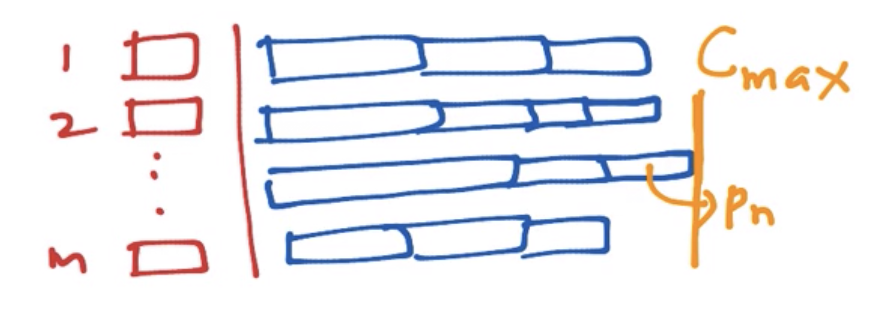
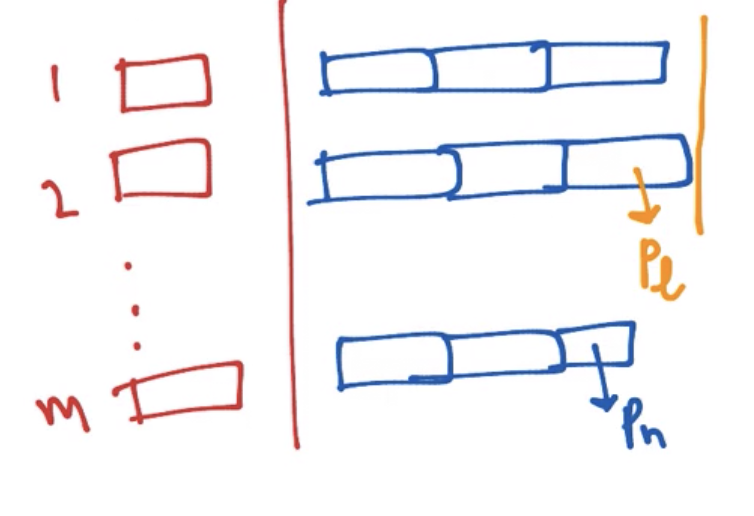
LPT is a 4/3 approx algo

Say there is an instance P1, P2, … Pn for which LPT is not 4/3 of OPT

Say without loss of generality, P1 \ge P2 \ge … \ge Pn. Suppose the schedule after applying LPT looks like this,



The job that defines the makespan i.e. the one that finishes last is actually the job Pn (the one with the smallest processing time. Suppose that was not the case.

**n**

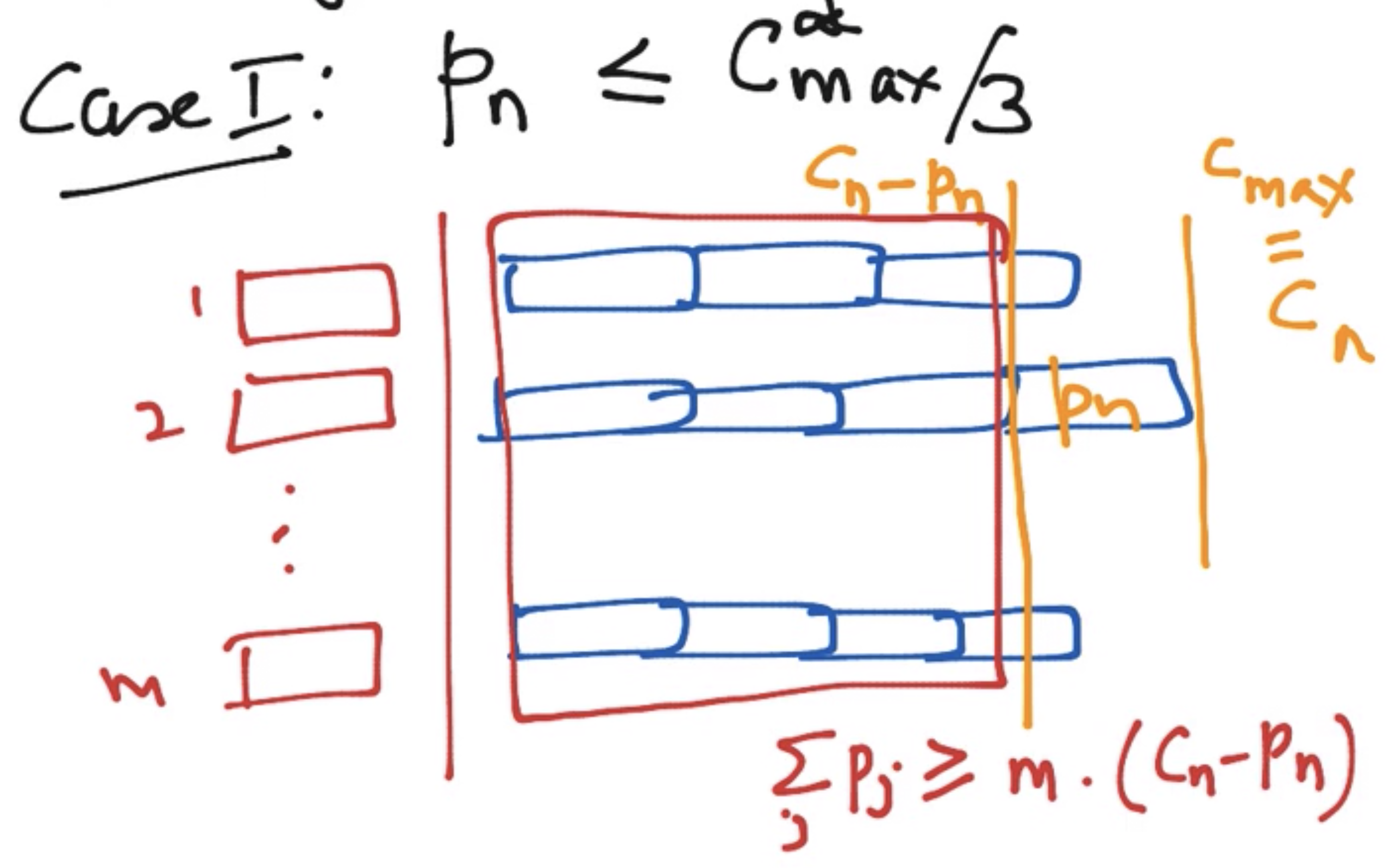
Say some other job L with time PL defined the makespan and PL \ge Pn. Now if we run LPT on jobs 1 to L, we would get the makespan of this subset to be C\_max again, since in the case of n jobs, the C\_max is defined by the L’th job.

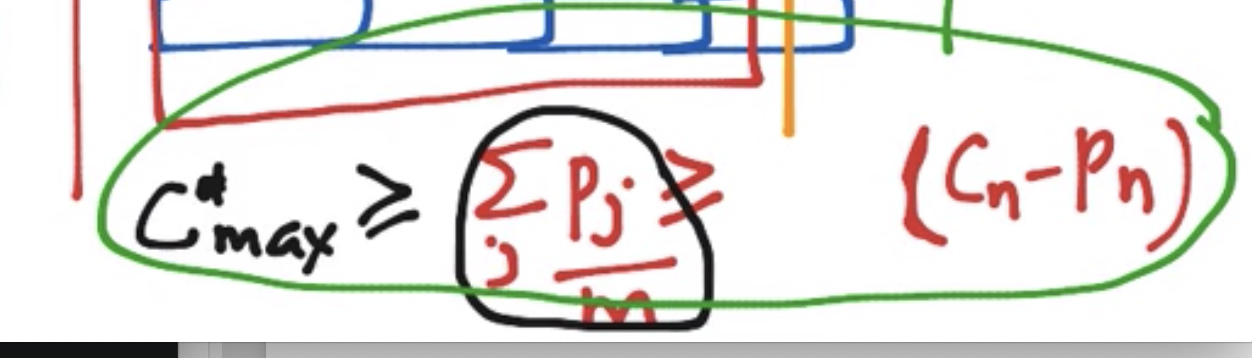
And moreover the OPT solution of the subset of L jobs can only be \le than the OPT solution of n jobs (because we have fewer jobs in the subset).

Now, if the above instant with n jobs is not within a factor of 4/3, we can also say that the approximation ratio of the subset of the first L jobs will not be within a factor of 4/3 as well. In that case we can take the subset P1, P2, … PL too. In this case C\_max would be defined by PL (the last job) in that list, and hence our analysis would be similar to the case when the n’th job is the last one.

Now taking set P1, P2, … Pn, and assuming that Pn defines C\_max, we can consider two cases:

**Case 1:** Length of nth job pn \le C\*\_max/3, where C\*\_max is the makespan of the optimal schedule. Say C\_max = Cn. Now if we exclude the nth job, we can clearly say \sum\_{j} p\_j \ge m\*(C\_n – p\_n) (See picture) and we also know that C\*\_max \ge 1/m\sum\_{j} p\_j.



Hence 

C\*max \ge Cn - pn

And

C\*\_max/3 \ge pn

* Cn \le 4C\*max/3.

**Case 2:** pn \ge C\*\_max/3.

We can show that in this case LPT gives the optimal solution.

Then, T\* < 3pn. Since pn is the smallest processing time this implies that the optimal schedule has at most 2 jobs per machine. (Say we have 3 or more jobs on a machine. We know that pi >= pn. And hence, the makespan of that machine >= 3pn > 3\* (T\*/3) > T\*max (Notice the strict ineq) which is not possible. So the number of jobs J \le 2m.

When the number of jobs \le 2m and each processor has at most 2 jobs, we can show that the greedy sorted algorithm gives the best allocation.