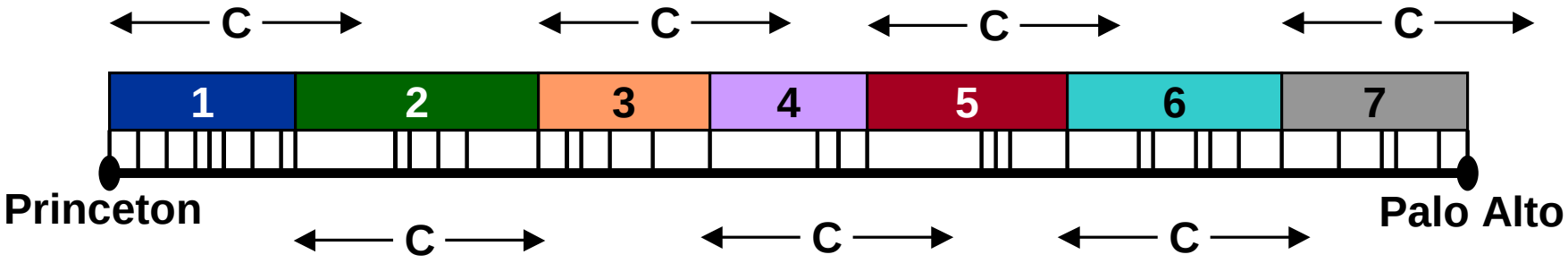
Selecting Breakpoints

Minimizing breakpoints.

- Truck driver going from Princeton to Palo Alto along predetermined route.
- Refueling stations at certain points along the way.
- Truck fuel capacity = C.

Greedy algorithm.

- Go as far as you can before refueling.



Selecting Breakpoints: Greedy Algorithm

Greedy Breakpoint Selection Algorithm

Sort breakpoints by increasing value:
$0 = b_0 < b_1 < b_2 < \ldots < b_n$.

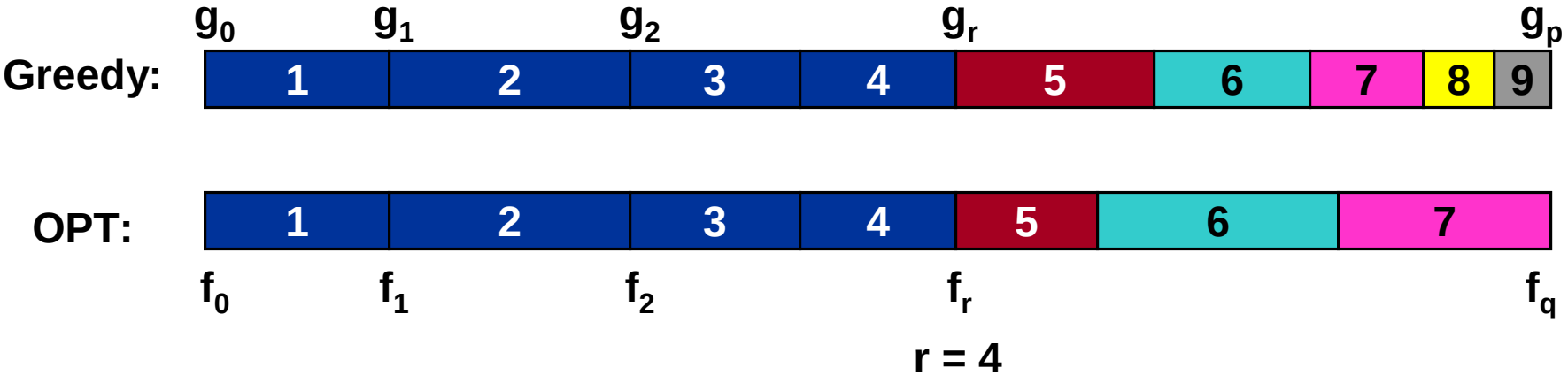
$S \leftarrow \{0\}$ ← S = breakpoints selected.
$x = 0$
while ($x \neq b_n$)
    let p be largest integer such that $b_p \leq x + C$
    if ($b_p = x$)
        return "no solution"
    $x \leftarrow b_p$
    $S \leftarrow S \cup \{p\}$
return S

Selecting Breakpoints

Theorem: greedy algorithm is optimal.

Proof (by contradiction):

- Let $0 = g_0 < g_1 < \ldots < g_p = L$ denote set of breakpoints chosen by greedy and assume it is not optimal.
- Let $0 = f_0 < f_1 < \ldots < f_q = L$ denote set of breakpoints in optimal solution with $f_0 = g_0, f_1 = g_1, \ldots, f_r = g_r$ for largest possible value of r.
- Note: q < p.

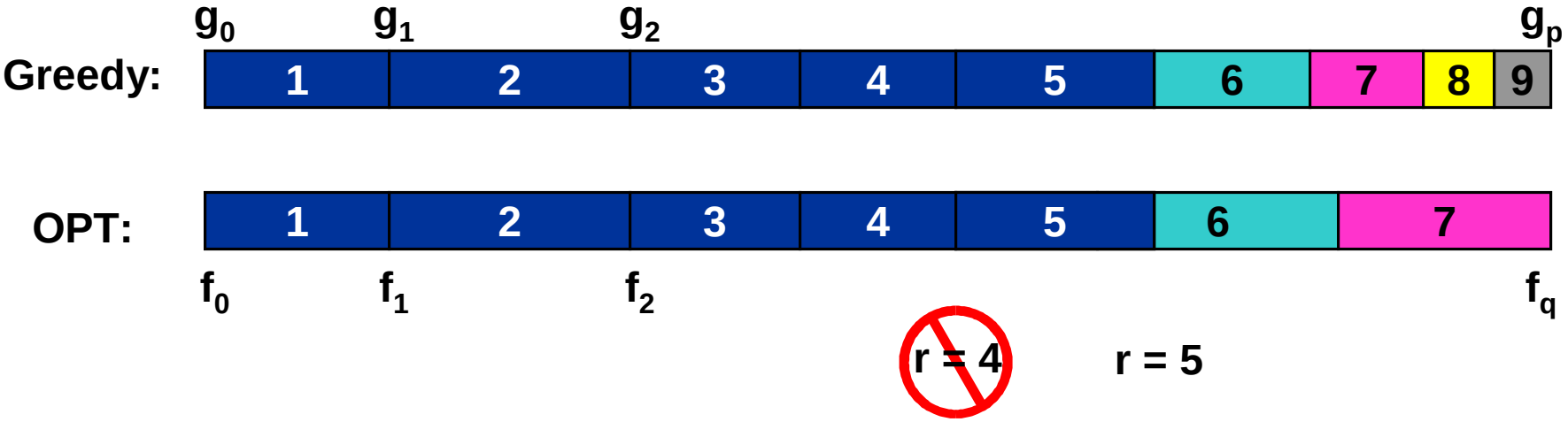


Selecting Breakpoints

Theorem: greedy algorithm is optimal.

Proof (by contradiction):

- Let $0 = g_0 < g_1 < \ldots < g_p = L$ denote set of breakpoints chosen by greedy and assume it is not optimal.
- Let $0 = f_0 < f_1 < \ldots < f_q = L$ denote set of breakpoints in optimal solution with $f_0 = g_0, f_1 = g_1, \ldots, f_r = g_r$ for largest possible value of r.
- Note: q < p.

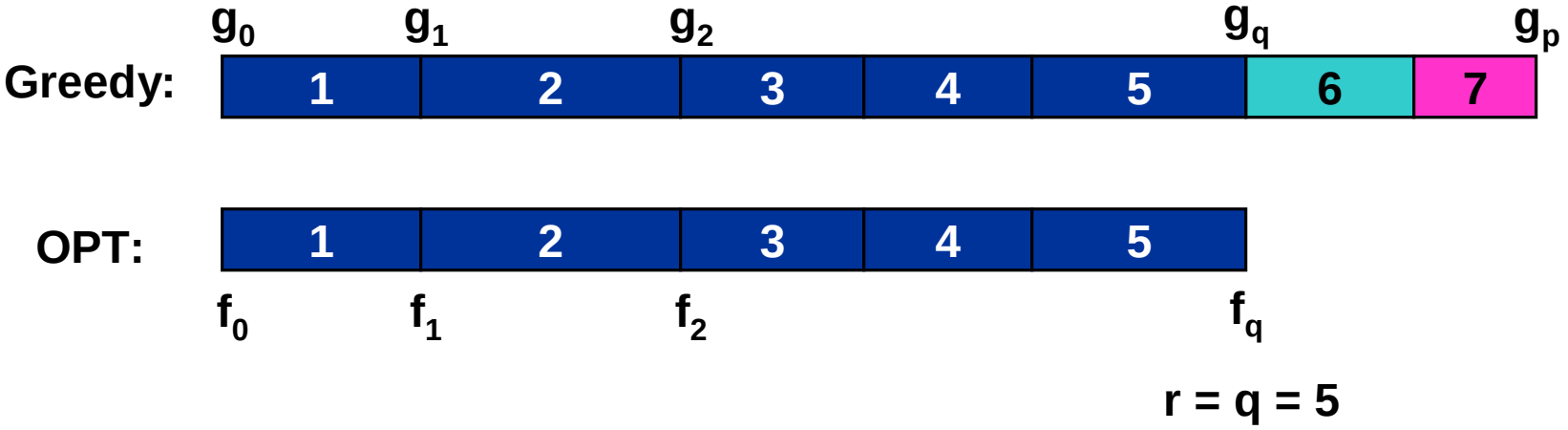


Selecting Breakpoints

Theorem: greedy algorithm is optimal.

Proof (by contradiction):

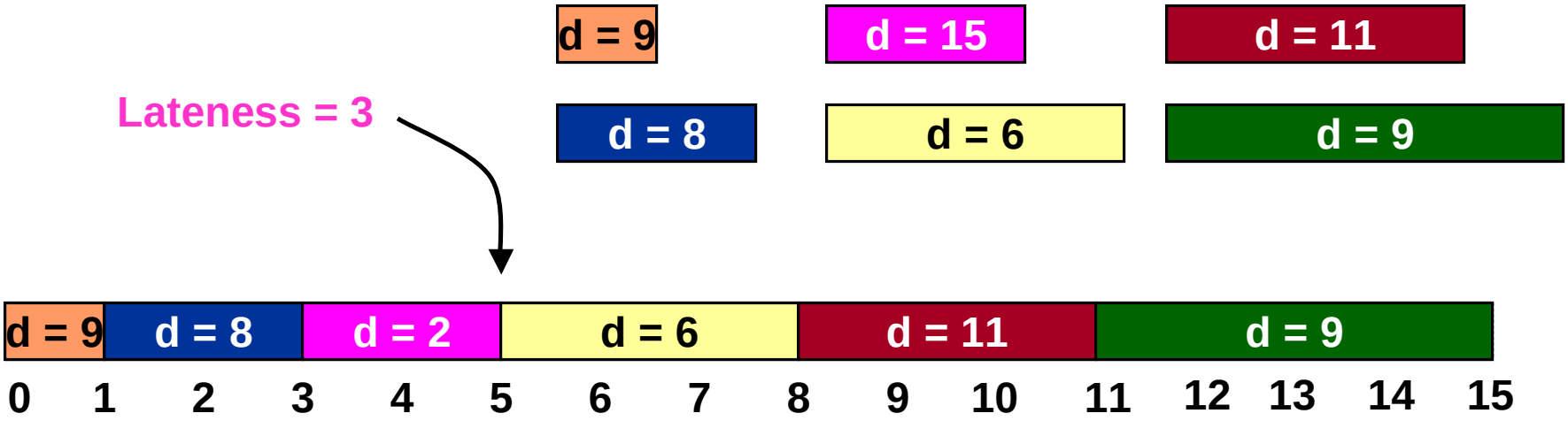
- Let $0 = g_0 < g_1 < \ldots < g_p = L$ denote set of breakpoints chosen by greedy and assume it is not optimal.
- Let $0 = f_0 < f_1 < \ldots < f_q = L$ denote set of breakpoints in optimal solution with $f_0 = g_0, f_1 = g_1, \ldots, f_r = g_r$ for largest possible value of r.
- Note: $q < p$.
- Thus, $f_0 = g_0, f_1 = g_1, \ldots, f_q = g_q$



Minimizing Lateness

Minimizing lateness problem.

- Single resource can process one job at a time.
- n jobs to be processed.
  - job j requires p_j units of processing time.
  - job j has due date d_j.
- If we assign job j to start at time s_j, it finishes at time $f_j = s_j + p_j$.
- Lateness: $\ell_j = \max \{ 0, f_j - d_j \}$.
- Goal: schedule all jobs to minimize maximum lateness $L = \max \ell_j$.

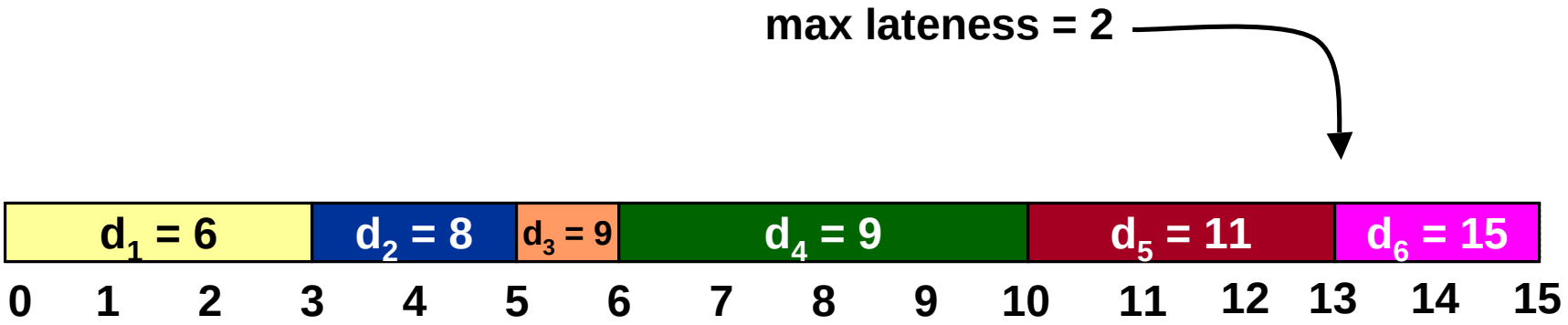


Minimizing Lateness: Greedy Algorithm

Greedy Activity Selection Algorithm

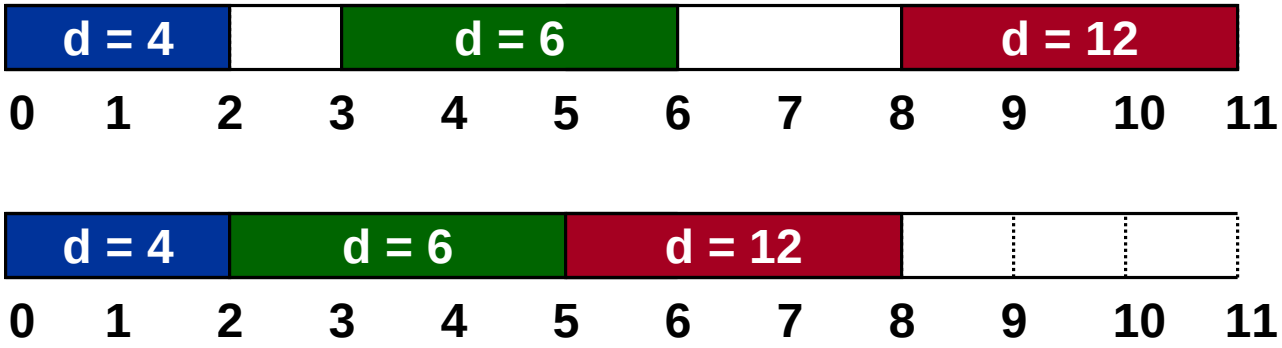
Sort jobs by increasing deadline so that $d_1 \le d_2 \le \ldots \le d_n$.

```
t = 0
for j = 1 to n
   Assign job j to interval [t, t + p_j]
   s_j ← t, f_j ← t + p_j
   t ← t + p_j
output intervals [s_j, f_j]
```



Minimizing Lateness: No Idle Time

Fact 1: there exists an optimal schedule with no idle time.

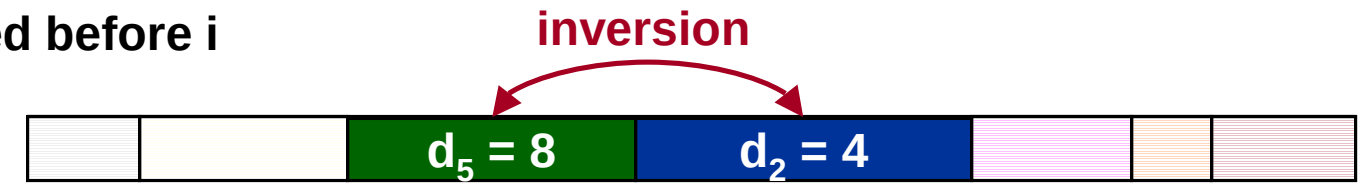


Fact 2: the greedy schedule has no idle time.

Minimizing Lateness: Inversions

An inversion in schedule S is a pair of jobs i and j such that:

- **i < j**
- **j scheduled before i**



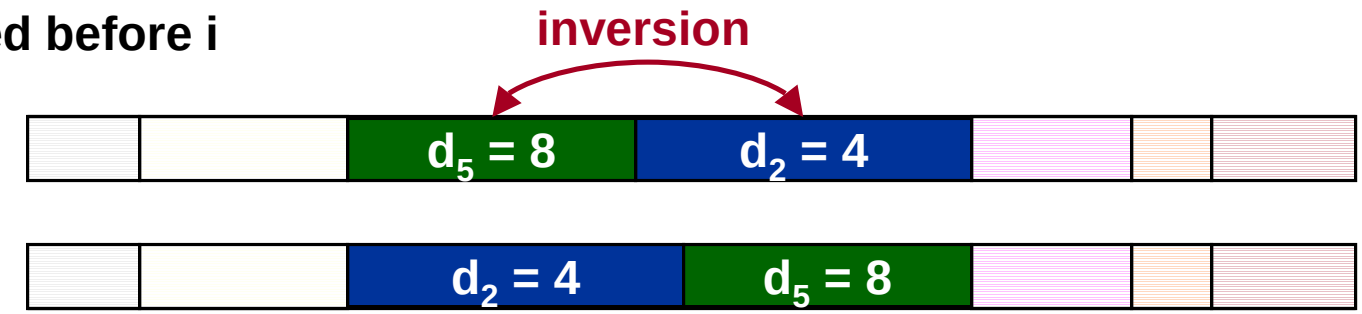
Fact 3: greedy schedule ⇔ no inversions.

Fact 4: if a schedule (with no idle time) has an inversion, it has one whose with a pair of inverted jobs scheduled consecutively.

Minimizing Lateness: Inversions

An inversion in schedule S is a pair of jobs i and j such that:

- **i < j**
- **j scheduled before i**



Fact 3: greedy schedule ⇔ no inversions.

Fact 4: if a schedule (with no idle time) has an inversion, it has one whose with a pair of inverted jobs scheduled consecutively.

Fact 5: swapping two adjacent, inverted jobs:

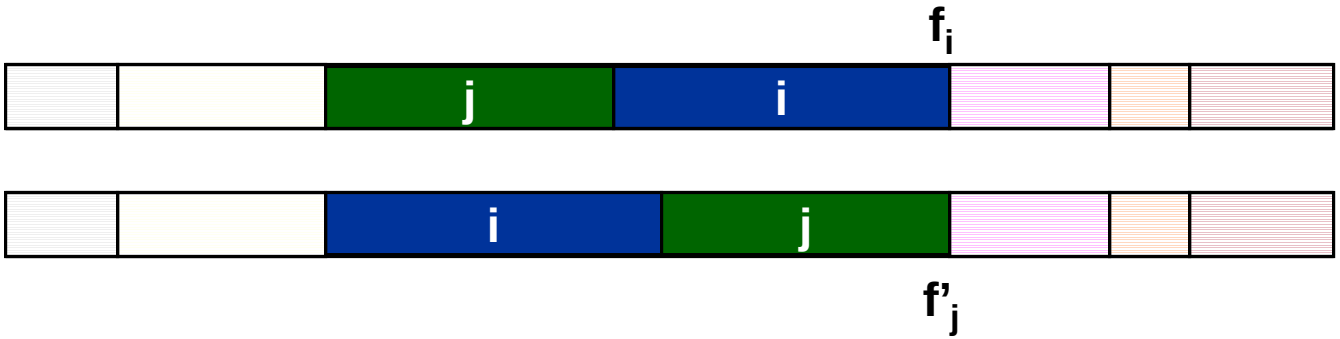
- **Reduces the number of inversions by one.**
- **Does not increase the maximum lateness.**

Theorem: greedy schedule is optimal.

Minimizing Lateness: Proof of Fact 5

An inversion in schedule S is a pair of jobs i and j such that:

- **i < j**
- **j scheduled before i**



Swapping two adjacent, inverted jobs does not increase max lateness.

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

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