Algorithms Illuminated Part 3, Ch. 13 — Andrew Doss

13.1

length $= \ell_j$

deadline $= d_j$

lateness $= \lambda_j(\sigma) = C_j(\sigma) - d_j$ (0 if $C_j(\sigma) \leq d_j$)

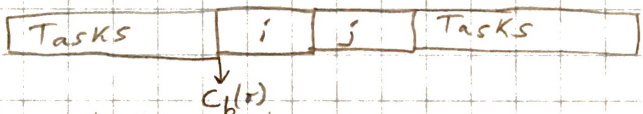
Objective: minimize the maximum lateness

$\max_{j=1}^{n} \lambda_j(\sigma)$

Which minimizes the maximum lateness?

Intuitively, seems we should schedule in order of increasing deadline.

Let's take an arbitrary sequence, similar to in the chapter. There must be at least one pair with i before j where $d_i > d_j$ (that is not ordered entirely by deadline)



Now the completion time of j $C_j(\sigma) > C_i(\sigma)$

$C_j(\sigma) - C_i(\sigma) = \ell_j$

If we flip them, $C_j(\sigma)$ decreases by ℓ_i

before we had $\lambda_j(\sigma) = C_b(\sigma) + \ell_i + \ell_j - d_j$

$\lambda_i(\sigma) = C_b(\sigma) + \ell_i - d_i$

after flipping we have:

$\lambda_j(\sigma') = C_b(\sigma) + \ell_j - d_j$

$\lambda_i(\sigma') = C_b(\sigma) + \ell_i + \ell_j - d_i$

$d_i > d_j$ so in the first case

$C_b(\sigma) + \ell_i + \ell_j - d_j$ is critical

in the later case, it is indeterminate. However, it is clear that $C_b(\sigma) + \ell_j - d_j$ is larger than the lateness of either task after the flip.

Therefore, an arbitrary flip can only improve the objective and we can make such flips until the schedule is sorted by deadline.

Answer is a.

13.2 Now we are trying to minimize <u>total lateness</u>

$\sum_{j=1}^{n} \lambda_j(\sigma)$

a) Schedule in increasing order of deadlines

Contradiction/counterexample:

$\ell_1 = 1, d_1 = 2$

$\ell_2 = 3, d_2 = 1$

If task 2 is first (this policy): $(3-1)+(4-2) = 4$

If task 1 is first (other policy): $0 + (4-1) = 3$

Therefore sorting by deadline is not always optimal. (False)

b) counterexample:

$\ell_1 = 1, d_1 = 4$

$\ell_2 = 3, d_2 = 4$

This policy puts task 1 first at a cost of $0 + (4-1) = 3$. If this order is flipped, the cost is $(3-1) = 2$. Therefore, this policy cannot be optimal. (False.)

c) Scheduling by increasing product:

$\ell_1 = 1, d_1 = 2$

$\ell_2 = 3, d_2 = 1$

Policy schedules task 2 first. cost is $2 + 2 = 4$

Alternate order puts task 1 first. cost is $0 + 3 = 3$.

This shows that c is not optimal. (False)

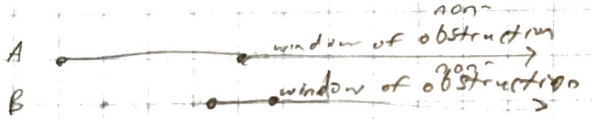
a-c are ruled out, so (d) must be the answer.

13.3

example : 0 1 2 3 4 5 6 7

Objective: select maximum subset of jobs with no conflicts.

a) is correct. I imagine selecting a job with a completion time that is not earliest. There is no way that this job interferes with fewer jobs than the one with the earliest completion time. Think about it this way.

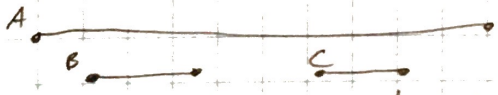


The window of non-obstruction is strictly larger for a. What if a task could squeeze in before b? well then it would have the earliest completion time instead.

c)

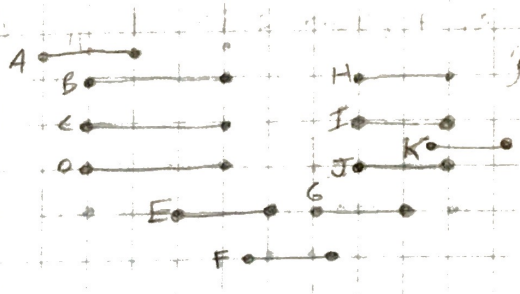
This policy fails here because a and c are possible together, yet only b is selected.

b) This is also easy to show a counter example for,



Earliest start time is clearly wrong.

d)



Min conflicts picks F. Then one of A-D and one of H-K for 3 total. First completion picks A, E, G, and K for 4 total. Therefore, min conflicts cannot be correct.