Generic List Scheduling

Algorithm List Scheduling
Input:
    Network DAG of Tasks to be Completed
Output:
    Task sequence \( I = T_1, T_2, \ldots, T_k \)
Algorithm:
    PriorityList PL list of nodes with no unscheduled predecessors
    for each task \( T_i \), let \( \text{Pred}(T_i) \) be the number of predecessors
    for each \( i \)
        if \( \text{Pred}(T_i) = 0 \) then PL.Insert(\( T_i \))
    while (! PL.IsEmpty())
        CurrTask = PL.Delete()
        DoSchedule(CurrTask)
        for each successor \( j \) of CurrTask
            Pred(\( T_j \)) = Pred(\( T_j \)) - 1
            if \( \text{Pred}(T_j) == 0 \) then PL.Insert(\( T_j \))
Algorithm Time-Ordered List Scheduling

Input:
Network DAG of Tasks to be Completed

Output:
Time sequence \( S = S_1, S_2, ..., S_k \)

Algorithm:
PriorityList PL list of nodes with no unscheduled predecessors
for each task \( T_i \), let \( \text{Pred}(T_i) \) be the number of predecessors
for each \( i \)
if \( \text{Pred}(T_i) == 0 \) then PL.Insert(\( T_i \))

CurrTime = 1
while (! PL.IsEmpty())

CurrTask = PL.GetNext(CurrTime) find operation ready at CurrTime
if (CurrTask == NULL) then CurrTime++ instruction filled
else
if DoScheduleAtCurrTime(CurrTask,CurrTime)
    PL.Delete(CurrTask)
    for each successor \( j \) of CurrTask,
        \( \text{Pred}(T_j) = \text{Pred}(T_j) - 1 \)
        if \( \text{Pred}(T_j) == 0 \) then PL.Insert(\( T_j \))
    else CurrTask cannot be scheduled at CurrTime
    PL.Insert(CurrTask)