**Class Structures**

|  |
| --- |
| Node  Task Weight: int  Incoming nodes: node <>  Outgoing nodes: node <>  incoming edges  outgoing edges |
| Processor  Processor number  Node list  Busy until: int |
| Edge  Edge weight: int  From-node: node  To-node: node |
| Optimal Algorithm |

**Single Core Algorithm (DFS)**

Given

* G(n,e)

UnSchNodes:list = All nodes

AvailNodes: list = currently schedulable nodes that has no unshedules parent node

ParentNode: Node = arbitrary startNode

ProcessorList: List = list of all processors

* Put all available nodes in AvailNode
* Pick one node ni from Available Nodes
* Schedule ni to the processor so that ni finishes the earliest
* Add next available nodes into AvailNodes
* ParentNode = ni

While(AvailableNodes is not empty)

{

* In AvailNodes, pick one node ni with ni.parent == ParentNode && has the biggest incoming edge cost out of all other available nodes nj with nj.parent = ParentNode
  + If there is no available node with ni.parent = ParentNode, set ParentNode = ParentNode.parent. And this step again until there is one
* Schedule ni to the processor ProcessorLIst[k] so that ni finishes the earliest
* Remove ni from UnSchNodes and AvailNodes
* Add next available nodes into AvailNodes
* Set ni as the parent node

}

Return ProcessorList

**Parallelized Multi Core Algorithm (NOT COMPLETED YET)**

Given

* G(n,e)

UnSchNodes:list = All nodes

AvailNodes: list = currently schedulable nodes that has no unshedules parent node

ParentNode: Node = arbitrary startNode

ProcessorList: List = list of all processors

ThreadList: list of threads with length tNum

tNum: number of threads

* Put all available nodes in AvailNode
* Pick tNum node ni from Available Nodes and assign it to each thread

For each thread:

* Schedule ni to the processor so that ni finishes the earliest
* Add next available nodes into AvailNodes
* ParentNode = ni

For each thread:

While(AvailableNodes is not empty)

{

* In AvailNodes, pick one node ni with ni.parent == ParentNode && has the biggest incoming edge cost out of all other available nodes nj with nj.parent = ParentNode
  + If there is no available node with ni.parent = ParentNode, set ParentNode = ParentNode.parent. And this step again until there is one
* Schedule ni to the processor ProcessorLIst[k] so that ni finishes the earliest
* Remove ni from UnSchNodes and AvailNodes
* Add next available nodes into AvailNodes
* Set ni as the parent node

}

Return ProcessorList