Project for 430

Student:	Student ID:
Wanglin Song	A20354898
Ruo Yang	A20427384

By assumption, we have m machines and n jobs to schedule, and we use greedy algorithm.
Algorithm:

Step 1: Sort n jobs by deadline with increasing order, such that sequence of jobs is J
Give machine an order, that is sequence M, with m elements. Each elements, s.t M[j] has an end time indicator and Jobs attributes. For beginning, each M[j] has end time indicator with 0, which means available for any job.

Step 2:

```
For i in range(n):
    Job = J[i]
    Start_time = Job's start time
    End_time = Job's deadline
    For j in range(m):
        If Start_time >= M[j].end_time_indicator
            CompatibleMachineList.append(M[j])
    For machine M in CompatilbeMachineList
        target = the machine with smallest waiting time
        Break ties arbitrarily by choosing the machine with smallest ID
    target.Jobs.append(Job)
    target.end_time_indicator = End_time
```

Step 3:

```
Array = []
For j in range(m):
    Array[j] = length(M[j].Jobs)
```

Then, Array will has the information about how many jobs are scheduled for each machines.

Time Complexity:

Step 1: Sort jobs with O(nlog(n))
Step 2: each job at most try m machines, plus at most m attempts to find the machine with earliest end time, so O(n*2m).
Step 3: O(m)
So the time complexity is O(n*Max(log(n), 2m))

Prove of correctness:

By this algorithm, there will be no time conflict of each jobs, since if there is any conflict with current machine, the algorithm will move to next. If conflict with all, then the algorithm will skip the job.

Prove of optimal:

Let n = number of jobs we have evaluated, G(k) is the number of jobs we arranged with greedy algorithm, O(k) is the optimal number of jobs we can arranged.

If n = 1, G(k) = O(k) obviously, they are all 1 if there are one or more machine.

Assume n = k-1, G(k-1) = O(k-1) is true (k>=1), we want to prove when we arranged kth Job (n = k), G(k) = O(k) is still true. There are 2 possible outcomes when arranging kth job:

1) G(k) = G(k-1) + 1 (we have arranged kth job with greedy algorithm): since O(k) can't be smaller than G(k) by definition and can't be greater than O(k-1) + 1 (we only tried to arrange one more job), O(k) = O(k-1) + 1 = G(k-1) + 1 =G(k).
2) G(k) = G(k-1), the question is will O(k) = O(k-1) +1 = G(k-1) + 1>G(k)? We can prove the answer to be "nope" by contradiction.

   Assume there is a machine M, the latest finished job of it (marked as J) is conflicted with k th job by greedy algorithm, but if the latest finished job of M is another arranged job J1 with earlier finishing time, we can put k th job on M (The hypothetical scenario greedy algorithm will fail, but true optimal algorithm succeeds).

   Since we choose the tightest fit schedule (among all the available machine we choose the one with smallest waiting time), the greedy algorithm would have arranged jobs after J1 behind J and kth job behind J1 instead. In this case G(k) = G(k-1) +1 which contradicts with our assumption. Therefore G(k) = G(k-1) = O(k-1) = O(k).

Combining 1) and 2), we know G(k) = O(k), the number of arranged job from greedy algorithm is optimal.