Greedy Algorithms

4

A Task Scheduling Problem with Deadlines and Penalties

CLRS Chapter 16.5

مسئلهٔ زمان‌بندی با مهلت و جریمه

Scheduling Tasks with Deadlines and Penalties for a Single Processor

صورت ساده سازی شده مسئله زمان بندی با مهلت و جریمه تاخیر

- Scheduling unit-time tasks
- To be processed by a single processor
- Each task has a deadline ; d_j
- Penalty paid if the task misses its deadline

فرض کنید: $t_j = 1$ طول زمان اجرای هر وظیفه (تسک) یک است.

وظیفه ها قرار است توسط یک پردازنده طبق زمان بندی مشخصی به انجام برسند.

w_j : مقدار جریمه ای که در صورت دیرکرد اجرای وظیفه j ام باید بپردازیم

* توجه کنید که جریمه تاخیر، ثابت و مستقل از طول تاخیر است.

Scheduling Tasks with Deadlines and Penalties for a Single Processor

- a set $S = \{a_1, a_2, \ldots, a_n\}$ of n unit-time tasks;
- a set of n integer ***deadlines*** $d_1, d_2, \ldots, d_n$, such that each d_i satisfies $1 \le d_i \le n$ and task a_i is supposed to finish by time d_i; and
- a set of n nonnegative weights or ***penalties*** $w_1, w_2, \ldots, w_n$, such that we incur a penalty of w_i if task a_i is not finished by time d_i, and we incur no penalty if a task finishes by its deadline.

n = 7

Sort →

	Task						
a_i	1	2	3	4	5	6	7
d_i	4	2	4	3	1	4	6
w_i	70	60	50	40	30	20	10

50+20 = 70

Example 2, 5, 4, 6, 1, 7, 3

	Task 1	2	3	4	5	6	7
a_i	4	0	6	2	1	3	5
d_i	4	2	4	3	1	4	6
w_i	70	60	50	40	30	20	10

$S^1 = \langle 2,5^*,4,6,1^*,7,3^* \rangle \rightarrow \text{penalty}^1 = 30+70+50 = 150$

$S^2 = \langle 2,4,6,7 \mid 5^*,1^*,3^* \rangle \rightarrow \text{penalty}^2 = 30+70+50 = 150$

$S^3 = \langle 5,2,4,3 \mid 1^*,6^*,7^* \rangle \rightarrow \text{penalty}^3 = 70+20+10 = 100$

فرم کانونی

"دیرهنگام" late

اگر زمان پایان اجرای یک وظیفه قبل از مهلت تعیین شده و یا برابر با آن باشد: "به‌هنگام" On-time

می گوییم یک برنامه زمان‌بندی در فرم کانونی نوشته شده است اگر:

۱) تمام وظیفه‌های دیرهنگام در انتهای لیست زمان‌بندی قرار گرفته باشند.

۲) وظیفه‌های بهنگام برحسب مهلت (deadline) به ترتیب صعودی مرتب شده باشند.

از آنجائیکه بازنویسی هر برنامه زمانبندی در فرم کانونی، هزینه جریمه نهایی تغییر نمی‌دهد

⟸ کافی است برای پیدا کردن زمانبندی بهینه تنها زمانبندی‌هایی که در فرم کانونی هستند را جستجو کنیم.

در این مسئله انتخاب حریصانه، به ترتیب انتخاب وظیفه‌ای است که بیشترین جریمه را دارد.

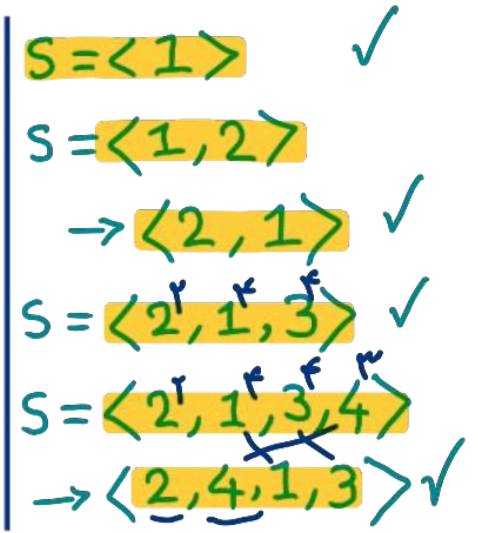
Example

	Task						
a_i	1	2	3	4	5	6	7
d_i	4	2	4	3	1	4	6
w_i	70	60	50	40	30	20	10

$\mathbf{S^1 = \langle 2,5^*,4,6,1^*,7,3^* \rangle \quad \rightarrow \quad penalty^1 = 30+70+50 = 150}$

$\mathbf{S^2 = \langle 2,4,6,7 \mid 5^*,1^*,3^* \rangle \quad \rightarrow \quad penalty^2 = 30+70+50 = 150}$

$\mathbf{S^3 = \langle 5,2,4,3 \mid 1^*,6^*,7^* \rangle \quad \rightarrow \quad penalty^3 = 70+20+10 = 100}$



$S = \langle 2^{2}, 4^{3}, 1^{4}, 3^{4}, 5^{1} \rangle$

$\rightarrow \langle 5, 2, 4, 1, 3^{*} \rangle \Rightarrow$ 5 reject

به سمت دیر هنگام ها منتقل می‌کنم

$\langle 2, 4, 1, 3 \mid 5^{*} \rangle$

$S = \langle 2^{2}, 4^{2}, 1^{4}, 3^{4}, 6^{4*} \mid 5^{*} \rangle \Rightarrow$ 6 reject

به سمت دیر هنگام ها منتقل می‌کنم

$\rightarrow S = \langle 2, 4, 1, 3 \mid 5^{*}, 6^{*} \rangle$

$S = \langle 2^{2}, 4^{2}, 1^{4}, 3^{4}, 7^{4} \mid 5^{*}, 6^{*} \rangle$

$\Rightarrow$ مجموع کل $= 30 + 20 = 50$

5 Scheduling to minimizing lateness

مطالعه آزاد

مسئله کمینه سازی "دیرکرد"

Minimizing lateness problem.

- Single resource processes one job at a time.
- Job j requires t_j units of processing time and is due at time d_j.
- If j starts at time s_j, it finishes at time $f_j = s_j + t_j$.
- Lateness: $\ell_j = \max \{ 0, f_j - d_j \}$.
- Goal: schedule all jobs to minimize maximum lateness $L = \max_j \ell_j$.

یک پردازنده که در آن واحد جداکثر یک وظیفه می تواند اجرا کند.

t_j : طول زمان اجرای وظیفه ام

d_j : مهلت اجرای ام

s_j و f_j : زمان شروع و خاتمه

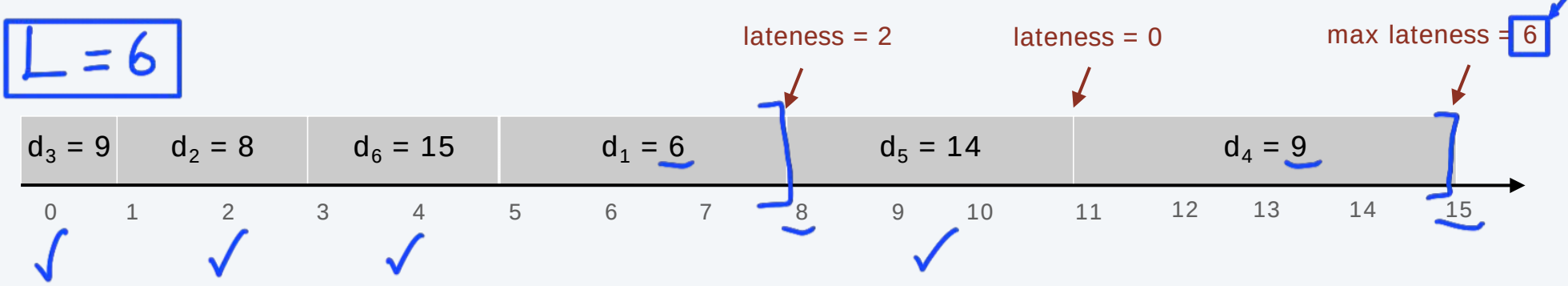
if $f_j \leq d_j$ then "on-time"

else

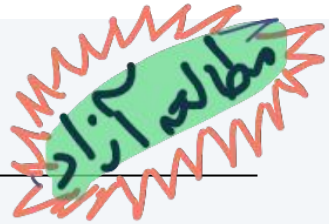
$\ell_j = f_j - d_j$

	1	2	3	4	5	6
t_j	3	2	1	4	3	2
d_j	6	8	9	9	14	15

$L = 6$



Minimizing lateness: greedy algorithms



Greedy template. Schedule jobs according to some natural order.

1 • [Shortest processing time first] Schedule jobs in ascending order of processing time t_j.

انتخاب حریصانه بر حسب طول زمان اجرا

"کوتاهتر اول"

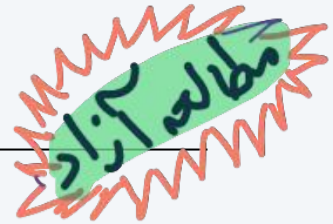
2 • [Earliest deadline first] Schedule jobs in ascending order of deadline d_j.

"زودآمدترین مهلت تحویل" ← جواب بهینه

3 • [Smallest slack] Schedule jobs in ascending order of slack $d_j - t_j$.

"کم فرصت ترین"

Minimizing lateness: greedy algorithms



Greedy template. Schedule jobs according to some natural order.

- [Shortest processing time first] Schedule jobs in ascending order of processing time t_j.

<2, 1> در حالتی
هیچ دیرکردی نداشت

	1	2
t_j	1	10
d_j	100 ✓	10*

0 1 11

L = 1

counterexample

مثال نقض برای
انتخاب کوتاهترین طول اجرا
"مرتب سازی بر حسب t_j"

- [Smallest slack] Schedule jobs in ascending order of slack $d_j - t_j$.

<1,2> در حالتی اگر
1: s=0, f=1, ontime
2: s=1, f=11, l_2=1
=> L=1

	1	2
t_j	10 1 11	0 10 10
d_j	2	10

slack 1 0 ✓

L=9

counterexample

مثال نقض برای
انتخاب کم فرصت ترین
"مرتب سازی بر حسب $d_j - t_j$"

Minimizing lateness: earliest deadline first

EARLIEST-DEADLINE-FIRST $(n, t_1, t_2, \ldots, t_n, d_1, d_2, \ldots, d_n)$

SORT n jobs so that $d_1 \le d_2 \le \ldots \le d_n$.

$t \leftarrow 0$, $L = 0$

t: زمان شروع

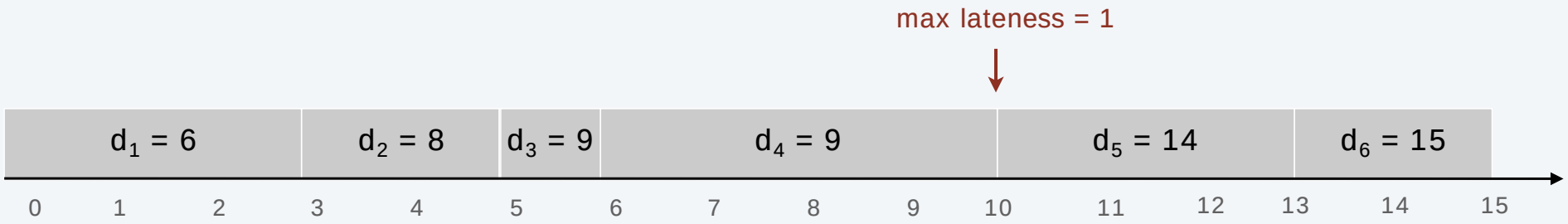
FOR $j = 1$ TO n

  Assign job j to interval $[t, t + t_j]$.

  $s_j \leftarrow t$; $f_j \leftarrow t + t_j$

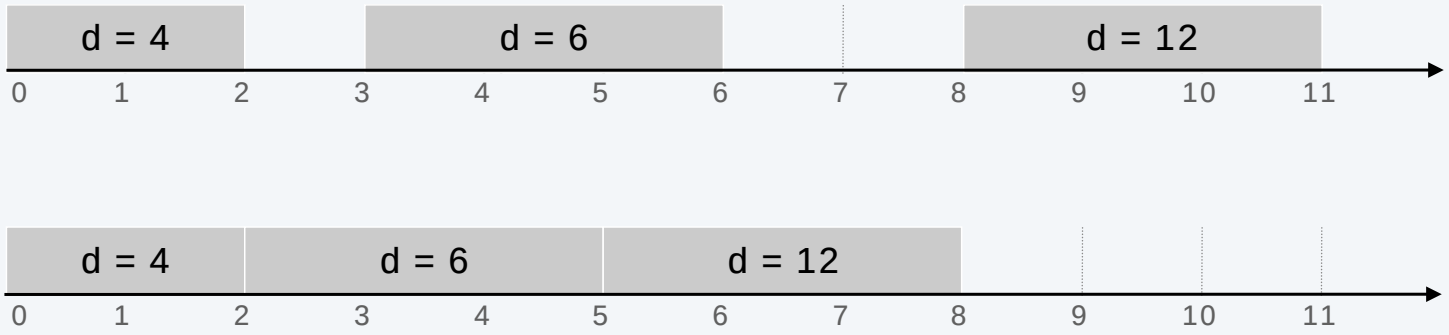
  $t \leftarrow t + t_j$; $L = \max(L, f_j - d_j)$

RETURN intervals $[s_1, f_1], [s_2, f_2], \ldots, [s_n, f_n]$, L.



Minimizing lateness: no idle time

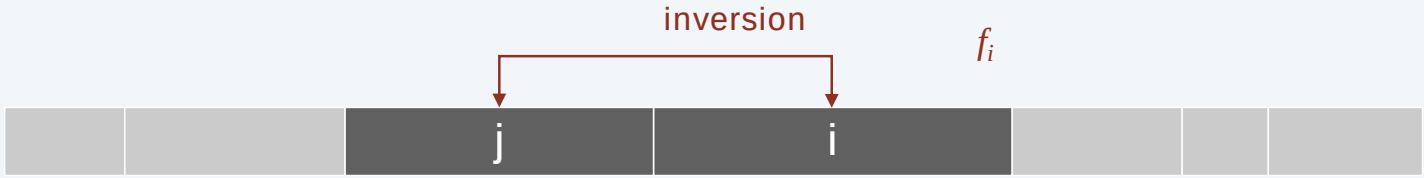
Observation 1. There exists an optimal schedule with no idle time.



Observation 2. The earliest-deadline-first schedule has no idle time.

Minimizing lateness: inversions

Def. Given a schedule S, an inversion is a pair of jobs i and j such that: $i < j$ but j scheduled before i.

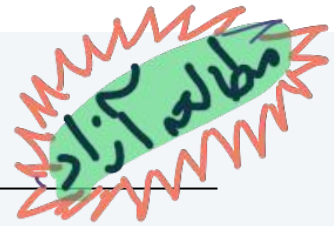


[as before, we assume jobs are numbered so that $d_1 \leq d_2 \leq \ldots \leq d_n$]

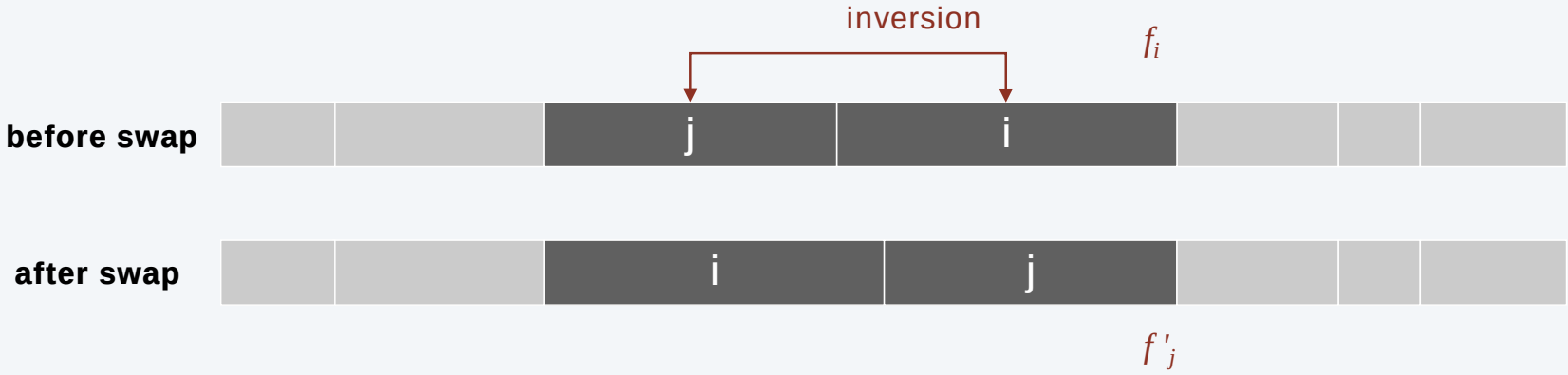
Observation 3. The earliest-deadline-first schedule has no inversions.

Observation 4. If a schedule (with no idle time) has an inversion, it has one with a pair of inverted jobs scheduled consecutively.

Minimizing lateness: inversions



Def. Given a schedule S, an inversion is a pair of jobs i and j such that: $i < j$ but j scheduled before i.



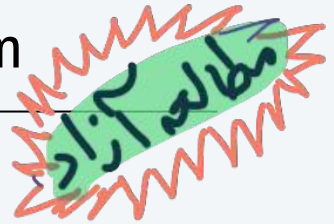
Claim. Swapping two adjacent, inverted jobs reduces the number of inversions by one and does not increase the max lateness.

Pf. Let ℓ be the lateness before the swap, and let ℓ' be it afterwards.

- $\ell'_k = \ell_k$ for all $k \neq i, j$.
- $\ell'_i \leq \ell_i$.
- If job j is late,

$$\begin{aligned} \ell'_j &= f'_j - d_j && \text{(definition)} \\ &= f_i - d_j && (j \text{ now finishes at time } f_i) \\ &\leq f_i - d_i && (\text{since } i \text{ and } j \text{ inverted}) \\ &\leq \ell_i . && \text{(definition)} \end{aligned}$$

Minimizing lateness: analysis of earliest-deadline-first algorithm



Theorem. The earliest-deadline-first schedule S is optimal.

Pf. [by contradiction]

Define S^* to be an optimal schedule that has the fewest number of inversions, and let's see what happens.

- Can assume S^* has no idle time.
- If S^* has no inversions, then $S = S^*$.
- If S^* has an inversion, let i–j be an adjacent inversion.
- Swapping i and j
  - does not increase the max lateness
  - strictly decreases the number of inversions
- This contradicts definition of S^* ▪