1

1. The best sequence is $(2,1,3,4)$ with an objective value of 333
2. By putting items with more weight towards the beginning, you minimize the values of C_j they will be multiplied by
3. By putting the jobs with shorter processing time towards the beginning, you make it so that the average value of C_j is lower
4. $w_j - p_j = (3,6,2,1)$ giving us order $(2,1,3,4)$.
   $\frac{w_j}{p_j} = (2, 2.2, 1.29, 1.25)$ giving us order $(2,1,3,4)$.
   Although both give us the same order, doing it by decreasing order of $\frac{w_j}{p_j}$ is optimal.

2

1. The best sequence is $(1,2,3,4)$ with an objective value of 6
2. The earlier the due date, the earlier you want to have it done, generally
3. By putting jobs with smaller processing time earlier, you produce less late jobs overall. This doesn't seem like a good metric for reducing $L_{\max}$ though.
4. Sequencing jobs in $d_j p_j$ order is the most suitable generic rule since it factors both due date and processing time.

3

1. The best sequences are $[(1,3,2,4),(1,3,4,2),(2,3,1,4),(2,3,4,1),(2,4,1,3),(2,4,3,1),(3,4,1,2),$ with an objective value of 2
2. Sequence them in decreasing $d_j - p_j$ order

4

1. The best sequences are $[(1,2,4,3)]$ with an objective value of 22
2. Not possible, this is NP hard

5

1. The optimal solution is $(1,7),(2,8),(3,5),(4,6),(9,10,11)$ with an objective value of 15

6

The best sequences are $[(3,2,4,1)]$ with an objective value of 34

7

The best sequences are $[(3,4,2,1)]$ with an objective value of 35

8

The best sequences are $[(3,4,2,1)]$ with an objective value of 35

9

The best schedule does $(3,2,4,1)$ on machine 1 and $(4,1,3,2)$ on machine 2 and the objective value is 30

10

The best schedule does $(2,1,4,3)$ on machine 1 and $(4,3,2,1)$ on machine 2 and the objective value is 30

11

It's easier because you can treat the first case as a special case of the second where you break each job i into p_i different segments with $p=1$ and $r=r_i$

12

Each job i will take a_i to setup, p_i to run, and b_i to break down. Therefore the order doesn't matter and the makespan is $C_{\max} = \sum a_i + b_i + p_i$

13

With $p = 0$, this is just a task of scheduling jobs to reduce the total setup time which is equivalent to TSP

14

Can introduce jobs with $w_j = \infty$ that have r_j equal to when the breakdown occurs and p_j equal to the duration of the breakdown

15

There are n jobs and n slots and each job belongs in a slot and you are therefore assigning each job to minimize a cost function

16

You are assigning each job to one of the m parallel machines and get the same A matrix as you would in the transportation problem

17

LHS: The best case is that all m machines have an equal load and the minimum load is $\frac{j}{m}$ RHS: The worst case is there is

18

For all machines $i \notin M_j, v_{ij} = 0, p_{ij} = \infty$ which makes $Pm|M_j|\gamma$ a special case of $Rm||\gamma$

19

Block means the item must stay in machine 1 until machine 2 is ready and nwt means that machine 2 must be ready right after machine 1. Both will have the same $C_{\max}$ as the conditions are equivalent when there is only one set of 2 chained machines. Can make this $1|s_{jk}|c_{\max}$ by using the following s, $s_{0j} = p_{1j}, s_{k0} = p_{2k}, s_{jk} = \max(p_{2j}, p_{1k})$

20

Om is a relaxation of Fm as you have more options in which order to run the machines. Therefore the objective in Fm will at best be the same as Om

21

If there is one very long job on both machines this will be exceeded.

jobs	1	2	3
p_{1j}	100	10	5
p_{2j}	100	10	5

This would take 200 ro run whn the lower bound is 115

22

(i) -¿ (iii) -¿ (ii) -¿ (iv) —-¿ (v) -¿ (vi)

23

When minimizing $\sum U_j$, it is split into two parts: the first part has a set of jobs that will be completed on time and the second set that will be late.