Critical Path Method

```
line = list() #contains a single line
singleElement = list()
tasks = dict() #contains all the tasks
number = -1
fhand = open('cpm.txt')
#task id, task name, duration, dependencies
```

```
for line in fhand: #slide the file line by line
    singleElement=(line.split(',')) #split a line in subparts
    number += 1
    for i in range(len(singleElement)): #creating the single task element
        tasks['task'+ str(singleElement[0])]= dict()
        tasks['task'+ str(singleElement[0])]['id'] = singleElement[0]
        tasks['task'+ str(singleElement[0])]['name'] = singleElement[1]
        tasks['task'+ str(singleElement[0])]['duration'] = singleElement[2]
        if(singleElement[3] != "\n"):
            tasks['task'+ str(singleElement[0])]['dependencies'] = singleElemen
        else:
            tasks['task'+ str(singleElement[0])]['dependencies'] = ['-1']
        tasks['task'+ str(singleElement[0])]['ES'] = 0
        tasks['task'+ str(singleElement[0])]['EF'] = 0
        tasks['task'+ str(singleElement[0])]['LS'] = 0
        tasks['task'+ str(singleElement[0])]['LF'] = 0
        tasks['task'+ str(singleElement[0])]['float'] = 0
        tasks['task'+ str(singleElement[0])]['isCritical'] = False
```

Forward Pass

```
for taskFW in tasks: #slides all the tasks
    if('-1' in tasks[taskFW]['dependencies']): #checks if it's the first task
        tasks[taskFW]['ES'] = 1
        tasks[taskFW]['EF'] = (tasks[taskFW]['duration'])
    else: #not the first task
        for k in tasks.keys():
            for depend in tasks[k]['dependencies']: #slides all the dependency
                #print('task ' + taskFW + ' k '+ k + ' depend ' +depend)
                if(depend != '-1' and len(tasks[k]['dependencies']) == 1): #if
                    tasks[k]['ES'] = int(tasks['task'+ str(depend)]['EF']) +1
                    tasks[k]['EF'] = int(tasks[k]['ES']) + int(tasks[k]['durati
                elif(depend !='-1'): #if the task k has more dependency
                    if(int(tasks['task'+depend]['EF']) > int(tasks[k]['ES'])):
                        tasks[k]['ES'] = int(tasks['task'+ depend]['EF']) +1
                        tasks[k]['EF'] = int(tasks[k]['ES']) + int(tasks[k]['du
```

```
aList = list() #list of task keys
```

```
for element in tasks.keys():
    aList.append(element)

bList = list() #reversed list of task keys
while len(aList) > 0:
    bList.append(aList.pop())
```

Backward Pass

```
for taskBW in bList:
    if(bList.index(taskBW) == 0): #check if it's the last task (so no more task
        tasks[taskBW]['LF']=tasks[taskBW]['EF']
        tasks[taskBW]['LS']=tasks[taskBW]['ES']

    for depend in tasks[taskBW]['dependencies']: #slides all the dependency in a
        if(depend != '-1'): #check if it's NOT the last task
            if(tasks['task'+ depend]['LF'] == 0): #check if the the dependency i
                #print('ID depend: '+str(tasks['task'+depend]['id']) + ' taskBW
                tasks['task'+ depend]['LF'] = int(tasks[taskBW]['LS']) -1
                tasks['task'+ depend]['LS'] = int(tasks['task'+ depend]['LF'])
                tasks['task'+ depend]['float'] = int(tasks['task'+ depend]['LF'
                #print('IF1 dip LS: '+str(tasks['task'+depend]['LS']) +' dip LF
            if(int(tasks['task'+ depend]['LF']) >int(tasks[taskBW]['LS']) ): #pu
                tasks['task'+ depend]['LF'] = int(tasks[taskBW]['LS']) -1
                tasks['task'+ depend]['LS'] = int(tasks['task'+ depend]['LF'])
                tasks['task'+ depend]['float'] = int(tasks['task'+ depend]['LF'
                #print('IF2 dip LS: '+str(tasks['task'+depend]['LS']) +' dip LF
```

Critical Path

```
int('task id, task name, duration, ES, EF, LS, LF, float, isCritical')
r task in tasks:
  if(tasks[task]['float'] == 0):
      tasks[task]['isCritical'] = True
  print(str(tasks[task]['id']) +', '+str(tasks[task]['name']) +', '+str(tasks[ta
```

```
task id, task name, duration, ES, EF, LS, LF, float, isCritical
1, A, 12, 1, 12, 1, 12, 0, True
2, B, 6, 13, 18, 31, 36, 18, False
3, E, 12, 13, 24, 19, 30, 6, False
4, F, 18, 13, 30, 13, 30, 0, True
5, C, 2, 19, 20, 37, 38, 18, False
6, G, 10, 31, 40, 31, 40, 0, True
7, I, 8, 31, 38, 37, 44, 6, False
8, D, 8, 21, 28, 39, 46, 18, False
9, H, 6, 41, 46, 41, 46, 0, True
10, J, 2, 39, 40, 45, 46, 6, False
11, K, 8, 47, 54, 47, 54, 0, True
```