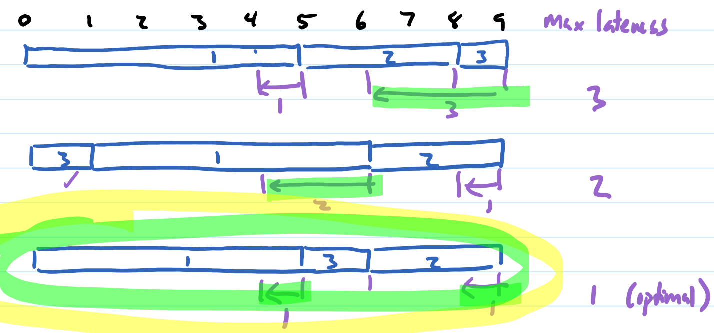
Minimizing Lateness

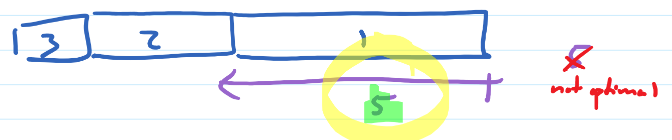
Given requests with lengths $t_1, \dots, t_n$, deadlines $d_1, \dots, d_n$, find schedule that minimizes maximum lateness.

Ex:

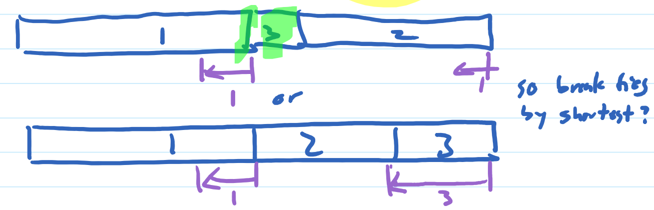
i	t_i	d_i
1	5	4
2	3	8
3	1	6



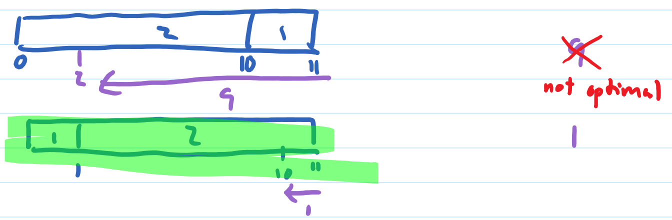
shortest first?



most pressing first?
latest if done next
$(\max\ t_{curr} + t_i - d_i)$
or $\max\ t_i - d_i$



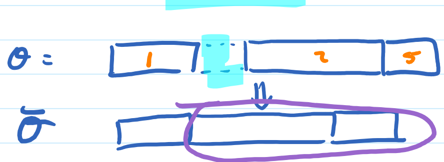
	t_i	d_i
1	1	2
2	10	10



optimal for both examples

earliest deadline 1st
↪ schedule tasks one after another w/ no idle time in order of ↑ deadline

1) There is an optimal schedule with no idle time



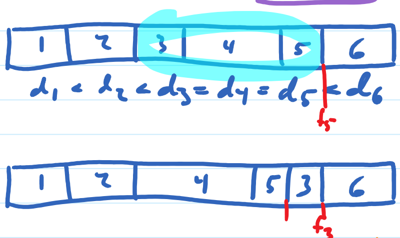
finish time ↓
lateness = finish − deadline ↓

$L = \max(\ell_1, \ell_2, \dots, \ell_n)$
terms all = or ↓
so max = or ↓

$\bar{L} \le L$ L is optimal
so Θ is still optimal

tasks i, j s.t. i before j in sched but $d_j < d_i$

2) All schedules with no idle time and no inversions have same max lateness



$L = \max(\ell_1, \ell_2, \ell_3, \ell_4, \ell_5, \ell_6) = \max(\max(\ell_3, \ell_4, \ell_5), \ell_1, \ell_2, \ell_6)$

$\bar{L} = \max(\max(\bar{\ell}_3, \bar{\ell}_4, \bar{\ell}_5), \bar{\ell}_1, \bar{\ell}_2, \bar{\ell}_6)$

$= \max(\bar{\ell}_3, \ell_1, \ell_2, \ell_6)$

$= \max(\ell_5, \ell_1, \ell_2, \ell_6)$ b/c $\bar{\ell}_3 = \bar{f}_3 - d_3 = f_5 - d_3 = f_5 - d_5 = \ell_5$

$= L$

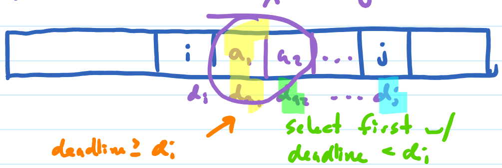
In general, reordering consecutive tasks w/ equal deadline doesn't change overall max lateness b/c it doesn't change lateness of the other tasks and the max lateness of the reordered tasks is the latest finish time − their deadline, and that doesn't change after reordering

3) There is a optimal schedule with no inversions, no idle time.

Find opt schedule σ with no idle time (1)

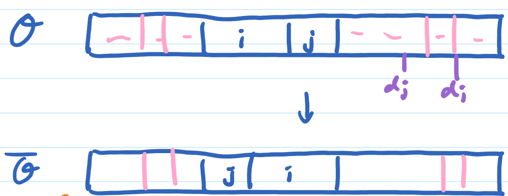
If σ has no inversions, :) (σ is what we need)

Else σ has an inversion i immediately before j but $d_j < d_i$



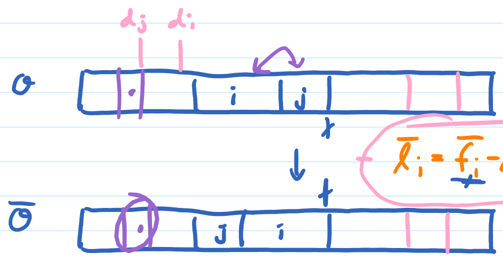
consider tasks between i, j; select those w/ deadlines before d_i

if i, j on time in both $\sigma, \bar{\sigma}$



[want max lateness in $\bar{\sigma}$ = max lateness in σ]
neither i nor j late in σ or $\bar{\sigma}$, no other tasks change finish, so no change in lateness, so no change in max lateness

if i, j late in both



$\bar{\ell}_i = \bar{f}_i - d_i = f_j - d_i$
$< f_j - d_j$ b/c $d_j < d_i$
$= \ell_j$

$L = \max(\ell_1, \ldots, \ell_{i-1}, \ell_i, \ell_{i+1}, \ldots, \ell_{j-1}, \ell_j, \ell_{j+1}, \ldots, \ell_n)$

$\bar{L} = \max(\ell_1, \ldots, \ell_{i-1}, \bar{\ell}_i, \ell_{i+1}, \ldots, \ell_{j-1}, \bar{\ell}_j, \ell_{j+1}, \ldots, \ell_n)$ $\bar{\ell}_k = \ell_k$ for $k \neq i, j$

$\bar{\ell}_j = \ell_j - t_i < \ell_j$

$\leq \max(\ell_1, \ldots, \ell_{i-1}, \bar{\ell}_i, \ell_{i+1}, \ldots, \ell_{j-1}, \ell_j, \ell_{j+1}, \ldots, \ell_n)$

$\leq \max(\ell_1, \ldots, \ell_{i-1}, \ell_j, \ell_{i+1}, \ldots, \ell_{j-1}, \ell_j, \ell_{j+1}, \ldots, \ell_n)$ $\ell_j > \bar{\ell}_i$

$= \max(\ell_1, \dots, \ell_{i-1}, \quad \ell_{i+1}, \dots, \ell_{j-1}, \bar{\ell}_j, \ell_{j+1}, \dots, \ell_n)$ remove duplicate term

$\le \max(\ell_1, \dots, \ell_{i-1}, \ell_i, \ell_{i+1}, \dots, \ell_{j-1}, \bar{\ell}_j, \ell_{j+1}, \dots, \ell_n)$

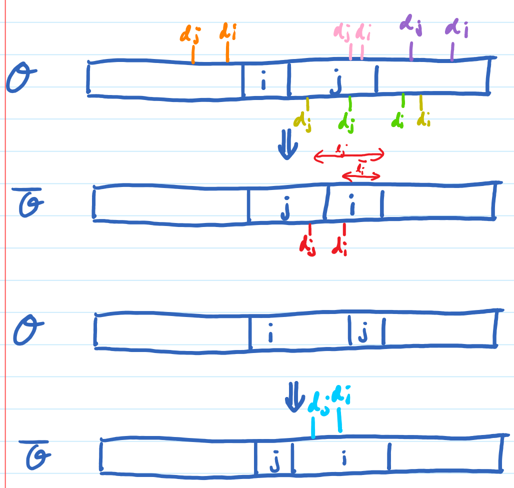
$= L$

$\bar{L} \le L$

$\bar{L} = L$ can't be better than OPT

$\bar{O}$ is optimal too (same max lateness $\bar{L}$ as O's max lateness L)

All possible cases (of lateness of i, j before/after fixing inversion)



in $\mathcal{O}$ i	j	in $\overline{\mathcal{O}}$ i	j	
N	N	N	N	$\max(\overline{\ell_i}, \overline{\ell_j}) = \max(0,0) = \max(\ell_i, \ell_j)$
~~N~~	~~N~~	~~Y~~	~~N~~	j not late $\Rightarrow$ i can't become late
N	Y	N	N	$\max(\overline{\ell_i}, \overline{\ell_j}) = 0 < \ell_j = \max(0, \ell_j) = \max(\ell_i, \ell_j)$
N	Y	N	Y	max lateness = j's lateness, which ↓
N	Y	Y	N	DONE on other slide
N	Y	Y	Y	same as
Y	Y	Y	N	
Y	Y	Y	Y	DONE on other slide

$$\max(\overline{\ell_i}, \overline{\ell_j}) = \max(\overline{\ell_i}, \ell_j - t_i) \le \max(\ell_j, \ell_j - t_i) = \max(\ell_j) \underset{\text{in YY}\to\text{YY case this is }\le}{=} \max(\ell_i, \ell_j)$$

$$\max(\overline{\ell_i}, \overline{\ell_j}) \underset{\overline{\ell_j}=0}{=} \overline{\ell_i} \underset{\text{def}}{=} \overline{f}(i) - d_i \underset{\overline{f}(i)=f(j)}{=} f(j) - d_i \underset{d_i > d_j}{<} f(j) - d_j \underset{\text{def}}{=} \ell_j \underset{\max}{\le} \max(\ell_i, \ell_j)$$

Optimal Caching

Memory Hierarchy: compiler registers — fast, expensive

CPU manages:
- L1 cache
- L2 cache
- L3 cache — 25MB

OS shuffles data here:
- main memory — RAM — 8GB = $75
- flash storage — 120GB = $75

disk storage

tape storage — slow, cheap

caching: keep soon-to-be-accessed data in fast memory

spatial locality – if use data, likely will use nearby data soon

temporal locality – if something used recently, likely to be used soon

Optimal caching: given cache size k, sequence of requests $d_1, \dots, d_m$ for data items, find eviction schedule to minimize cache misses

Ex: $k = 2$, initial cache = 1, 2   requests 1, 2, 3, 2, 1, 3, 4, 2, 3, 1

LRU
least recently used

1 ~~1~~ ~~3~~ | 1
2 ~~2~~ | 3

Farthest-in-Future
optimal (but not achievable w/o mystical powers)

proof works by showing that FF initially agrees with all opt schedules, and after each step it agrees w/ some opt schedules, so in the end it agrees w/ at least 1 opt and so is optimal itself same structure for MST proofs