Question 1. Enumerate all feasible solutions and find the best one. Try the data sizes 8, 10, 11, 12, ... that you can solve for.

In [18]:

```
import pandas as pd
from itertools import permutations
import time

data = pd.read_excel("./sum of completion times.xlsx",
                     engine='openpyxl', index_col=0, header=None)

#if size==12: memory error
size = 10
jobs = []
start = time.time()

for i in range(size):
    jobs.append(list(data[i+1]))

pers = permutations(jobs, size)

orders = []
start = time.time()
sum_times = []

for per in pers:

    order = []
    order_time = 0
    sum_time = 0

    for i in range(size):
        order.append(per[i][0])
        if order_time >= per[i][2]:
            order_time += per[i][1]
        else:
            order_time = per[i][2]
            order_time += per[i][1]

        sum_time += order_time

    orders.append(order)
    sum_times.append(sum_time)

min_time, min_time_index = min((val, idx)
                               for (idx, val) in enumerate(sum_times))
end = time.time()

print('最短sum of completion time:', min_time)
print('一個最佳job順序:', orders[min_time_index])
print('time_cost:', end-start)
```

```
最短sum of completion time: 696
一個最佳job順序: [1, 3, 4, 2, 5, 6, 7, 8, 9, 10]
time_cost: 10.660767316818237
```

Question 2. Apply the SRPT (shortest remaining processing time first) algorithm to find an optimal

solution value of the preemptive version. Compare the solution values with the optimal values of Question 1.

In [20]:

```python
import pandas as pd

data = pd.read_excel("./sum of completion times.xlsx",
                     engine='openpyxl', index_col=0, header=None)


size = 50
jobs = []

for i in range(size):
    jobs.append(list(data[i+1]))


init = True
init_job = [None, float('Inf'), 0]
curr_time = 0
next_stop_time = 0
arrived_jobs = []
have_curr = True
sum_of_completion_times = 0

while len(jobs) or len(arrived_jobs) or have_curr:
    if init:
        curr_job = init_job
        init = False

    next_stop_time = min(curr_time + curr_job[1],
                         min([jobs[i][2] for i in range(len(jobs))],
                             default = float('Inf')))
    curr_job[1] = curr_job[1] - (next_stop_time - curr_time)


    # current job finish
    if curr_job[1] <= 0:
        curr_job[1] = 0
        if curr_job[0] != None:
            print('job', curr_job[0],'finish','time=',next_stop_time)
            sum_of_completion_times += next_stop_time
            have_curr = False
            curr_job = init_job

    remove_jobs = []
    for job in jobs:
        if job[2] == next_stop_time:
            arrived_jobs.append(job)
            remove_jobs.append(job)
    for job in remove_jobs:
        jobs.remove(job)

    min_p = curr_job[1]
    for job in arrived_jobs:
        if job[1] < min_p:
            temp_job = job
            min_p = job[1]

    if min_p != curr_job[1]:
        if curr_job[0] != None:
            arrived_jobs.append(curr_job)
        arrived_jobs.remove(temp_job)
        curr_job = temp_job
        have_curr = True

    curr_time = next_stop_time

print('sum of completion times:', sum_of_completion_times)
```

```
job 1 finish time= 11
job 3 finish time= 20
job 4 finish time= 27
job 2 finish time= 43
job 5 finish time= 65
job 6 finish time= 79
job 7 finish time= 91
job 8 finish time= 104
job 9 finish time= 124
job 10 finish time= 132
job 11 finish time= 145
job 15 finish time= 154
job 16 finish time= 164
job 17 finish time= 172
job 19 finish time= 182
job 12 finish time= 194
job 13 finish time= 207
job 18 finish time= 220
job 22 finish time= 228
job 14 finish time= 242
job 20 finish time= 256
job 24 finish time= 273
job 25 finish time= 284
job 26 finish time= 305
job 27 finish time= 318
job 28 finish time= 331
job 30 finish time= 346
job 29 finish time= 364
job 32 finish time= 372
job 34 finish time= 383
job 33 finish time= 396
job 35 finish time= 405
job 31 finish time= 421
job 37 finish time= 436
job 38 finish time= 442
job 39 finish time= 455
job 36 finish time= 472
job 40 finish time= 485
job 41 finish time= 503
job 42 finish time= 511
job 44 finish time= 522
job 43 finish time= 536
job 21 finish time= 556
job 47 finish time= 575
job 48 finish time= 586
job 46 finish time= 602
job 49 finish time= 623
job 50 finish time= 639
job 23 finish time= 661
job 45 finish time= 684
sum of completion times: 16346
```

Question 3. Develop a branch-and-bound algorithm to improve the problem-solving process of Question 1. You may use Breadth FS, Depth FS, Best FS, or other possible strategies for your design.

DFS 無法跑完50個但在相同jobs數量下比第一題快

jobs_num = 10，Q1:10sec 、DFS:6sec

In [19]:

```
import pandas as pd
import time

data = pd.read_excel("./sum of completion times.xlsx",
                     engine='openpyxl', index_col=0, header=None)

start = time.time()
size = 10

jobs = []
"""
    jobs[i][0]:job name
    jobs[i][1]:process time
    jobs[i][2]:arrive time
"""
for i in range(size):
    jobs.append(list(data[i+1]))


best_cost = float('Inf')

def calCost(per):
    order_time = 0
    sum_time = 0
    for i in range(len(per)):
        if order_time >= per[i][2]:
            order_time += per[i][1]
        else:
            order_time = per[i][2]
            order_time += per[i][1]
        sum_time += order_time
    return sum_time


def perm(n,begin,end):
    global best_cost
    global bestjob

    if begin>=end and calCost(n)<best_cost:
        best_cost = calCost(n)

    else:
        i=begin
        for num in range(begin,end):
            n[num],n[i]=n[i],n[num]
            perm(n,begin+1,end)
            n[num],n[i]=n[i],n[num]


perm(jobs, 0, len(jobs))
end = time.time()
print('最短sum of completion time:',best_cost)
print('time_cost:', end-start)
```

```
最短sum of completion time: 696
time_cost: 6.425884962081909
```

non-preemptive SJF,可50個jobs都跑完,not branch and bound

In [14]:

```
import pandas as pd

data = pd.read_excel("./sum of completion times.xlsx",
                     engine='openpyxl', index_col=0, header=None)

size = 50
jobs = []

for i in range(size):
    jobs.append(list(data[i+1]))

curr_time = -1
next_stop_time = 0
done_jobs = []
count = 0
arrived_jobs = []
sum_of_completion_times = 0

while count < len(jobs):
    finish_job = None
    for job in jobs:
        if job[2] > curr_time and job[2] <= next_stop_time:
            arrived_jobs.append(job)

    min_p = float('Inf')
    for arrived_job in arrived_jobs:
        if arrived_job[1] < min_p:
            min_p = arrived_job[1]
            finish_job = arrived_job

    if finish_job != None:
        arrived_jobs.remove(finish_job)
        curr_time = next_stop_time
        next_stop_time = next_stop_time + min_p
        print('job', finish_job[0], '完成時間:', next_stop_time)
        sum_of_completion_times += next_stop_time

        count += 1

    else:
        next_stop_time += 1

print('sum of completion times:', sum_of_completion_times)
```

```
job 1 完成時間: 11
job 3 完成時間: 20
job 4 完成時間: 27
job 2 完成時間: 43
job 5 完成時間: 65
job 6 完成時間: 79
job 7 完成時間: 91
job 8 完成時間: 104
job 9 完成時間: 124
job 10 完成時間: 132
job 11 完成時間: 145
job 15 完成時間: 154
job 16 完成時間: 164
job 17 完成時間: 172
job 19 完成時間: 182
job 12 完成時間: 194
job 13 完成時間: 207
job 18 完成時間: 220
job 22 完成時間: 228
job 14 完成時間: 242
job 20 完成時間: 256
job 24 完成時間: 273
job 25 完成時間: 284
job 26 完成時間: 305
job 27 完成時間: 318
job 28 完成時間: 331
job 29 完成時間: 350
job 30 完成時間: 364
job 31 完成時間: 382
job 32 完成時間: 388
job 35 完成時間: 397
job 34 完成時間: 408
job 33 完成時間: 421
job 37 完成時間: 436
job 38 完成時間: 442
job 36 完成時間: 462
job 39 完成時間: 472
job 40 完成時間: 485
job 41 完成時間: 503
job 21 完成時間: 525
job 42 完成時間: 532
job 44 完成時間: 539
job 43 完成時間: 556
job 47 完成時間: 575
job 46 完成時間: 596
job 48 完成時間: 602
job 49 完成時間: 623
job 50 完成時間: 639
job 23 完成時間: 661
job 45 完成時間: 684
sum of completion times: 16413
```