**CSCI 323-33 Scheduling (C++)**

Han Wen Loh

Due date: Nov. 27, 2018

Algorithm steps:

step 0: inFile1 <-- open input-1 file from argv

inFile2 <-- open input-2 file from argv

outFile <-- open output file from argv

numNodes <-- get from inFile1.

loadMatrix (inFile1)

totalJobTimes <-- computeTotalJobTimes (inFile2)

procGiven <-- from console, ask user how many processors are needed

if (procGiven <= 0) returns with error message

if (procGiven > numNodes)

procGiven <-- numNodes

// There is no need to have more processors than the number of nodes.

- dynamically allocate all 1-D and 2-D arrays and initializing all arrays.

- procUsed <-- 0 // the actual number of processors used is 0 at the beginning.

- currentTime <-- 0

step 1: orphenNode <-- getUnMarkOrphen()

jobMarked[orphenNode] <-- 1

newNode <-- create a new node for orphenNode with its jobId and jobTime

// newNode has two fields, jobId and jobTime

// jobTime can be found in jobTime[] array

Insert2Open(newNode)

step 2: repeat step 1 until no more orphenNode in the graph,

// orphenNode is -1.

step 3: printList(OPEN)

step 4: availProc <-- findProcessor( )

if availProc > 0 // there is a processor available

ProcUsed ++

newJob <-- remove from the front of OPEN list

processJob[availProc]<-- newJob’s jobId

processTime[availProc] <-- newJob’s jobTime

updateTable(availProc, currentTime, newJob)

step 5: repeat 4 while OPEN is not empty \*and\* ProcUsed < ProcGiven

step 6: if checkCycle ( ) >= 1

output error message and exit (there is cycle in the graph)

step 7: printTable()

step 8: currentTime++

step 9: Decrease all processTime[i] by 1

step 10: job <-- findDoneJob()

// find a job that is done, ie., processTIME [i] == 0 ;

// findDoneJob also deletes the job from the processJob[i]

// (set processJob[i] to 0)

Step 11 deleteNode(job)

deleteEdge(job)

step 12: repeat step 10 – step 11 until no more finished job

step 13: debugging print the following to outFile2

with readable headings for each:

- currentTime,

- jobMarked[ ] 1D array

- processTime[ ] 1D array

- processJob[ ] 1D array

- jobDone [ ] 1D array

step 14: repeat step 1 to step 13 until the graph is empty

(i.e., jobDone[i] are == 1. )

step 15: printTable()

step 16: close all files