CSE 403

Software Engineering

Delta Debugging

Today

Delta Debugging

- Motivating example
- The core algorithm
- Live demo
- A little quiz

This is a crashing test case

```
<td align=left valign=top>
<SELECT NAME="op sys" MULTIPLE SIZE=7>
<OPTION VALUE="All">All
<OPTION VALUE="Windows 3.1">Windows 3.1
<OPTION VALUE="Windows 95">Windows 95
<OPTION VALUE="Windows 98">Windows 98
<OPTION VALUE="Windows ME">Windows ME
<OPTION VALUE="Windows 2000">Windows 2000
<OPTION VALUE="Windows NT">Windows NT
<OPTION VALUE="Mac System 7">Mac System 7
<OPTION VALUE="Mac System 7.5">Mac System 7.5
<OPTION VALUE="Mac System 7.6.1">Mac System 7.6.1
<OPTION VALUE="Mac System 8.0">Mac System 8.0
<OPTION VALUE="Mac System 8.5">Mac System 8.5
<OPTION VALUE="Mac System 8.6">Mac System 8.6
<OPTION VALUE="Mac System 9.x">Mac System 9.x
<OPTION VALUE="MacOS X">MacOS X
<OPTION VALUE="Linux">Linux
<OPTION VALUE="BSDI">BSDI
<OPTION VALUE="FreeBSD">FreeBSD
<OPTION VALUE="NetBSD">NetBSD
<OPTION VALUE="OpenBSD">OpenBSD
<OPTION VALUE="AIX">AIX
<OPTION VALUE="BeOS">BeOS
<OPTION VALUE="HP-UX">HP-UX
<OPTION VALUE="IRIX">IRIX
<OPTION VALUE="Neutrino">Neutrino
<OPTION VALUE="OpenVMS">OpenVMS
<OPTION VALUE="OS/2">OS/2
<OPTION VALUE="OSF/1">OSF/1
<OPTION VALUE="Solaris">Solaris
<OPTION VALUE="SunOS">SunOS
<OPTION VALUE="other">other</SELECT></td>
<td align=left valign=top>
<SELECT NAME="priority" MULTIPLE SIZE=7>
<OPTION VALUE="--">--<OPTION VALUE="P1">P1<OPTION VALUE="P2">P2<OPTION
VALUE="P3">P3<OPTION VALUE="P4">P4<OPTION
VALUE="P5">P5</SELECT>
</td>
<td align=left valign=top>
<SELECT NAME="bug severity" MULTIPLE SIZE=7>
<OPTION VALUE="blocker">blocker<OPTION VALUE="critical">critical<OPTION
VALUE="major">major<OPTION
VALUE="normal">normal<OPTION VALUE="minor">minor<OPTION
VALUE="trivial">trivial<OPTION VALUE="enhancement">enhancement</SELECT>
</tr>
</table>
```

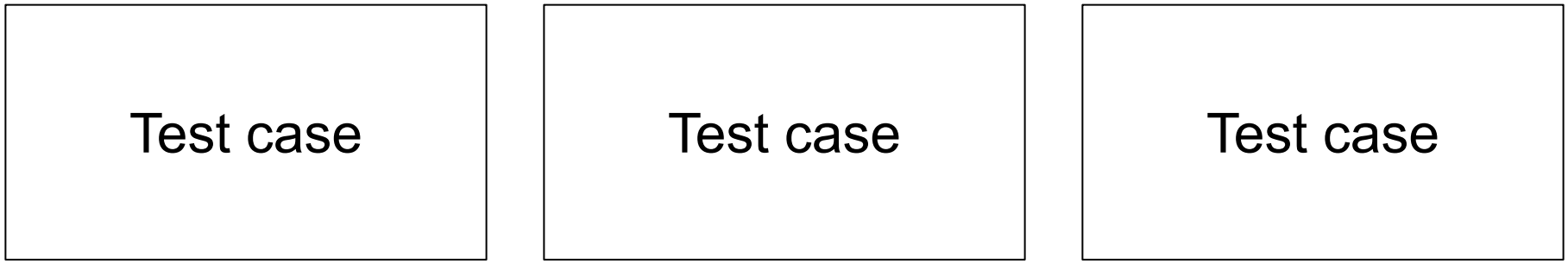
- Crashed Mozilla
- What content is sufficient to expose the bug?

This is a crashing test case

```
<td align=left valign=top>
<SELECT NAME="op sys" MULTIPLE SIZE=7>
<OPTION VALUE="All">All
<OPTION VALUE="Windows 3.1">Windows 3.1
<OPTION VALUE="Windows 95">Windows 95
<OPTION VALUE="Windows 98">Windows 98
<OPTION VALUE="Windows ME">Windows ME
<OPTION VALUE="Windows 2000">Windows 2000
<OPTION VALUE="Windows NT">Windows NT
<OPTION VALUE="Mac System 7">Mac System 7
<OPTION VALUE="Mac System 7.5">Mac System 7.5
<OPTION VALUE="Mac System 7.6.1">Mac System 7.6.1
<OPTION VALUE="Mac System 8.0">Mac System 8.0
<OPTION VALUE="Mac System 8.5">Mac System 8.5
<OPTION VALUE="Mac System 8.6">Mac System 8.6
<OPTION VALUE="Mac System 9.x">Mac System 9.x
<OPTION VALUE="MacOS X">MacOS X
<OPTION VALUE="Linux">Linux
<OPTION VALUE="BSDI">BSDI
<OPTION VALUE="FreeBSD">FreeBSD
<OPTION VALUE="NetBSD">NetBSD
<OPTION VALUE="OpenBSD">OpenBSD
<OPTION VALUE="AIX">AIX
<OPTION VALUE="BeOS">BeOS
<OPTION VALUE="HP-UX">HP-UX
<OPTION VALUE="IRIX">IRIX
<OPTION VALUE="Neutrino">Neutrino
<OPTION VALUE="OpenVMS">OpenVMS
<OPTION VALUE="OS/2">OS/2
<OPTION VALUE="OSF/1">OSF/1
<OPTION VALUE="Solaris">Solaris
<OPTION VALUE="SunOS">SunOS
<OPTION VALUE="other">other</SELECT></td>
<td align=left valign=top>
<SELECT NAME="priority" MULTIPLE SIZE=7>
<OPTION VALUE="--">--<OPTION VALUE="P1">P1<OPTION VALUE="P2">P2<OPTION
VALUE="P3">P3<OPTION VALUE="P4">P4<OPTION
VALUE="P5">P5</SELECT>
</td>
<td align=left valign=top>
<SELECT NAME="bug severity" MULTIPLE SIZE=7>
<OPTION VALUE="blocker">blocker<OPTION VALUE="critical">critical<OPTION
VALUE="major">major<OPTION
VALUE="normal">normal<OPTION VALUE="minor">minor<OPTION
VALUE="trivial">trivial<OPTION VALUE="enhancement">enhancement</SELECT>
</tr>
</table>
```

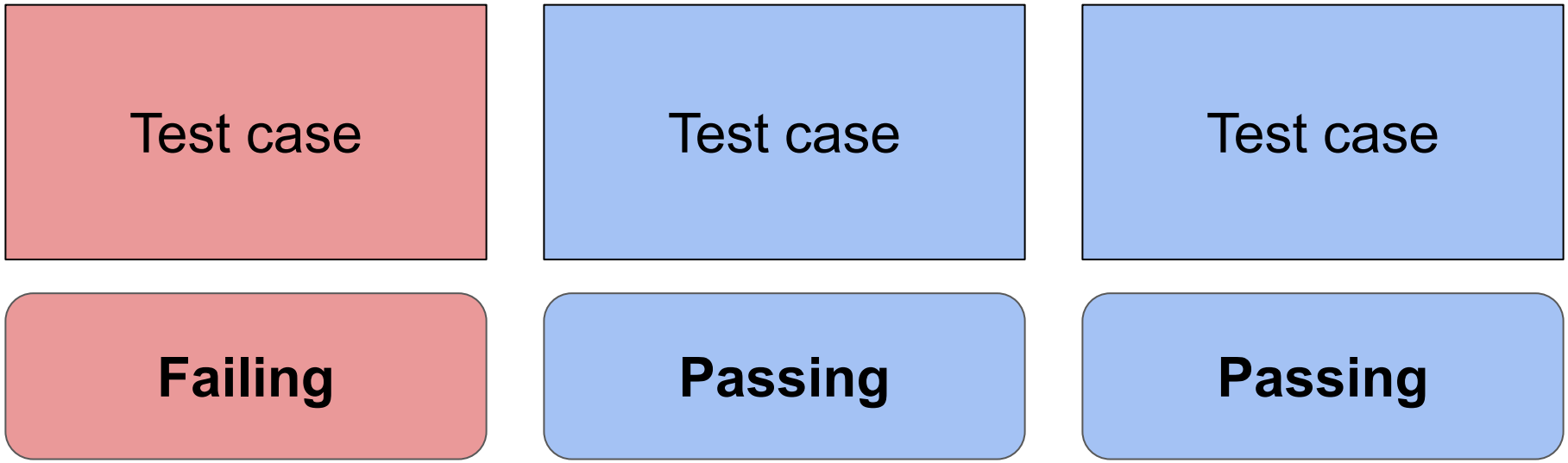
- Crashed Mozilla
- What content is sufficient to expose the bug?
- A minimal test case is: **<SELECT>**
- Can we automate the process of minimizing test cases?
- What's the naive approach for an **optimal** solution?

Minimizing test cases

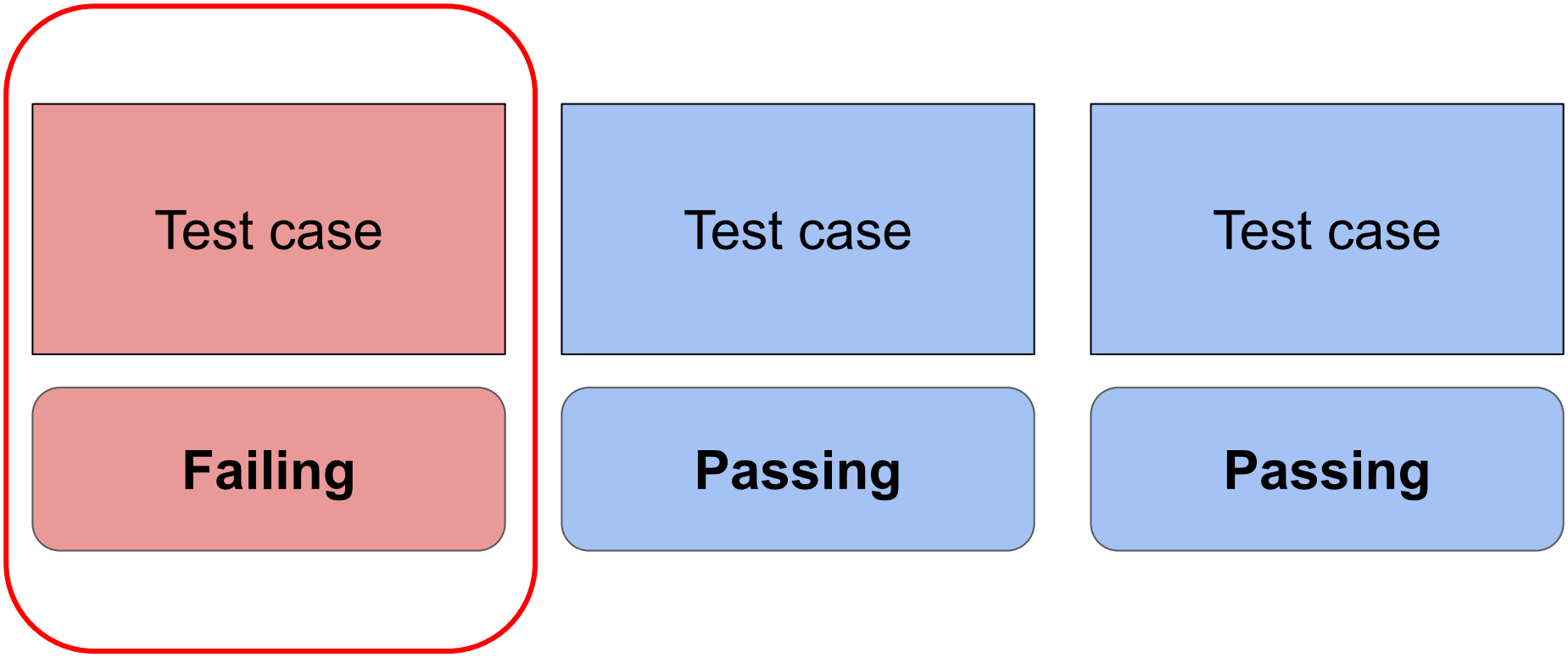


Think of each test case as an input file with n lines.

Minimizing test cases

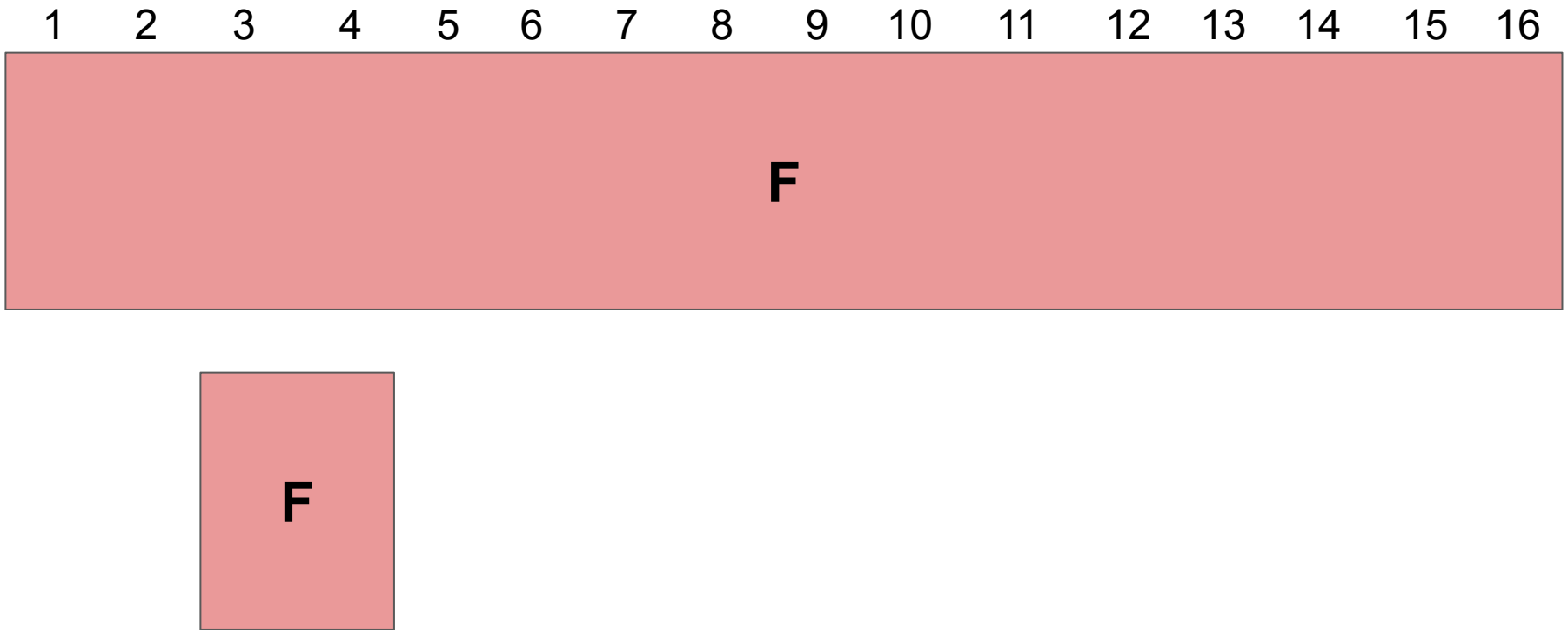


Minimizing test cases



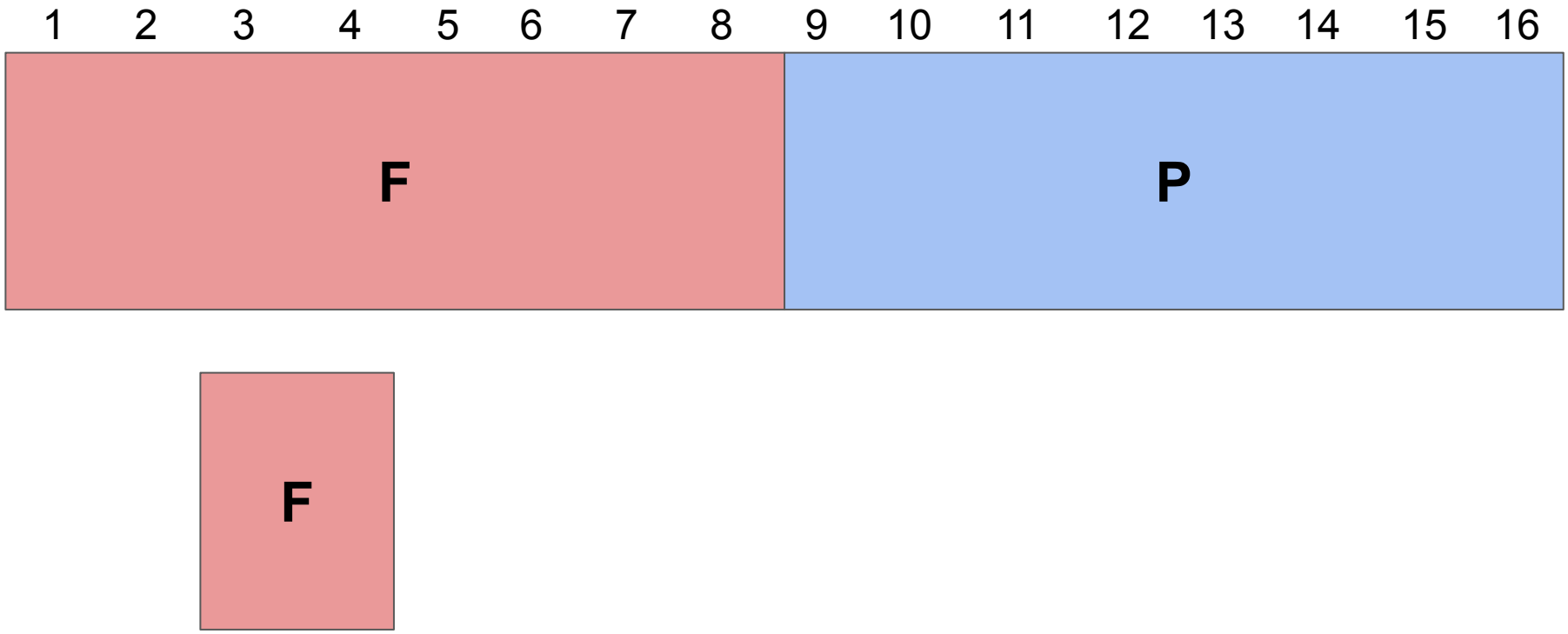
Goal: Minimize the failing test case

The happy path: binary search

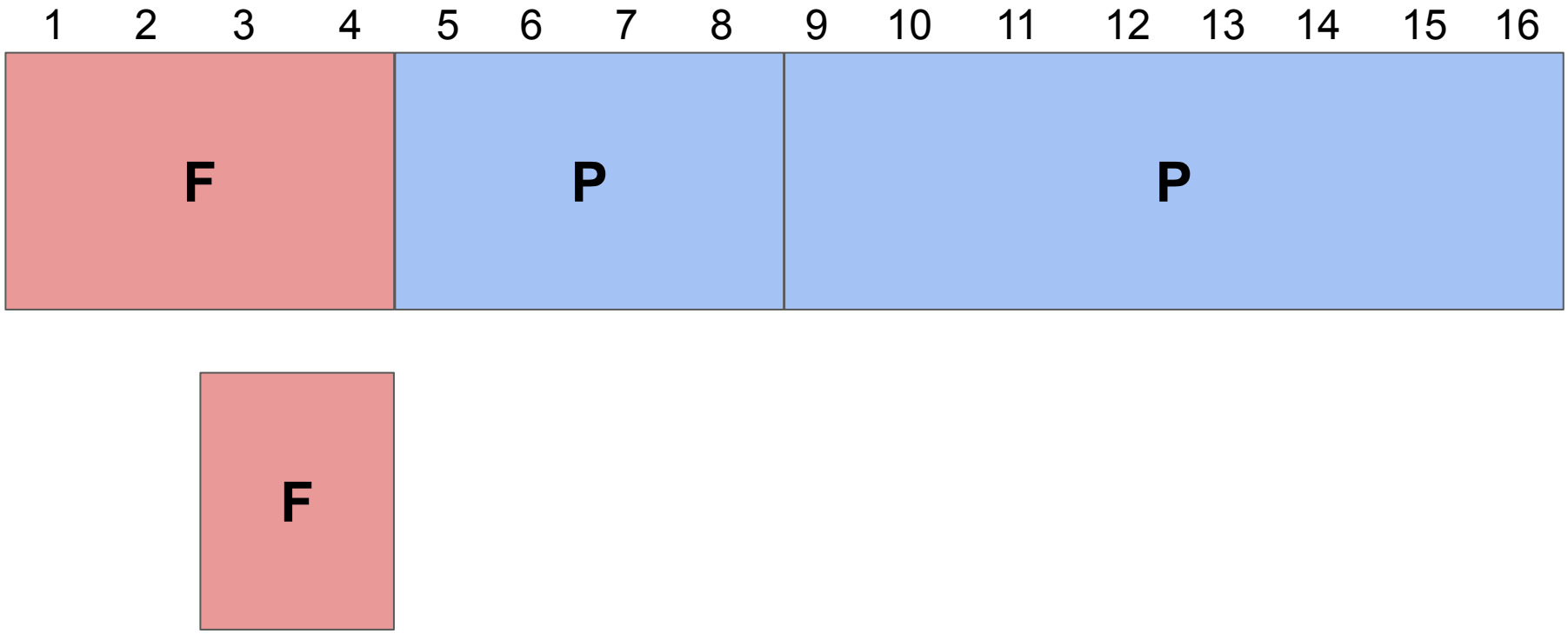


Failing test with 16 lines.
The minimal test has 2 lines.

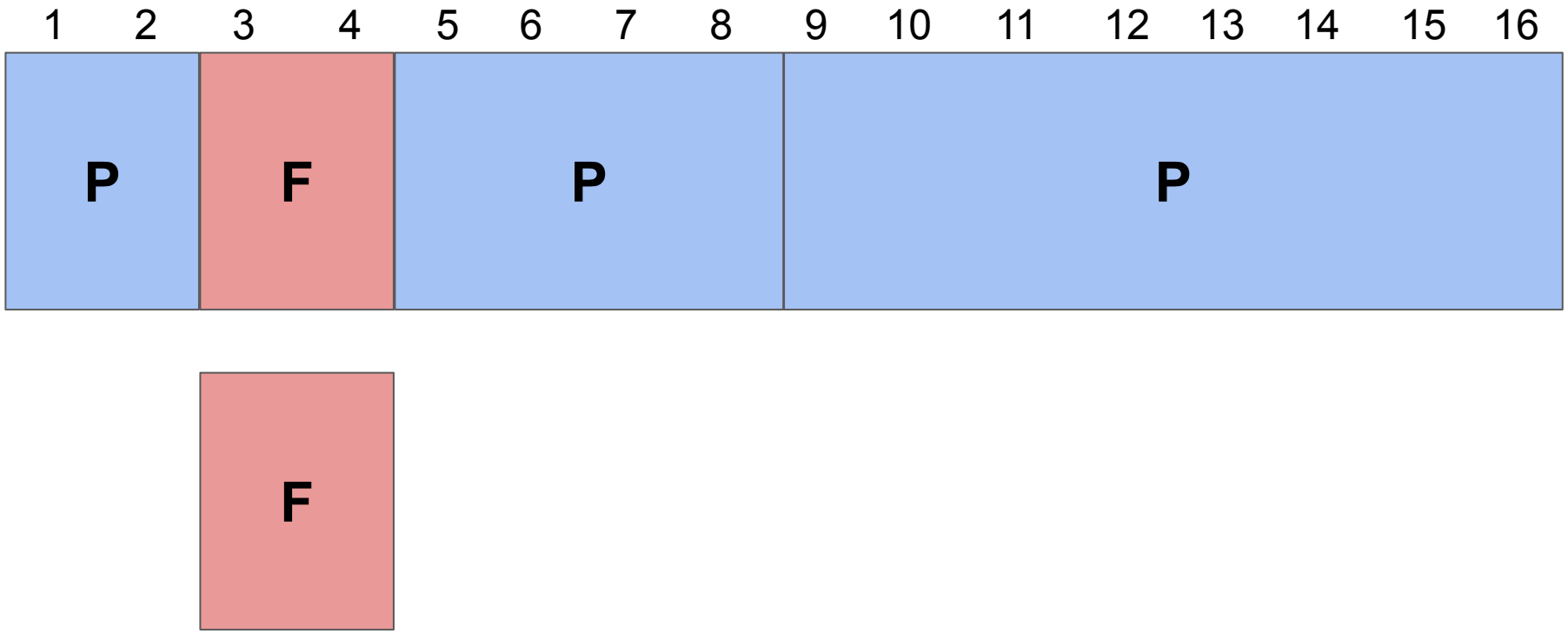
The happy path: binary search



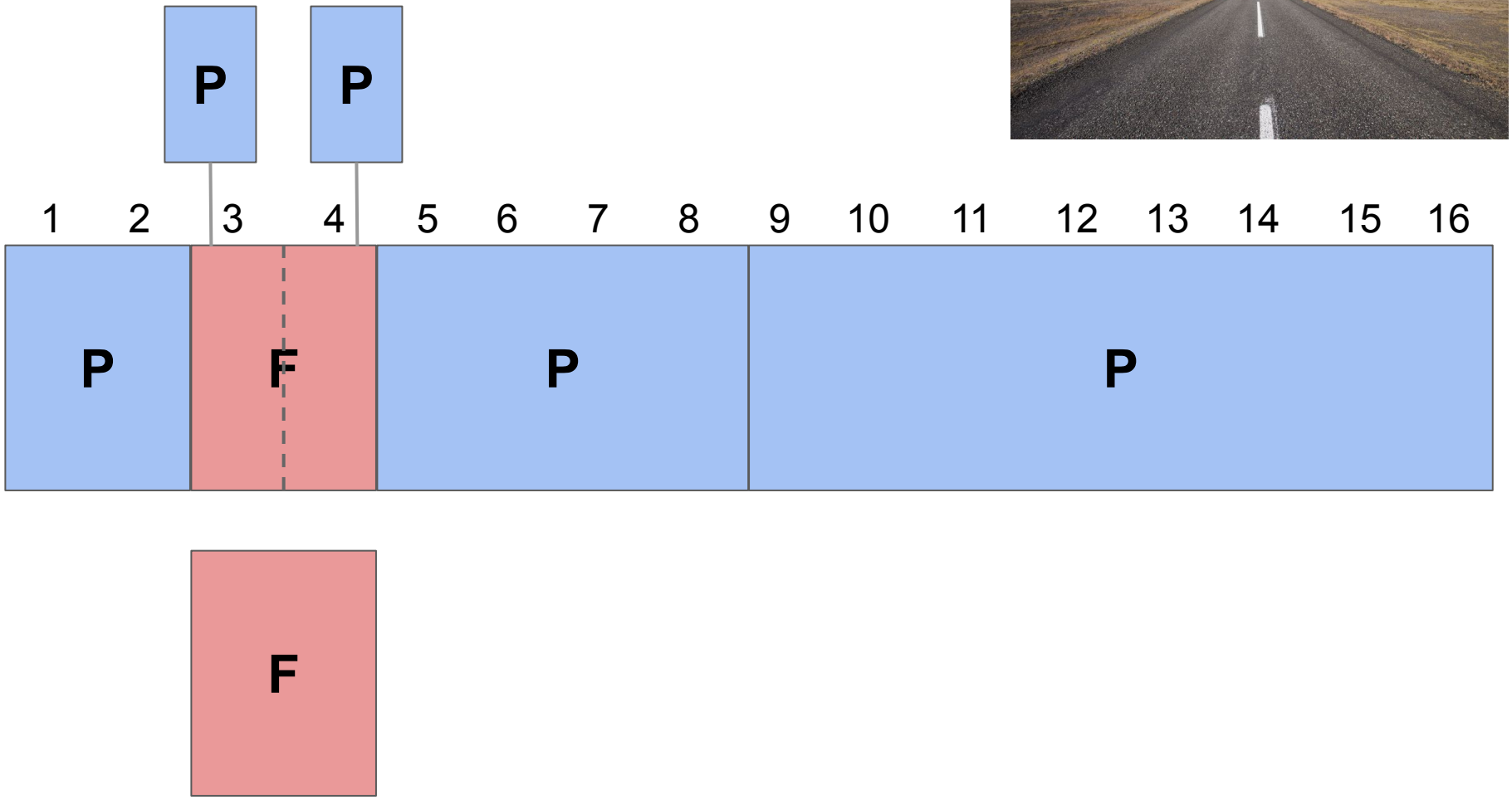
The happy path: binary search



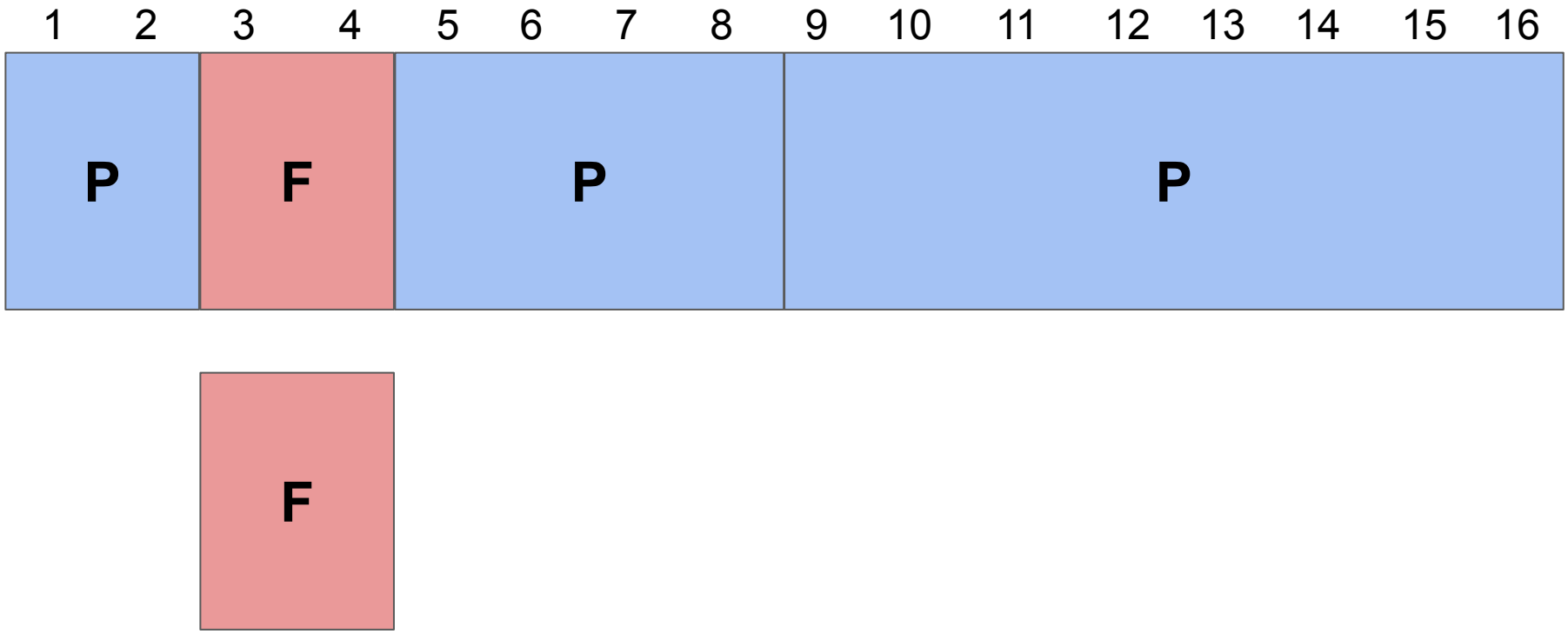
The happy path: binary search



The happy path: binary search

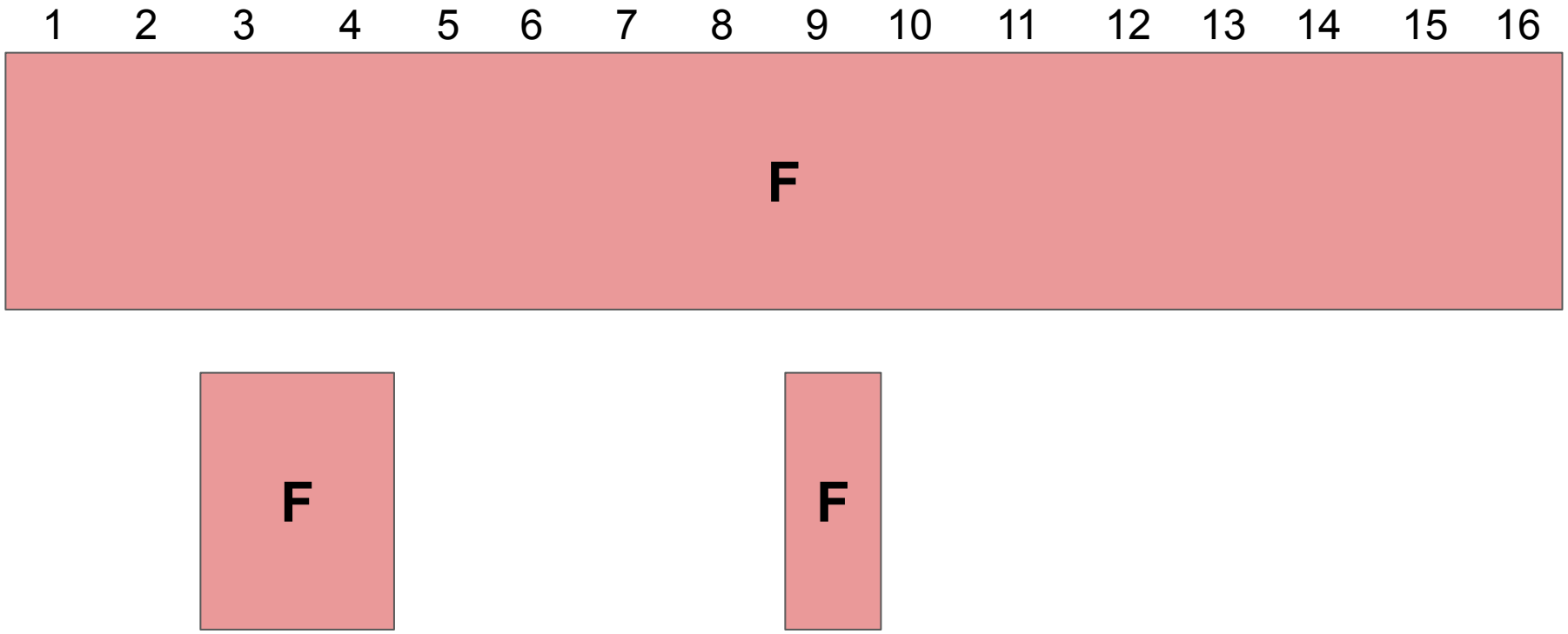


The happy path: binary search



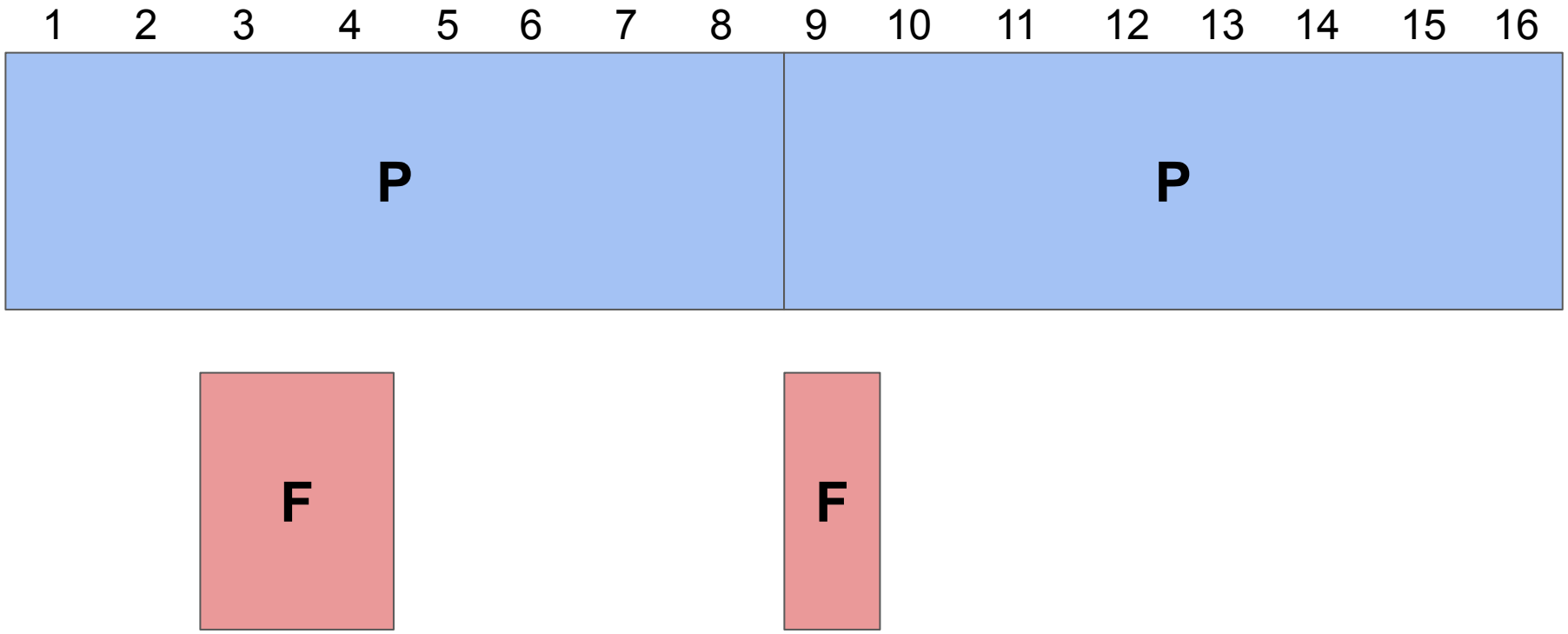
Successfully minimized the failing test to 2 lines

The not so happy path...



Suppose the failure pattern is more complex (all three lines must exist in a failing test case).

The not so happy path...



Binary search does not give optimal results.

Delta debugging: binary search + X

The DD algorithm

Minimizing Delta Debugging Algorithm

Let *test* and $c_{\text{✘}}$ be given such that $test(\emptyset) = \text{✔} \wedge test(c_{\text{✘}}) = \text{✘}$ hold.
The goal is to find $c'_{\text{✘}} = ddmin(c_{\text{✘}})$ such that $c'_{\text{✘}} \subseteq c_{\text{✘}}$, $test(c'_{\text{✘}}) = \text{✘}$, and $c'_{\text{✘}}$ is 1-minimal.
The *minimizing Delta Debugging algorithm* $ddmin(c)$ is

$$ddmin(c_{\text{✘}}) = ddmin_2(c_{\text{✘}}, 2) \quad \text{where}$$

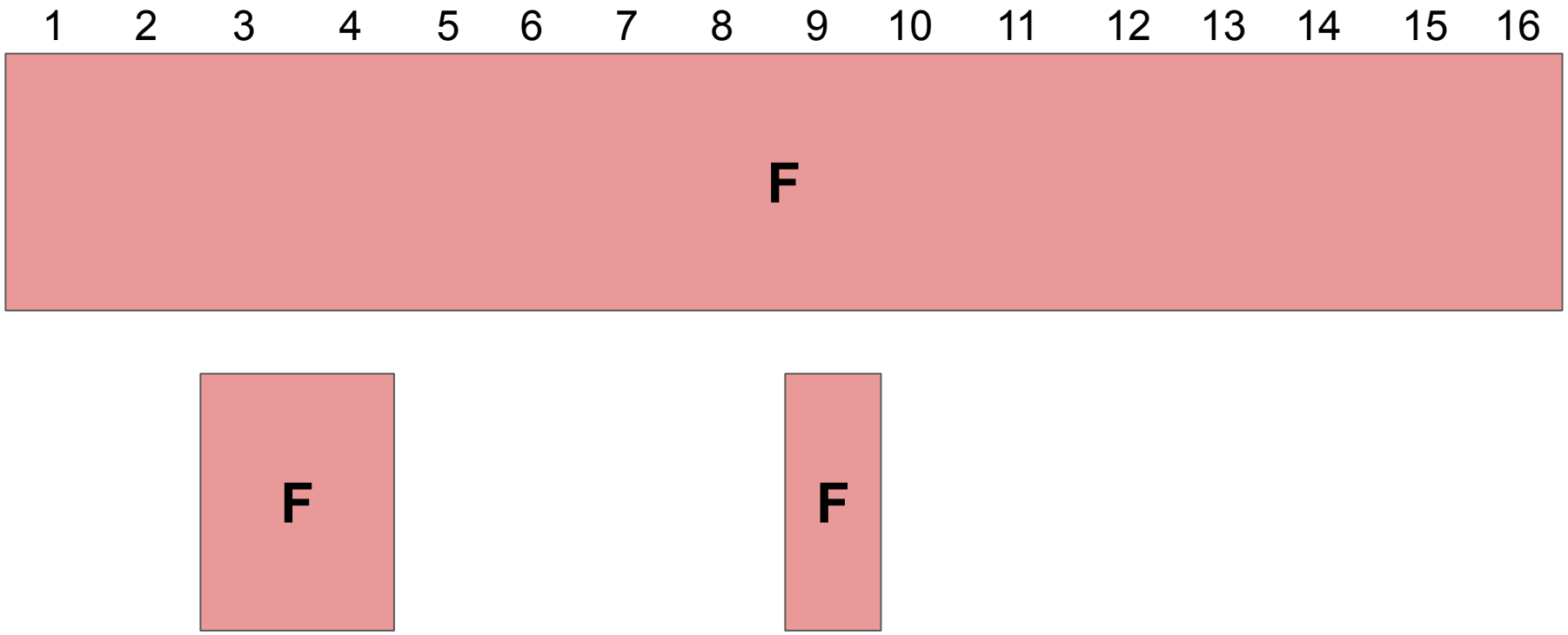
$$ddmin_2(c'_{\text{✘}}, n) = \begin{cases} ddmin_2(\Delta_i, 2) & \text{if } \exists i \in \{1, \ldots, n\} \cdot test(\Delta_i) = \text{✘} \text{ (“reduce to subset”)} \\ ddmin_2\big(\nabla_i, \max(n-1, 2)\big) & \text{else if } \exists i \in \{1, \ldots, n\} \cdot test(\nabla_i) = \text{✘} \text{ (“reduce to complement”)} \\ ddmin_2\big(c'_{\text{✘}}, \min(|c'_{\text{✘}}|, 2n)\big) & \text{else if } n < |c'_{\text{✘}}| \text{ (“increase granularity”)} \\ c'_{\text{✘}} & \text{otherwise (“done”).} \end{cases}$$

where $\nabla_i = c'_{\text{✘}} - \Delta_i$, $c'_{\text{✘}} = \Delta_1 \cup \Delta_2 \cup \cdots \cup \Delta_n$, all Δ_i are pairwise disjoint, and $\forall \Delta_i \cdot |\Delta_i| \approx |c'_{\text{✘}}|/n$ holds.
The recursion invariant (and thus precondition) for $ddmin_2$ is $test(c'_{\text{✘}}) = \text{✘} \wedge n \leq |c'_{\text{✘}}|$.

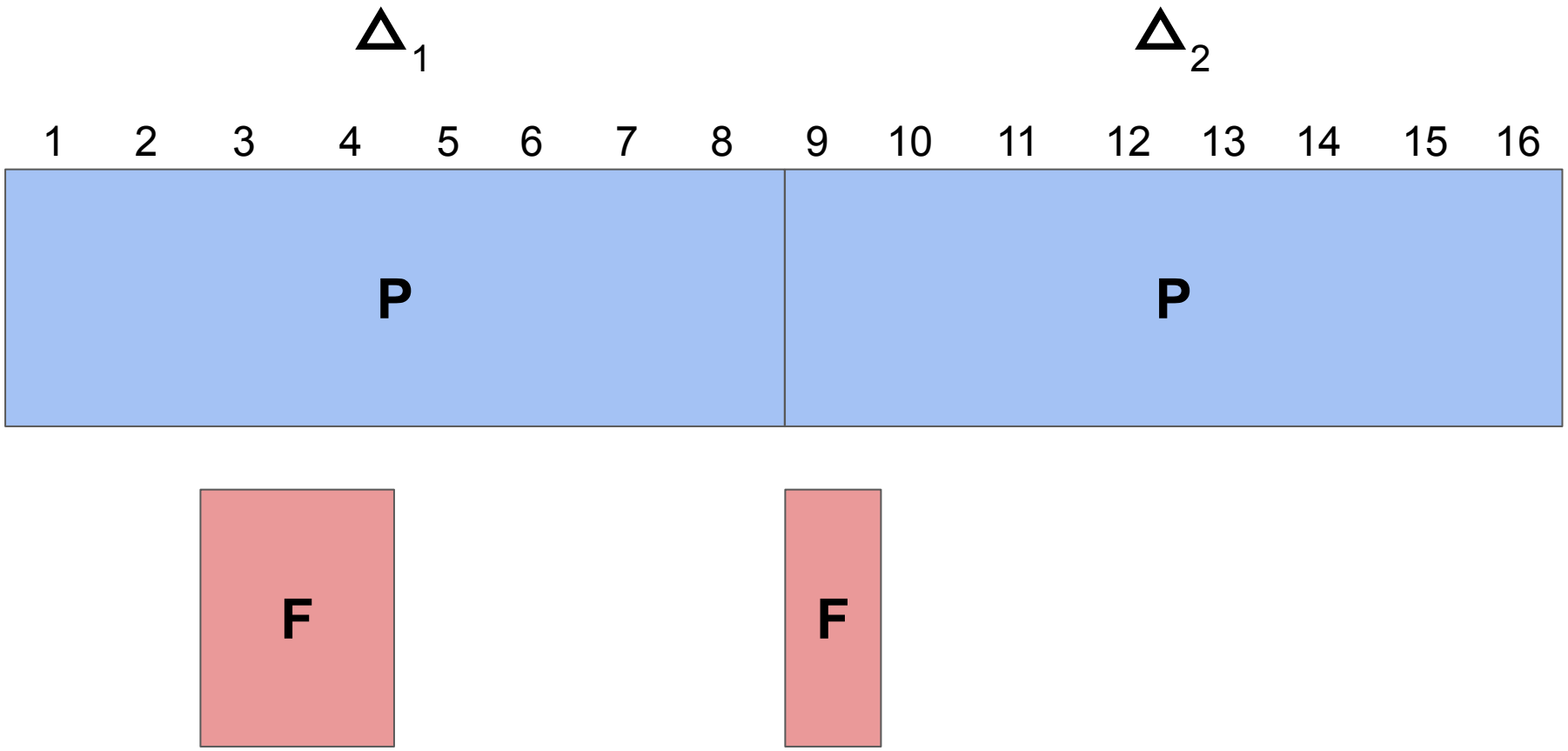
Four basic steps:

1. Test each subset
2. Test each complement
3. Increase granularity
4. Reduce

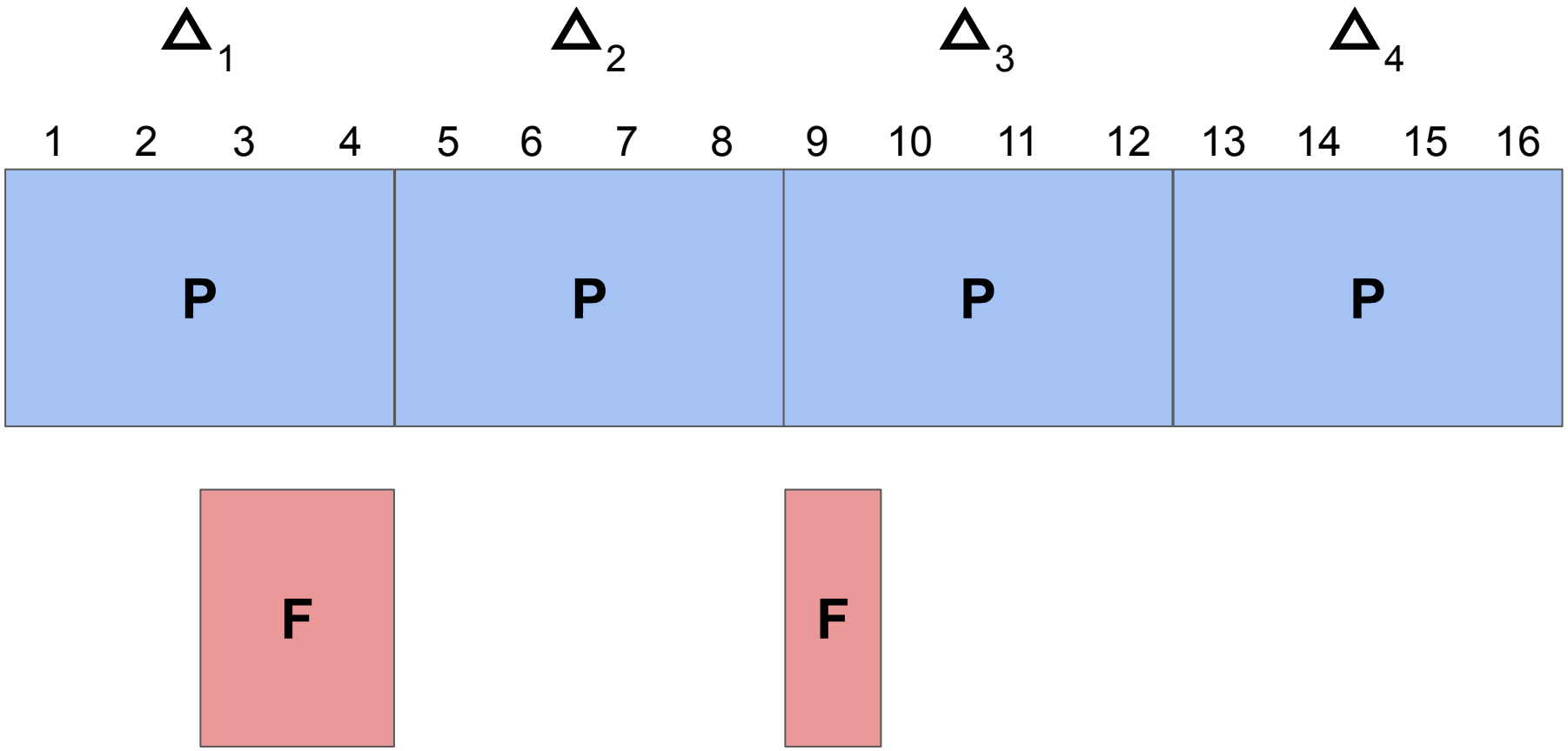
Delta Debugging: mostly binary search



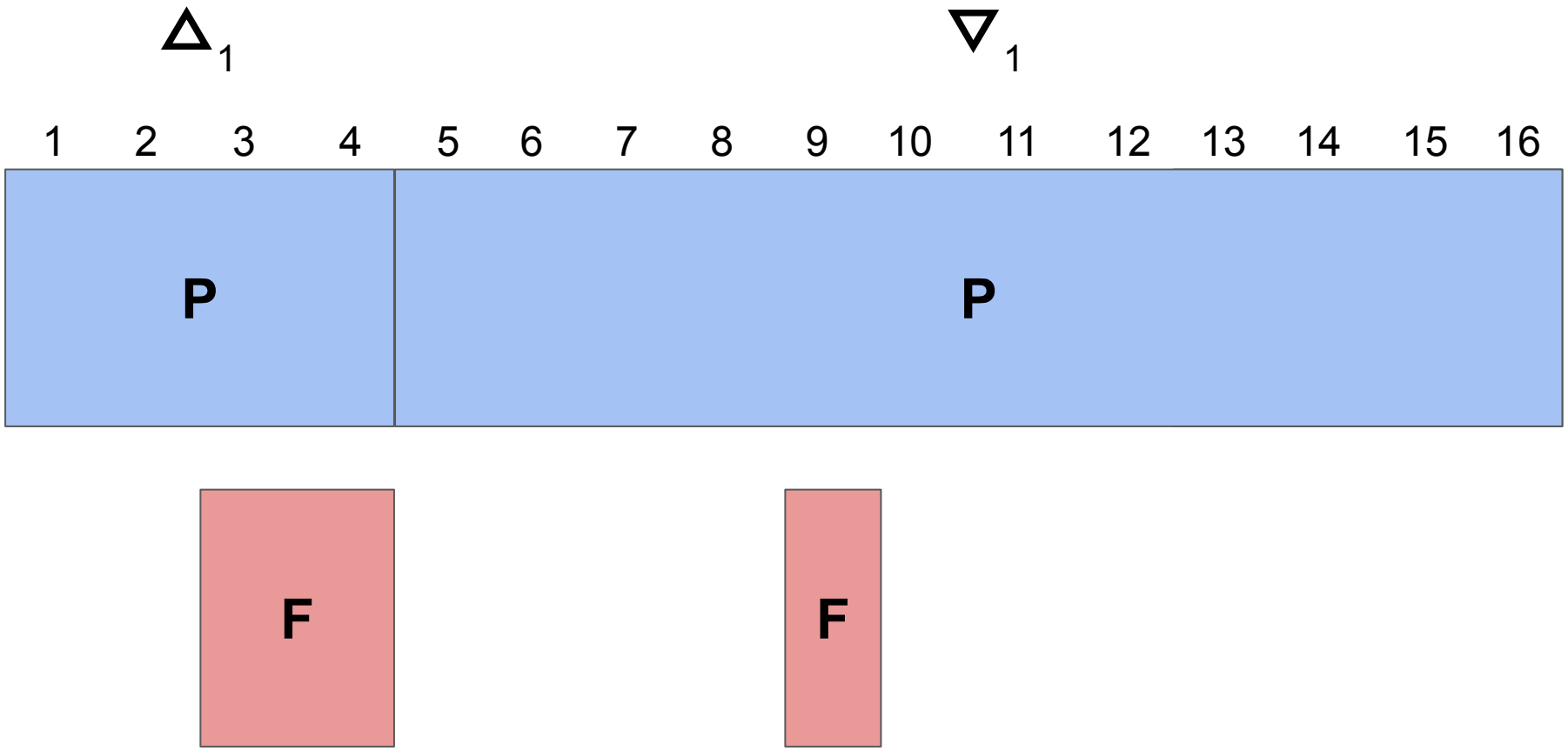
Delta Debugging: mostly binary search



Delta Debugging: granularity

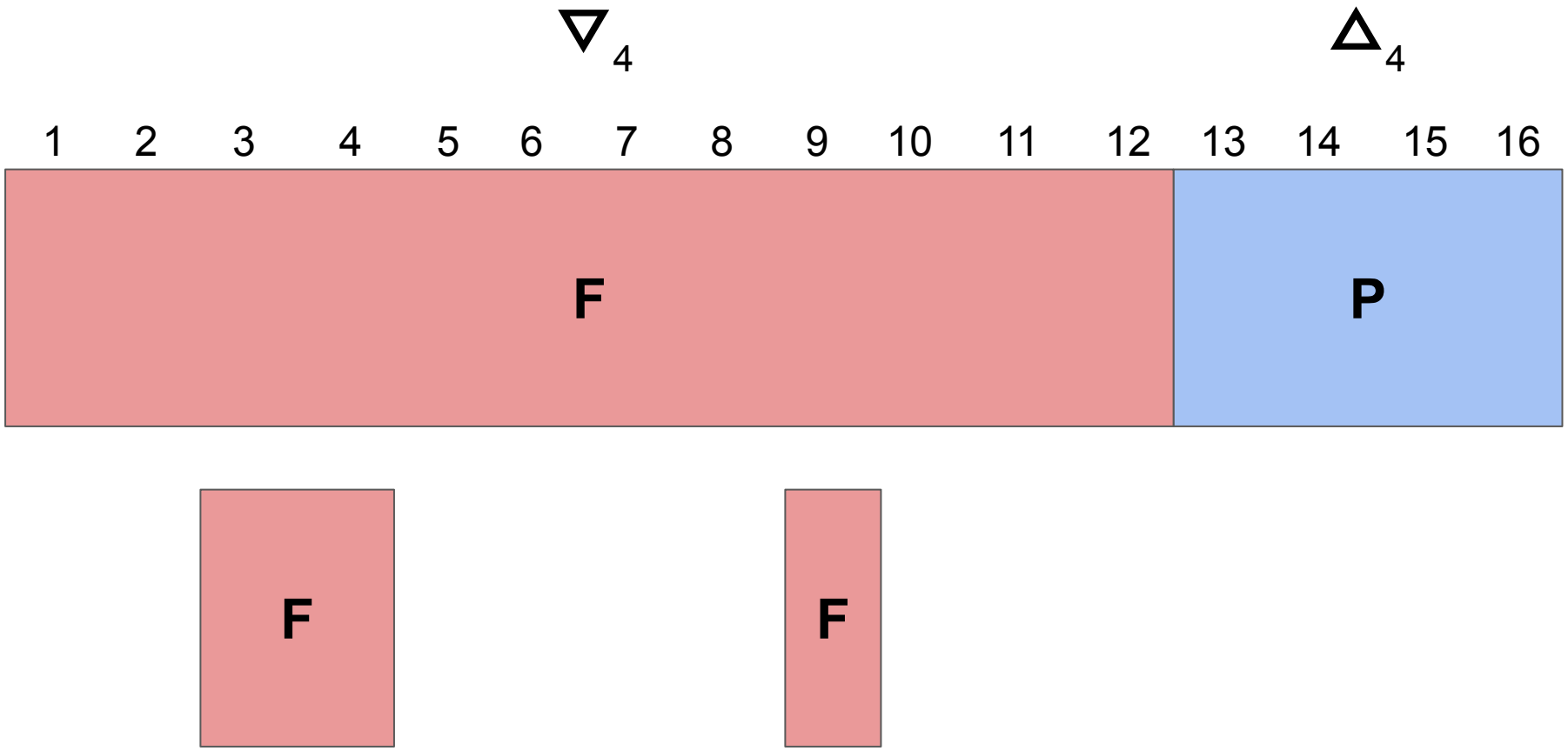


Delta Debugging: complements

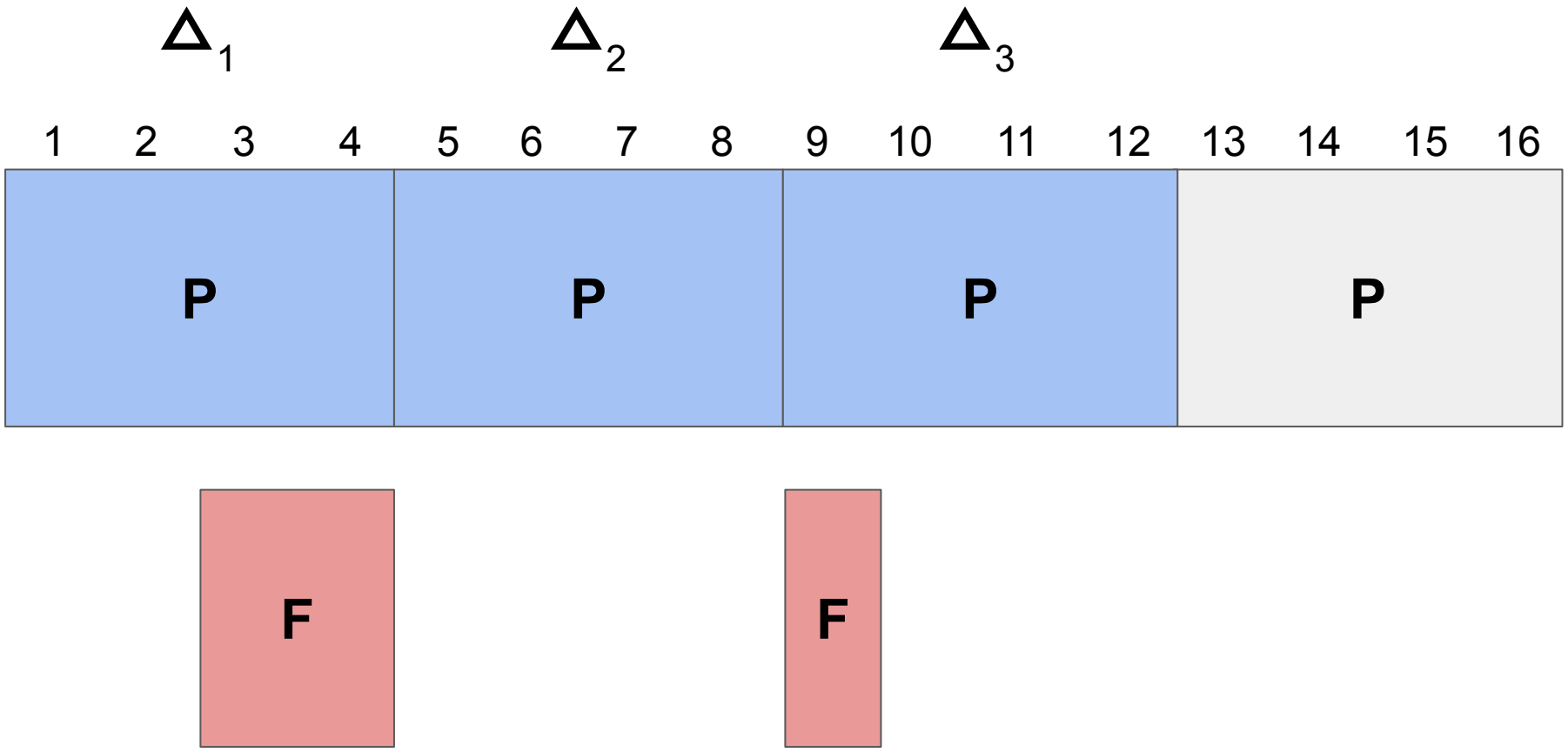


The order in which deltas and complements are evaluated may differ between implementations of the algorithm.

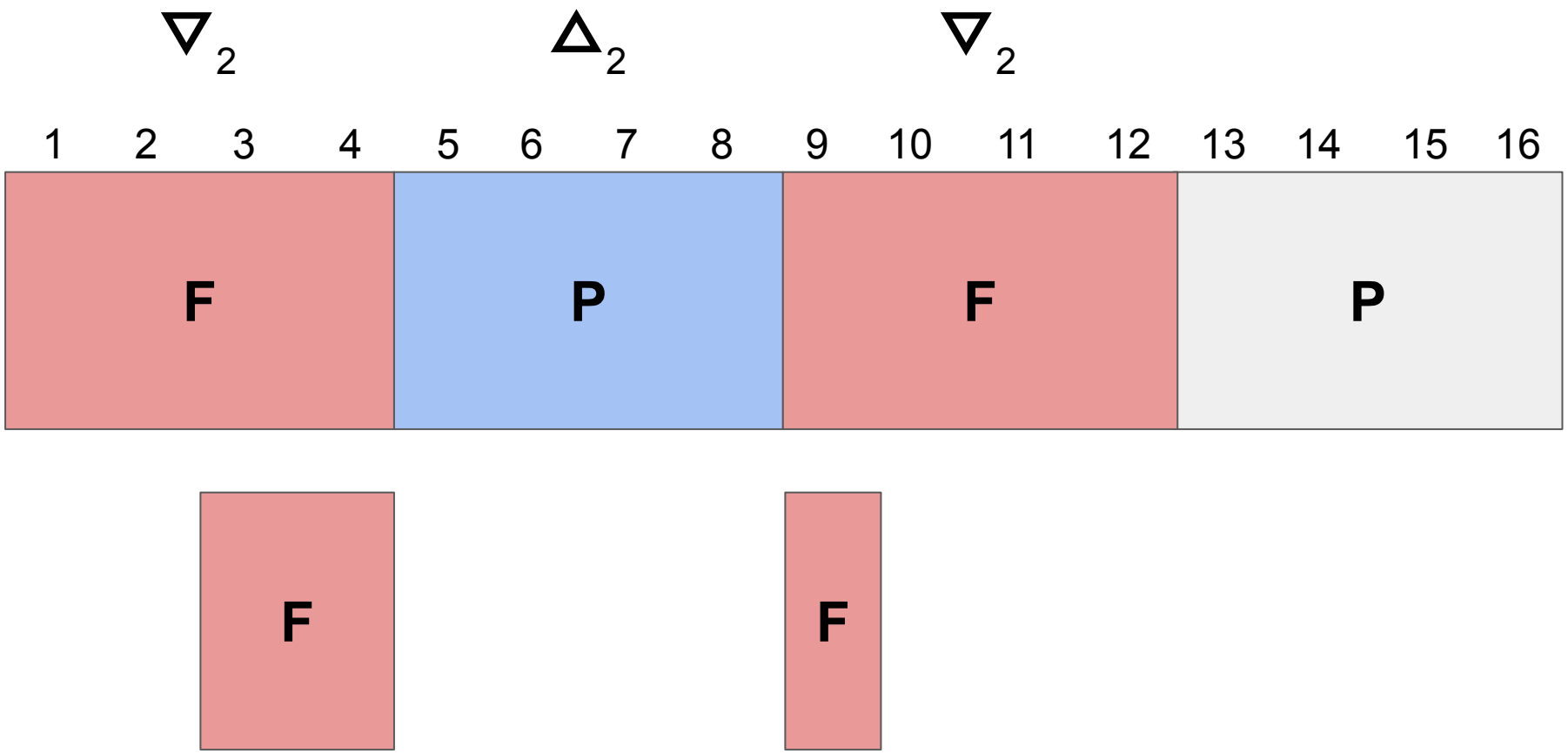
Delta Debugging: complements



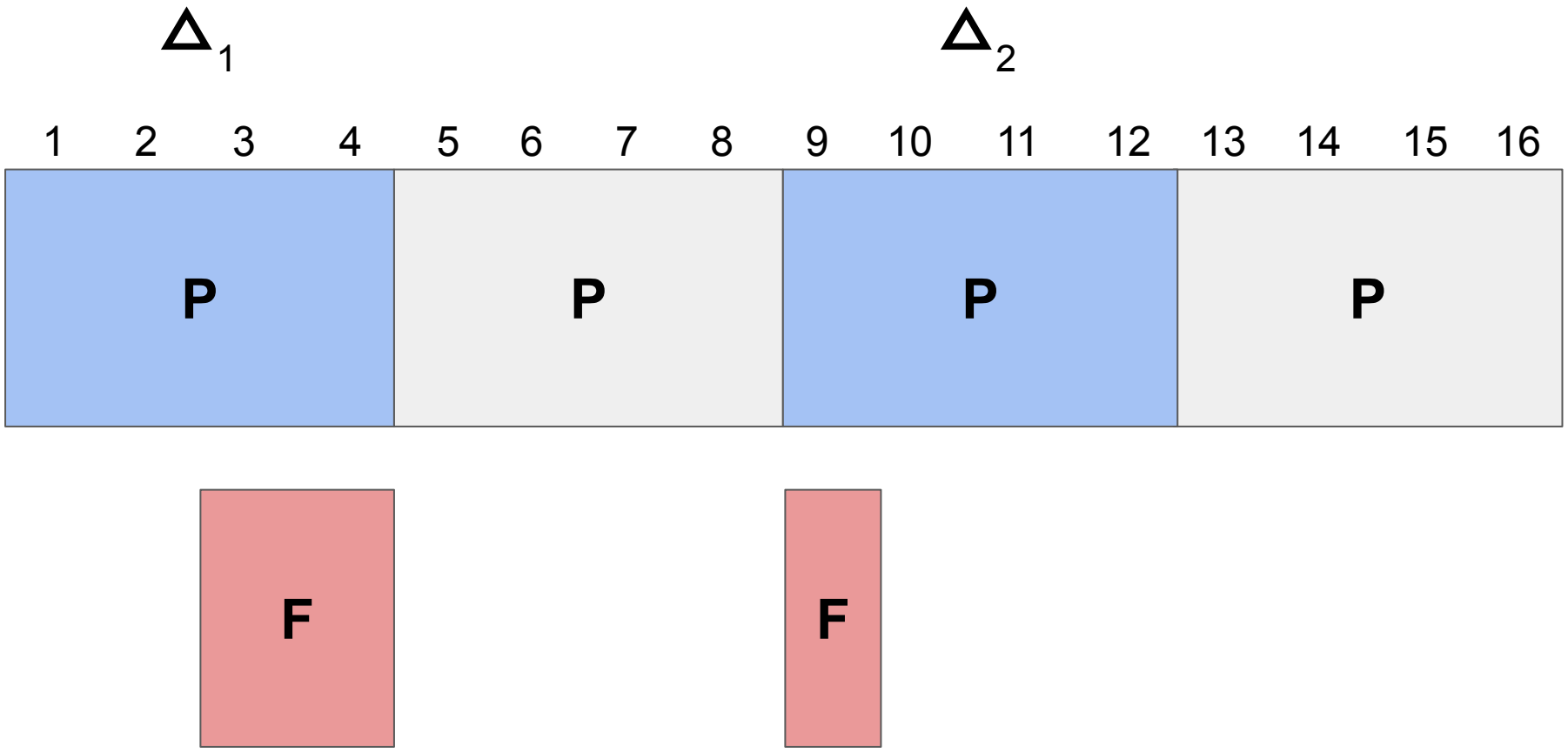
Delta Debugging: reduce



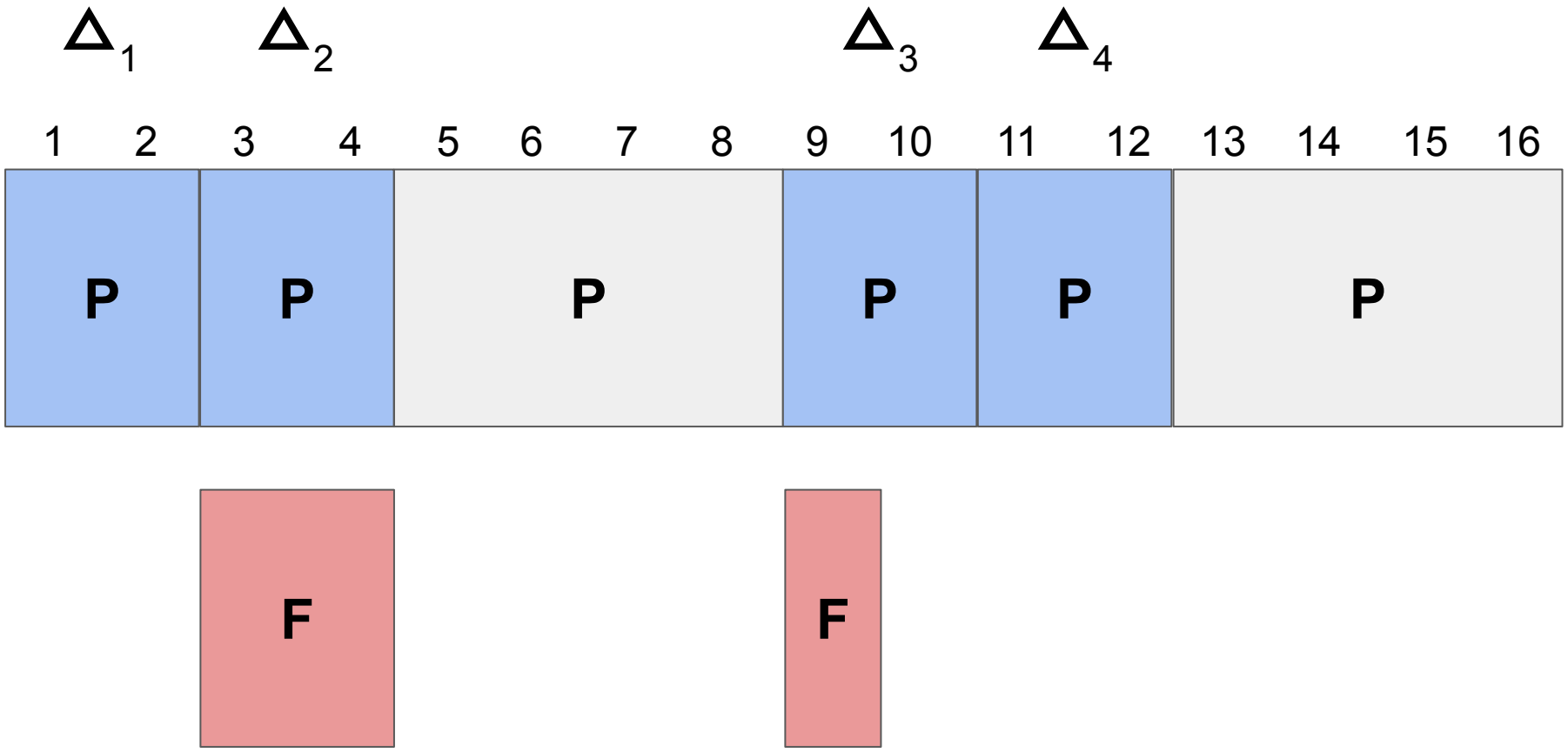
Delta Debugging: complements



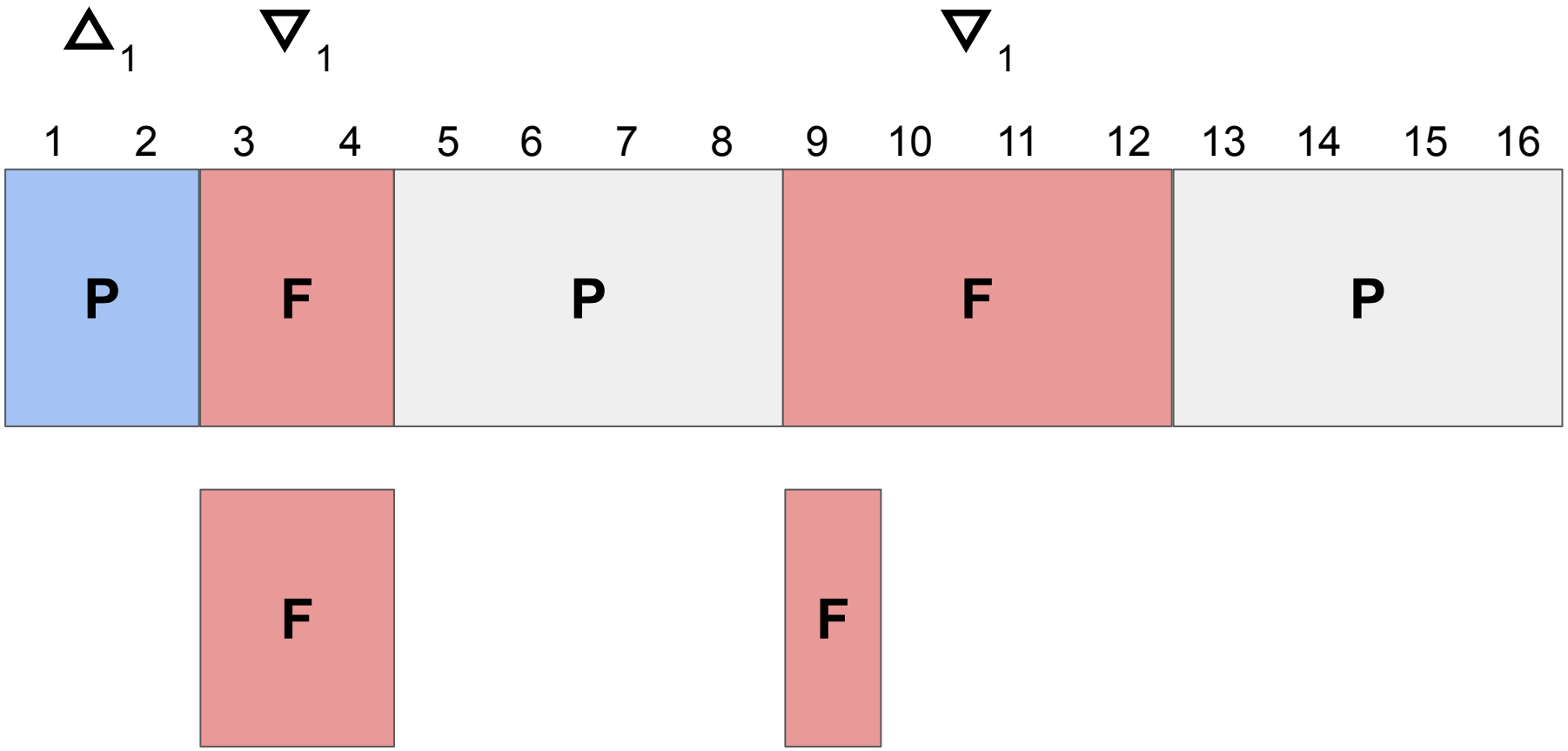
Delta Debugging: reduce



Delta Debugging: granularity

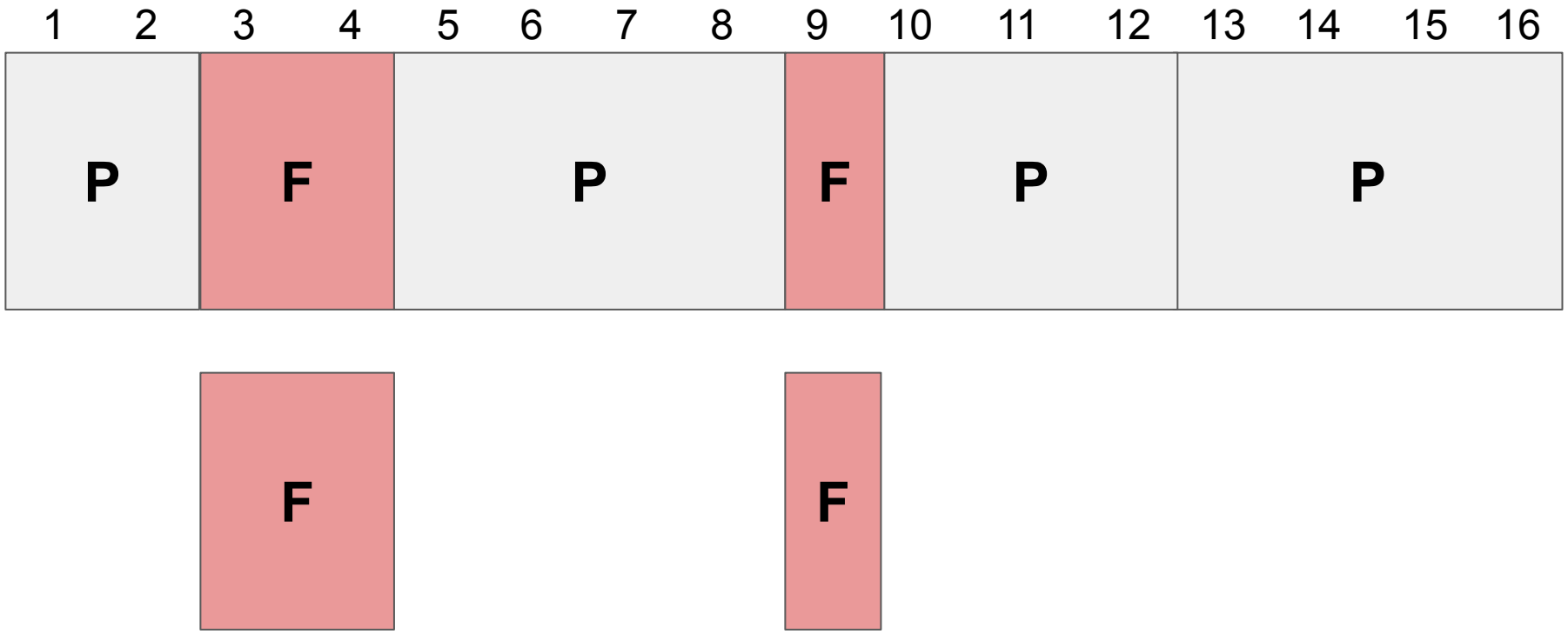


Delta Debugging: complements



And so on...

Delta Debugging: 1-minimality



Failing test cases must be deterministic and monotone.

Delta debugging: live example

Delta Debugging: live example

Program and initial test case

- Program ***P*** **crashes** whenever the input contains **`1 7 8`**
- **Initial** crashing **test case** is: **`1 2 3 4 5 6 7 8`**

Recall the basic approach:

1. Test each subset
2. Test each complement
3. Increase granularity
4. Reduce

Example taken from this week's reading.

Delta debugging: exercise

A little exercise

Program and initial test case

- Program *P* takes as **input a list of integers** l.
- ***P* crashes** whenever l **contains 4,2**.
- **Initial** crashing **test case** is: **`2,4,2,4`**

Complete the following table

Iteration	Granularity (n)	Input	$\Delta_1, \ldots, \Delta_n$ $\nabla_1, \ldots, \nabla_n$
1	2	2,4,2,4	2,4; (2,4)
2	4	…	…
…	…	…	…

https://forms.gle/YMCv1fyquUiNui6YA

A little exercise

Program and initial test case

- Program *P* takes as **input a list of integers** *l*.
- ***P* crashes** whenever *l* **contains 4,2**.
- **Initial** crashing **test case** is: **`2,4,2,4`**

Complete the following table

Iteration	Granularity (n)	Input	$\Delta_1, \ldots, \Delta_n$ $\nabla_1, \ldots, \nabla_n$
1	2	2,4,2,4	2,4; (2,4)
2	4	2,4,2,4	2; 4; (2); (4); **4,2,4**; (2,2,4); (2,4,4); (2,4,2)
3	3	4,2,4	(4); (2); (4); (2,4); 4,4; **4,2**
4	2	4,2	(4); (2)