5---6---7---8---9---10---11---12---13---14---15---16---17---18---19---20--
P(1)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(2)| 1 | 2 | 3 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(3)| 1 | 2 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
         -0---1---2---3---4---5---6---7---8---9---10---11---12---13---14---15---16---17---18---19---20--
P(1)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(2)| 1 | 2 | 3 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(3)| 1 | 2 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
         -0---1---2---3---4---5---6---7---8---9---10---11---12---13---14---15---16---17---18---19---20--
P(1)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(2)| 1 | 2 | 3 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(3)| 1 | 2 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
         -0---1---2---3---4---5---6---7---8---9---10---11---12---13---14---15---16---17---18---19---20--
P(1)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(2)| 1 | 2 | 3 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(3)| 1 | 2 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
         -0---1---2---3---4---5---6---7---8---9---10---11---12---13---14---15---16---17---18---19---20--
P(1)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(2)| 1 | 2 | 3 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(3)| 1 | 2 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
         -0---1---2---3---4---5---6---7---8---9---10---11---12---13---14---15---16---17---18---19---20--
P(1)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(2)| 1 | 2 | 3 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(3)| 1 | 2 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
         -0---1---2---3---4---5---6---7---8---9---10---11---12---13---14---15---16---17---18---19---20--
P(1)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(2)| 1 | 2 | 3 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(3)| 1 | 2 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
         -0---1---2---3---4---5---6---7---8---9---10---11---12---13---14---15---16---17---18---19---20--
P(1)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(2)| 1 | 2 | 3 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(3)| 1 | 2 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
         -0---1---2---3---4---5---6---7---8---9---10---11---12---13---14---15---16---17---18---19---20--
```

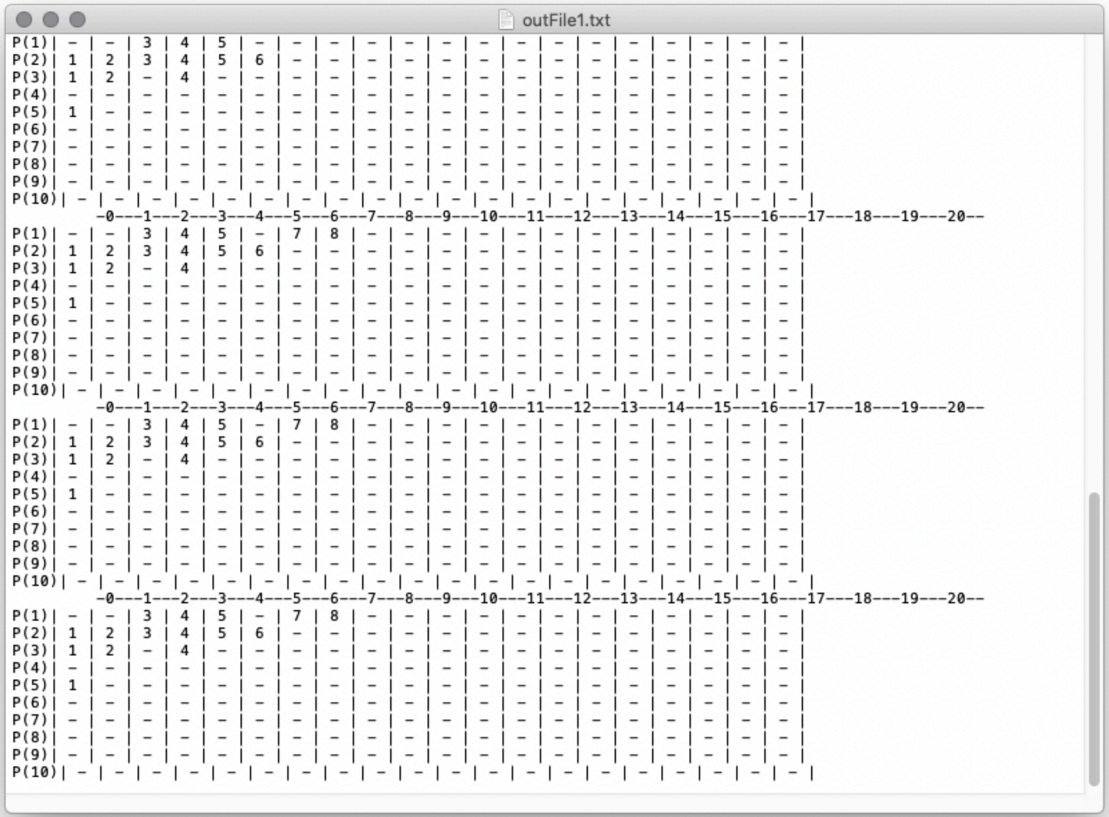


```
head-> (jobId:5, dependetCount: 3, next.jobId:10) ->(jobId:10, dependetCount: 3, next.jobId:1) ->(jobId:1, dependetCount: 2, next.jobId:2) ->(jobId:2, dependetCount: 2, next.jobId:4) ->(jobId:4, dependetCount: 2, next.jobId:NULL) -> NULL
head-> (jobId:2, dependetCount: 2, next.jobId:4) ->(jobId:4, dependetCount: 2, next.jobId:NULL) -> NULL
head-> (jobId:2, dependetCount: 2, next.jobId:4) ->(jobId:4, dependetCount: 2, next.jobId:NULL) -> NULL
head-> (jobId:2, dependetCount: 2, next.jobId:4) ->(jobId:4, dependetCount: 2, next.jobId:3) ->(jobId:3, dependetCount: 2, next.jobId:NULL) -> NULL
head-> (jobId:2, dependetCount: 2, next.jobId:4) ->(jobId:4, dependetCount: 2, next.jobId:3) ->(jobId:3, dependetCount: 2, next.jobId:9) ->(jobId:9, dependetCount: 1, next.jobId:NULL) -> NULL
head-> (jobId:2, dependetCount: 2, next.jobId:4) ->(jobId:4, dependetCount: 2, next.jobId:3) ->(jobId:3, dependetCount: 2, next.jobId:9) ->(jobId:9, dependetCount: 1, next.jobId:NULL) -> NULL
head-> (jobId:2, dependetCount: 2, next.jobId:4) ->(jobId:4, dependetCount: 2, next.jobId:3) ->(jobId:3, dependetCount: 2, next.jobId:9) ->(jobId:9, dependetCount: 1, next.jobId:NULL) -> NULL
head-> (jobId:2, dependetCount: 2, next.jobId:4) ->(jobId:4, dependetCount: 2, next.jobId:3) ->(jobId:3, dependetCount: 2, next.jobId:9) ->(jobId:9, dependetCount: 1, next.jobId:NULL) -> NULL
head-> (jobId:2, dependetCount: 2, next.jobId:4) ->(jobId:4, dependetCount: 2, next.jobId:3) ->(jobId:3, dependetCount: 2, next.jobId:9) ->(jobId:9, dependetCount: 1, next.jobId:NULL) -> NULL
head-> (jobId:2, dependetCount: 2, next.jobId:4) ->(jobId:4, dependetCount: 2, next.jobId:3) ->(jobId:3, dependetCount: 2, next.jobId:9) ->(jobId:9, dependetCount: 1, next.jobId:NULL) -> NULL
head-> (jobId:2, dependetCount: 2, next.jobId:4) ->(jobId:4, dependetCount: 2, next.jobId:3) ->(jobId:3, dependetCount: 2, next.jobId:9) ->(jobId:9, dependetCount: 1, next.jobId:NULL) -> NULL
head-> (jobId:2, dependetCount: 2, next.jobId:4) ->(jobId:4, dependetCount: 2, next.jobId:3) ->(jobId:3, dependetCount: 2, next.jobId:9) ->(jobId:9, dependetCount: 1, next.jobId:NULL) -> NULL
head-> (jobId:2, dependetCount: 2, next.jobId:4) ->(jobId:4, dependetCount: 2, next.jobId:3) ->(jobId:3, dependetCount: 2, next.jobId:9) ->(jobId:9, dependetCount: 1, next.jobId:NULL) -> NULL
head-> (jobId:2, dependetCount: 2, next.jobId:4) ->(jobId:4, dependetCount: 2, next.jobId:3) ->(jobId:3, dependetCount: 2, next.jobId:9) ->(jobId:9, dependetCount: 1, next.jobId:NULL) -> NULL
head-> (jobId:2, dependetCount: 2, next.jobId:4) ->(jobId:4, dependetCount: 2, next.jobId:3) ->(jobId:3, dependetCount: 2, next.jobId:9) ->(jobId:9, dependetCount: 1, next.jobId:NULL) -> NULL
head-> (jobId:2, dependetCount: 2, next.jobId:4) ->(jobId:4, dependetCount: 2, next.jobId:3) ->(jobId:3, dependetCount: 2, next.jobId:9) ->(jobId:9, dependetCount: 1, next.jobId:NULL) -> NULL
head-> (jobId:2, dependetCount: 2, next.jobId:4) ->(jobId:4, dependetCount: 2, next.jobId:3) ->(jobId:3, dependetCount: 2, next.jobId:9) ->(jobId:9, dependetCount: 1, next.jobId:NULL) -> NULL
head-> (jobId:2, dependetCount: 2, next.jobId:4) ->(jobId:4, dependetCount: 2, next.jobId:3) ->(jobId:3, dependetCount: 2, next.jobId:9) ->(jobId:9, dependetCount: 1, next.jobId:NULL) -> NULL
head-> (jobId:2, dependetCount: 2, next.jobId:4) ->(jobId:4, dependetCount: 2, next.jobId:3) ->(jobId:3, dependetCount: 2, next.jobId:9) ->(jobId:9, dependetCount: 1, next.jobId:NULL) -> NULL
head-> (jobId:2, dependetCount: 2, next.jobId:4) ->(jobId:4, dependetCount: 2, next.jobId:3) ->(jobId:3, dependetCount: 2, next.jobId:9) ->(jobId:9, dependetCount: 1, next.jobId:NULL) -> NULL
```

Data2 with numNodes+3 [13]

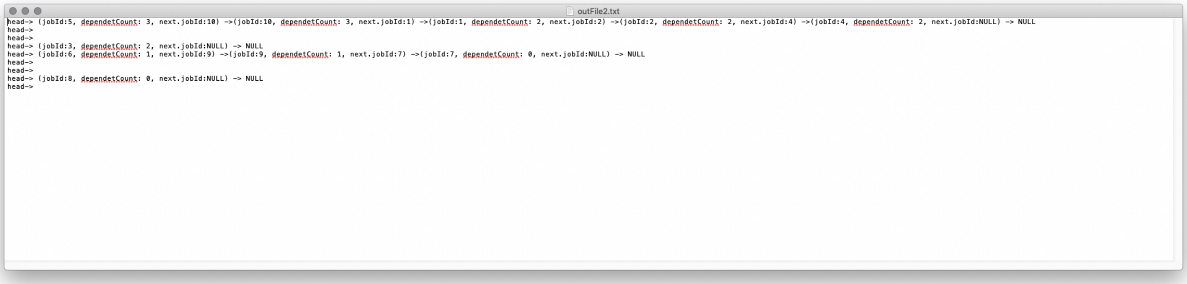


```
     -0---1---2---3---4---5---6---7---8---9---10---11---12---13---14---15---16---17---18---19---20--
P(1)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(2)| 1 | 2 | 3 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(3)| 1 | 2 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(4)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(5)| 1 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(6)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(7)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(8)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(9)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(10)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
     -0---1---2---3---4---5---6---7---8---9---10---11---12---13---14---15---16---17---18---19---20--
P(1)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(2)| 1 | 2 | 3 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(3)| 1 | 2 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(4)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(5)| 1 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(6)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(7)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(8)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(9)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(10)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
     -0---1---2---3---4---5---6---7---8---9---10---11---12---13---14---15---16---17---18---19---20--
P(1)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(2)| 1 | 2 | 3 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(3)| 1 | 2 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(4)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(5)| 1 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(6)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(7)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(8)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(9)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(10)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
     -0---1---2---3---4---5---6---7---8---9---10---11---12---13---14---15---16---17---18---19---20--
P(1)| - | - | 3 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(2)| 1 | 2 | 3 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(3)| 1 | 2 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(4)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(5)| 1 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(6)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(7)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(8)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(9)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(10)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
     -0---1---2---3---4---5---6---7---8---9---10---11---12---13---14---15---16---17---18---19---20--
P(1)| - | - | 3 | 4 | 5 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(2)| 1 | 2 | 3 | 4 | 5 | 6 | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(3)| 1 | 2 | - | 4 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(4)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(5)| 1 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(6)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(7)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(8)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(9)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(10)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
     -0---1---2---3---4---5---6---7---8---9---10---11---12---13---14---15---16---17---18---19---20--
P(1)| - | - | 3 | 4 | 5 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(2)| 1 | 2 | 3 | 4 | 5 | 6 | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(3)| 1 | 2 | - | 4 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(4)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(5)| 1 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(6)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(7)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(8)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(9)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(10)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
     -0---1---2---3---4---5---6---7---8---9---10---11---12---13---14---15---16---17---18---19---20--
P(1)| - | - | 3 | 4 | 5 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(2)| 1 | 2 | 3 | 4 | 5 | 6 | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(3)| 1 | 2 | - | 4 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(4)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(5)| 1 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(6)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(7)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(8)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(9)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(10)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
     -0---1---2---3---4---5---6---7---8---9---10---11---12---13---14---15---16---17---18---19---20--
P(1)|   |   | 3 | 4 | 5 |   | 7 | 8 |   |   |   |   |   |   |   |   |   |   |   |   |
```



outFile1.txt

```
P(1)| - | - | 3 | 4 | 5 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(2)| 1 | 2 | 3 | 4 | 5 | 6 | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(3)| 1 | 2 | - | 4 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(4)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(5)| 1 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(6)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(7)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(8)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(9)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(10)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
         -0---1---2---3---4---5---6---7---8---9---10---11---12---13---14---15---16---17---18---19---20--
P(1)| - | - | 3 | 4 | 5 | - | 7 | 8 | - | - | - | - | - | - | - | - | - | - | - | - |
P(2)| 1 | 2 | 3 | 4 | 5 | 6 | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(3)| 1 | 2 | - | 4 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(4)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(5)| 1 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(6)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(7)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(8)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(9)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(10)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
         -0---1---2---3---4---5---6---7---8---9---10---11---12---13---14---15---16---17---18---19---20--
P(1)| - | - | 3 | 4 | 5 | - | 7 | 8 | - | - | - | - | - | - | - | - | - | - | - | - |
P(2)| 1 | 2 | 3 | 4 | 5 | 6 | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(3)| 1 | 2 | - | 4 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(4)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(5)| 1 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(6)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(7)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(8)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(9)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(10)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
         -0---1---2---3---4---5---6---7---8---9---10---11---12---13---14---15---16---17---18---19---20--
P(1)| - | - | 3 | 4 | 5 | - | 7 | 8 | - | - | - | - | - | - | - | - | - | - | - | - |
P(2)| 1 | 2 | 3 | 4 | 5 | 6 | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(3)| 1 | 2 | - | 4 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(4)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(5)| 1 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(6)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(7)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(8)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(9)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
P(10)| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
```



outFile2.txt

```
head-> (jobId:5, dependetCount: 3, next.jobId:10) ->(jobId:10, dependetCount: 3, next.jobId:1) ->(jobId:1, dependetCount: 2, next.jobId:2) ->(jobId:2, dependetCount: 2, next.jobId:4) ->(jobId:4, dependetCount: 2, next.jobId:NULL) -> NULL
head->
head->
head-> (jobId:3, dependetCount: 2, next.jobId:NULL) -> NULL
head-> (jobId:6, dependetCount: 1, next.jobId:9) ->(jobId:9, dependetCount: 1, next.jobId:7) ->(jobId:7, dependetCount: 0, next.jobId:NULL) -> NULL
head->
head->
head-> (jobId:8, dependetCount: 0, next.jobId:NULL) -> NULL
head->
```