Greedy Approach

Md. Tanvir Rahman

Lecturer, Dept. of ICT

Mawlana Bhashani Science and Technology University

Greedy Algorithm

- Greedy algorithms make the choice that looks best at the moment.
- This locally optimal choice may lead to a globally optimal solution (i.e. an optimal solution to the entire problem).

Greedy Algorithm

- A *greedy algorithm* always makes the choice that looks best at the moment
  - Some examples:
    - Playing cards
    - Invest on stocks
    - Choose a university
  - The hope: a locally optimal choice will lead to a globally optimal solution
  - For some problems, it works
- greedy algorithms tend to be easier to code

Greedy vs Dynamic Programming

Dynamic Programming	Greedy Approach
• At each step, the choice is determined based on the solutions of sub problems	• At each step, we quickly make a choice that currently looks best. A local greedy choice.
• It is slower	• It is faster
• It is more complex	• It is simple

Example: Making Change

- Instance: amount (in cents) to return to customer
- Problem: do this using fewest number of coins
- Example:
  - Assume that we have an unlimited number of coins of various denominations:
    - 1c (pennies), 5c (nickels), 10c (dimes), 25c (quarters), 1$ (loonies)
  - Objective: Pay out a given sum $5.64 with the smallest number of coins possible.

Example: Making Change

- E.g.:

$5.64 = $1 + $1 + $1 + $1 + $1 +

.25 + .25 + .10 +

.01 + .01 + .01 + .01

An Activity Selection Problem (Conference Scheduling Problem)

- **Input: A set of activities S = {a_1,...., a_n}**
- Each activity has start time and a finish time
  - $A_i = (s_i, f_i)$
- Two activities are compatible if and only if their interval does not overlap
- **Output: a maximum-size subset of mutually compatible activities**

The Activity Selection Problem

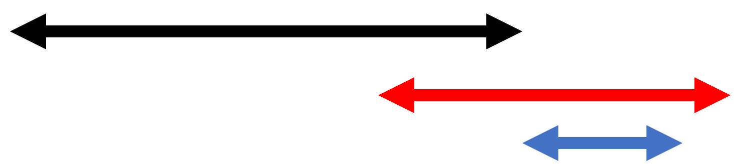
- Here are a set of start and finish times

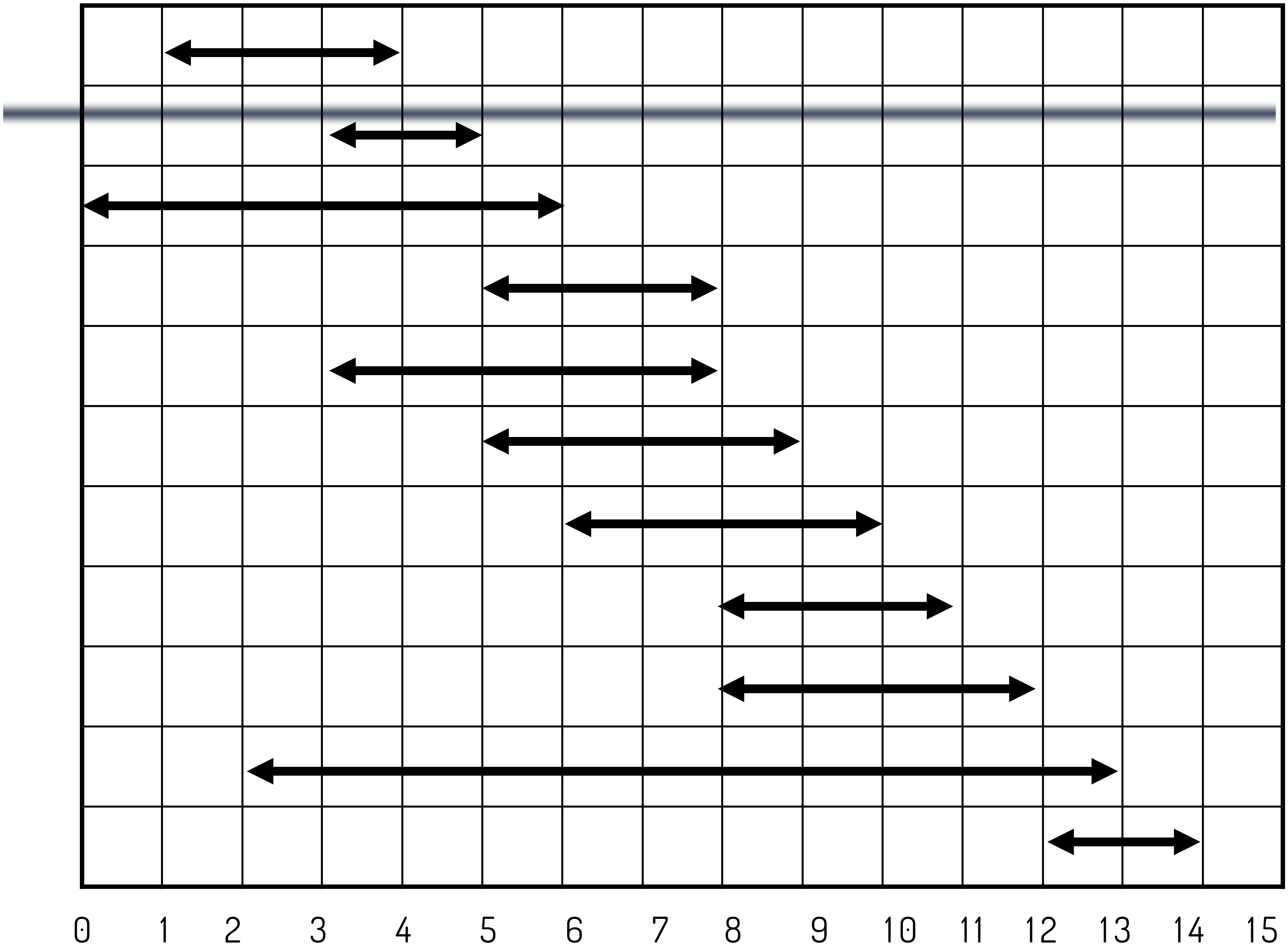
i	1	2	3	4	5	6	7	8	9	10	11
s_i	1	3	0	5	3	5	6	8	8	2	12
f_i	4	5	6	7	8	9	10	11	12	13	14

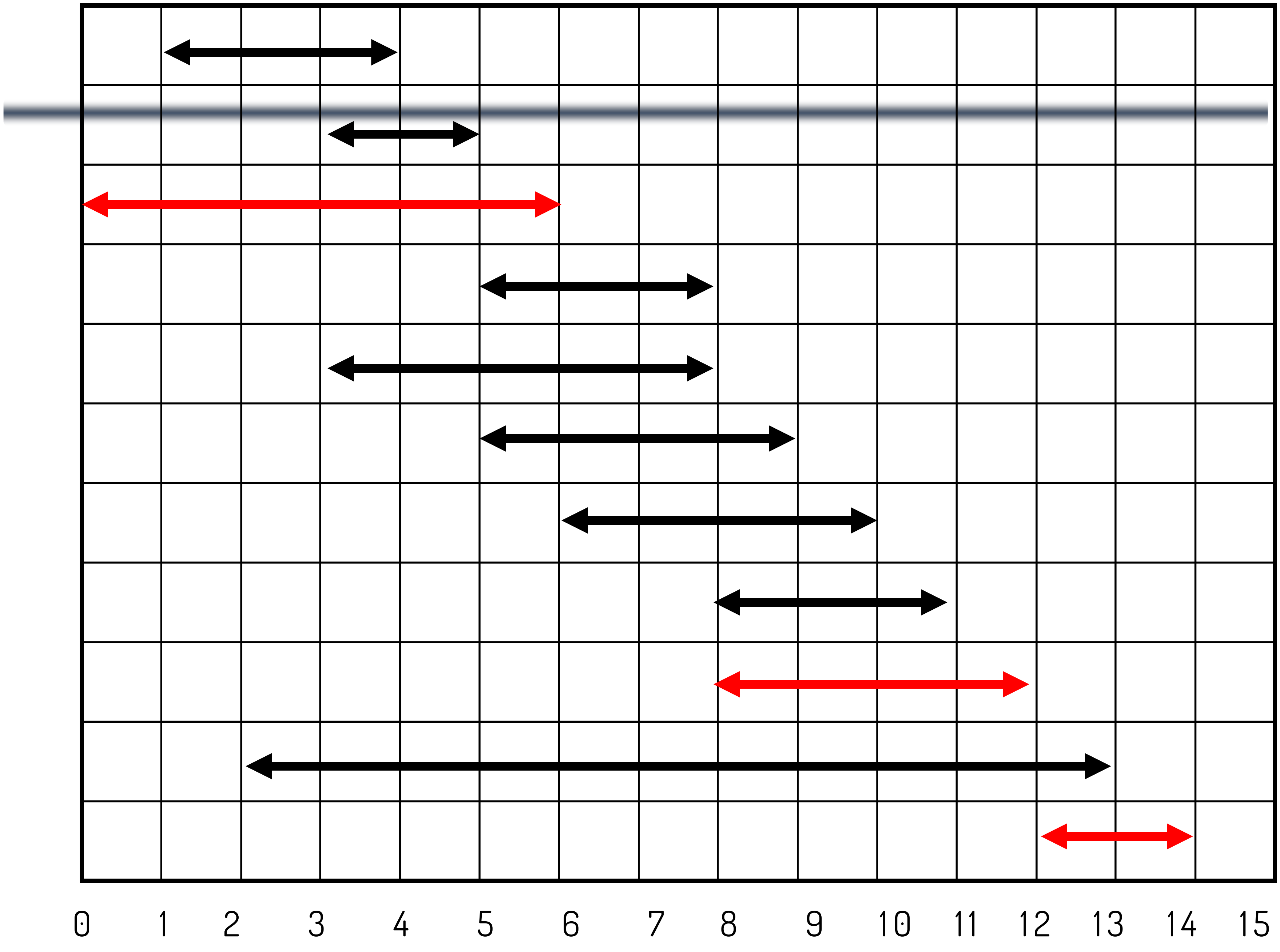
- What is the maximum number of activities that can be completed?
  - $\{a_3, a_7, a_{11}\}$ can be completed
  - $\{a_3, a_8, a_{11}\}$ can be completed
  - $\{a_3, a_9, a_{11}\}$ can be completed
  - But so can $\{a_1, a_4, a_8, a_{11}\}$ which is a larger set
  - But it is not unique, consider $\{a_2, a_4, a_9, a_{11}\}$

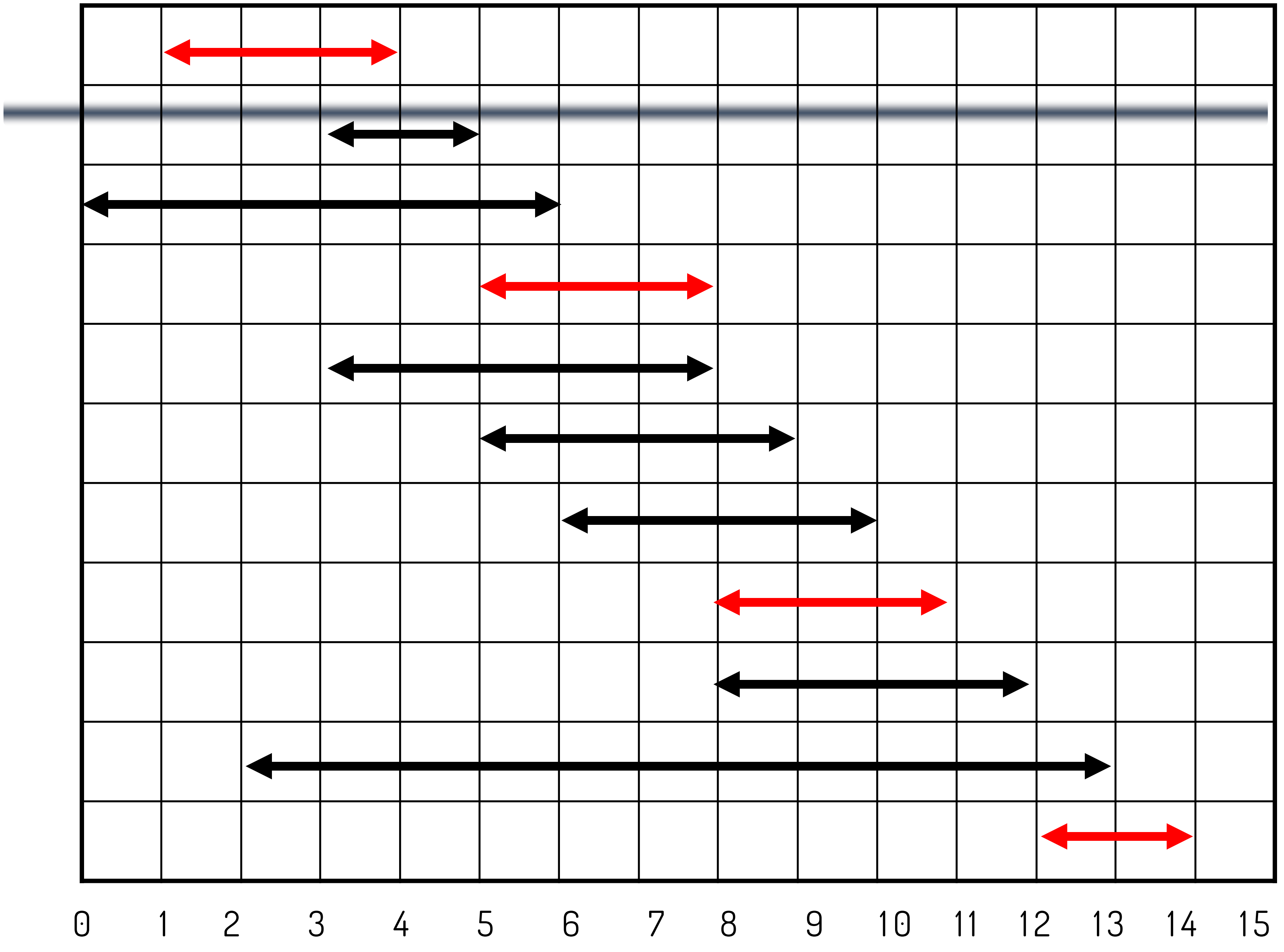
Interval Representation

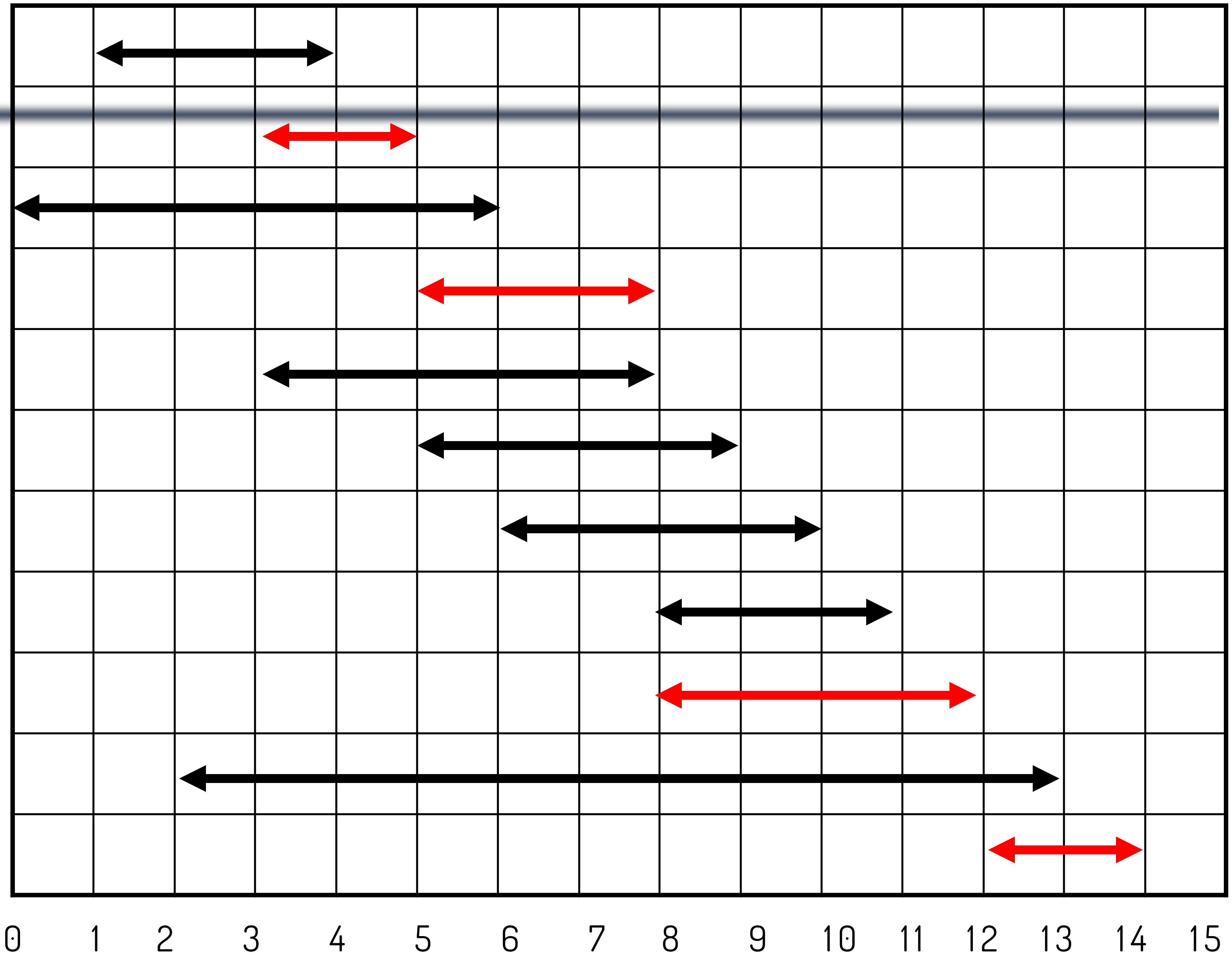
i	1	2	3	4	5	6	7	8	9	10	11
s_i	1	3	0	5	3	5	6	8	8	2	12
f_i	4	5	6	7	8	9	10	11	12	13	14




0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

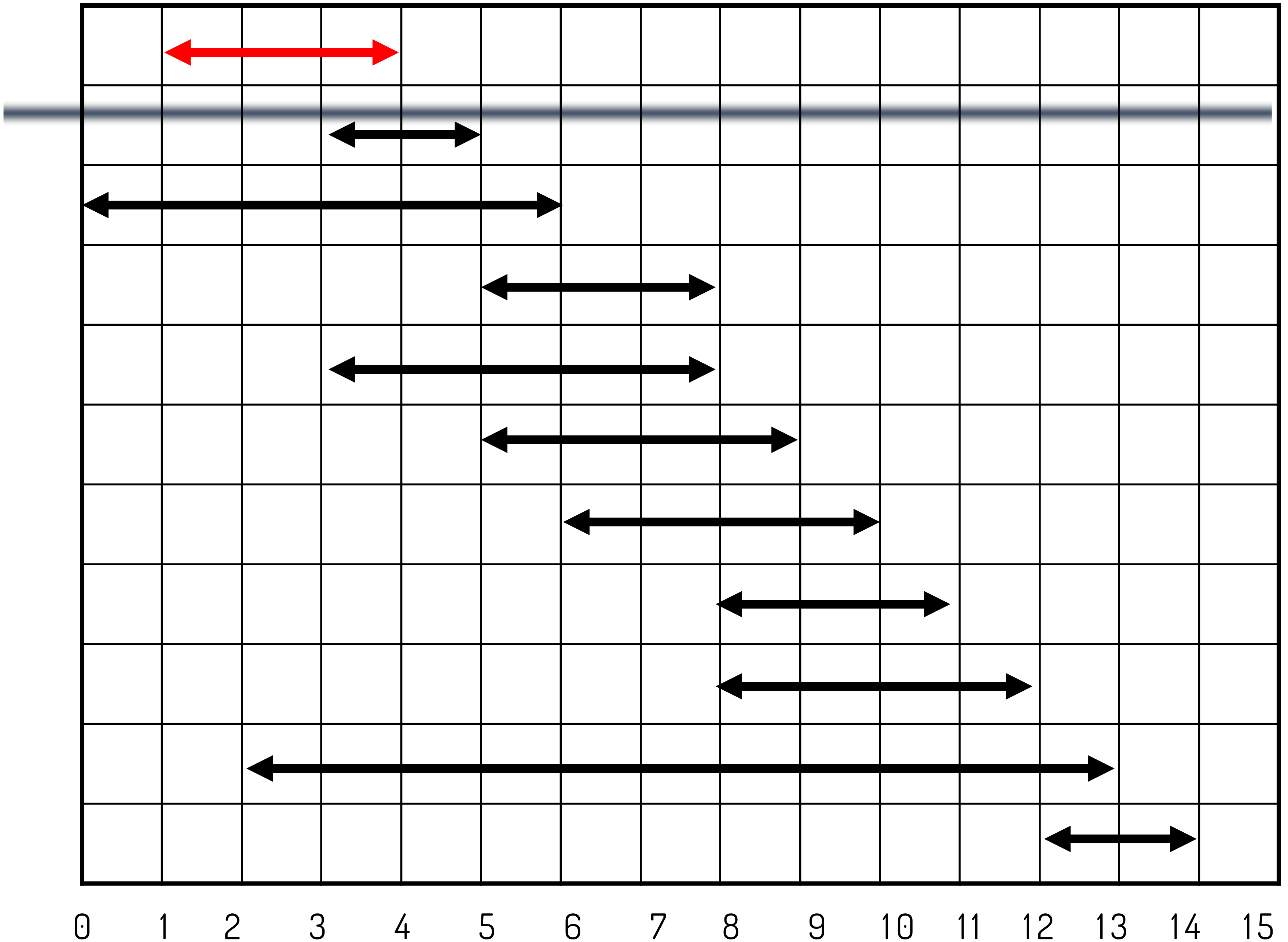

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

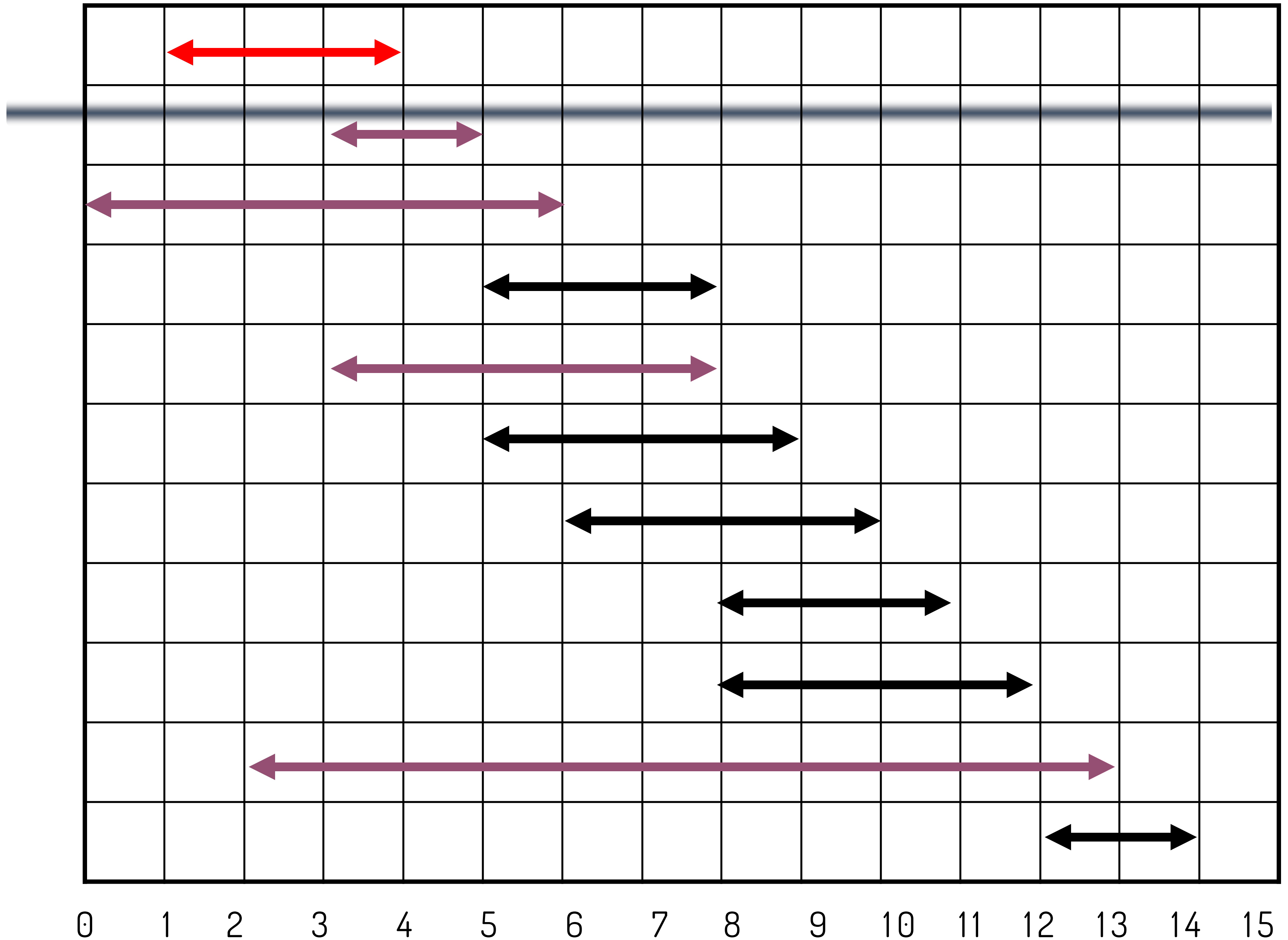

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

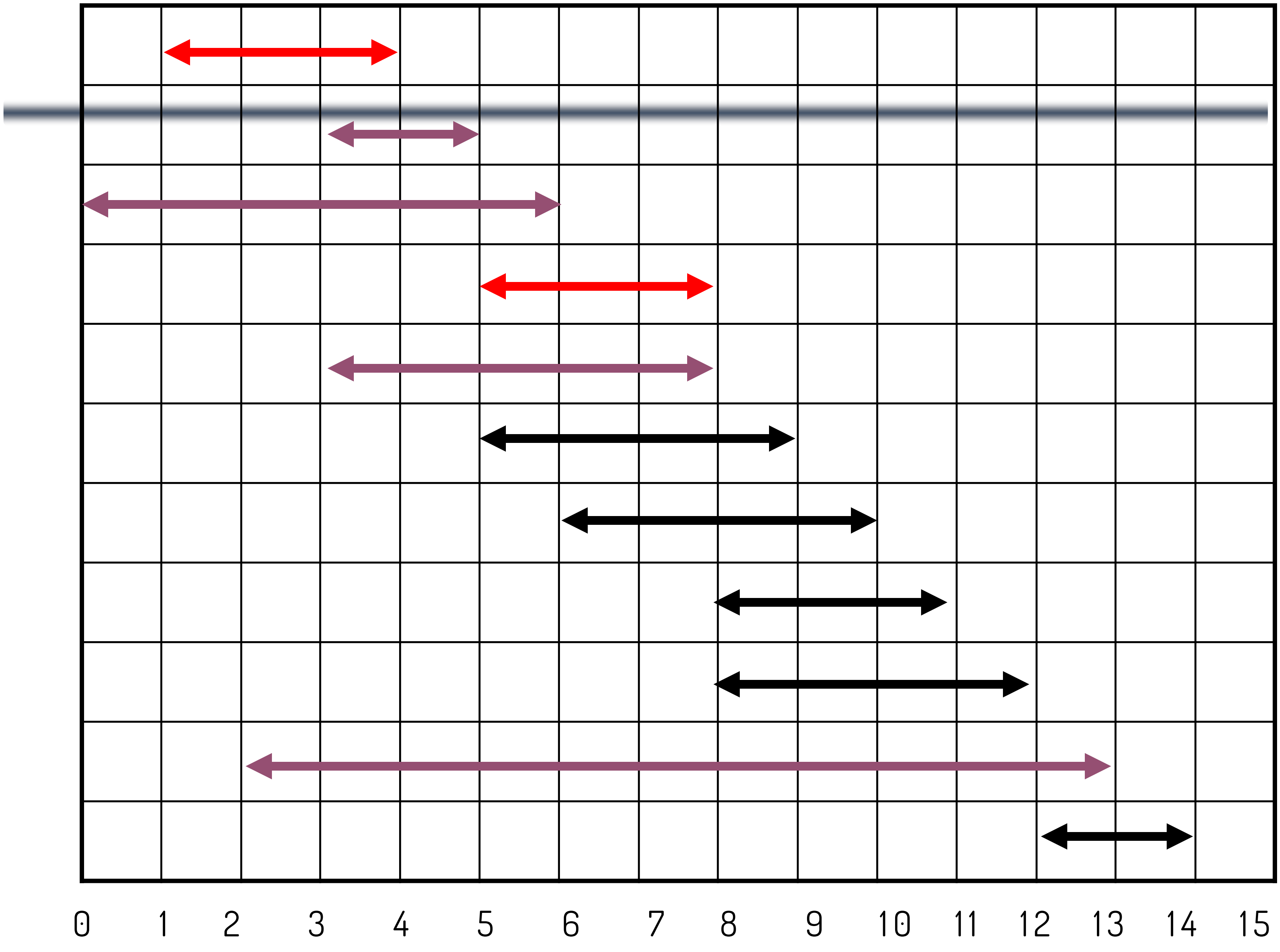

0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15

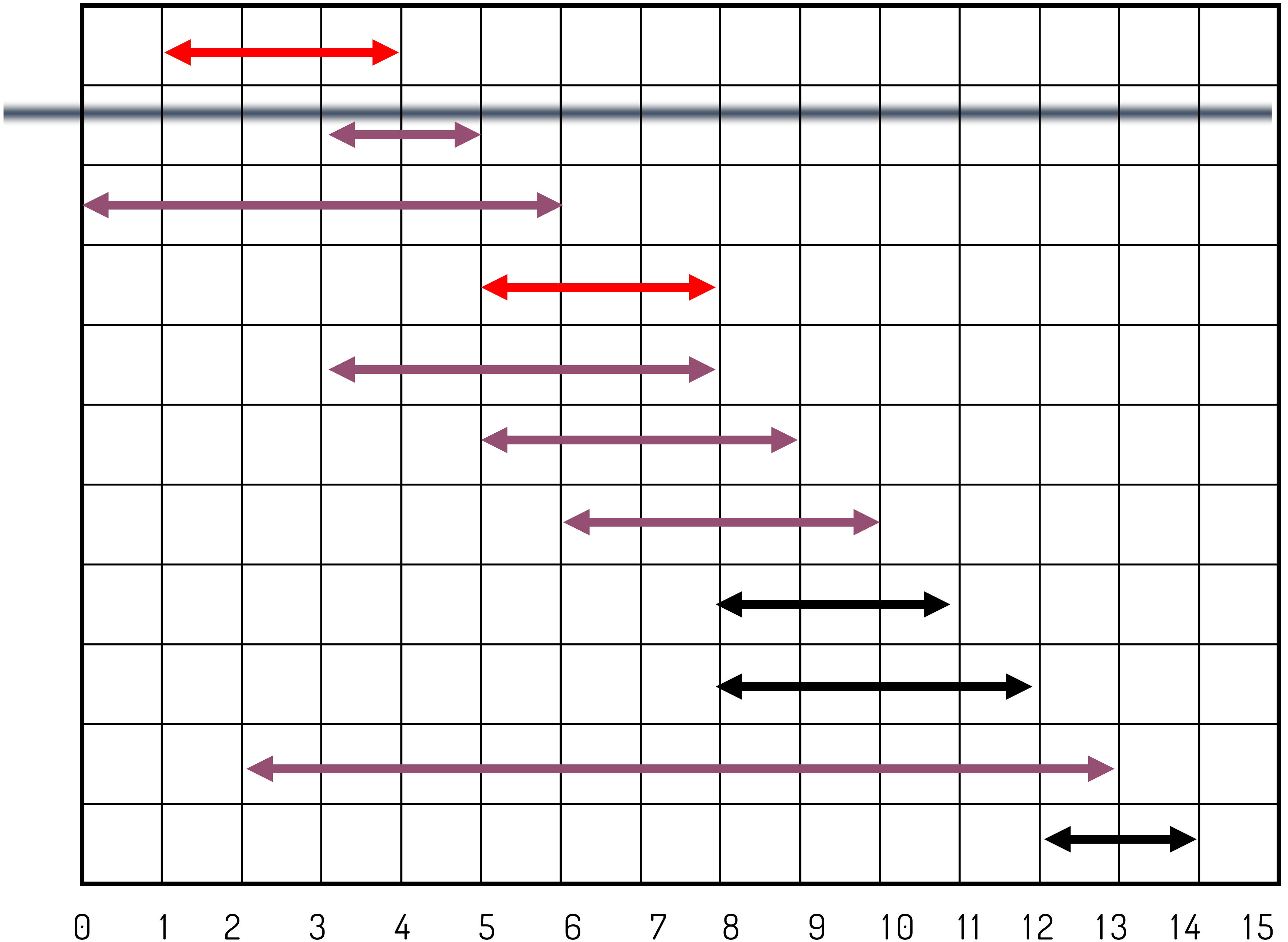
Early Finish Greedy

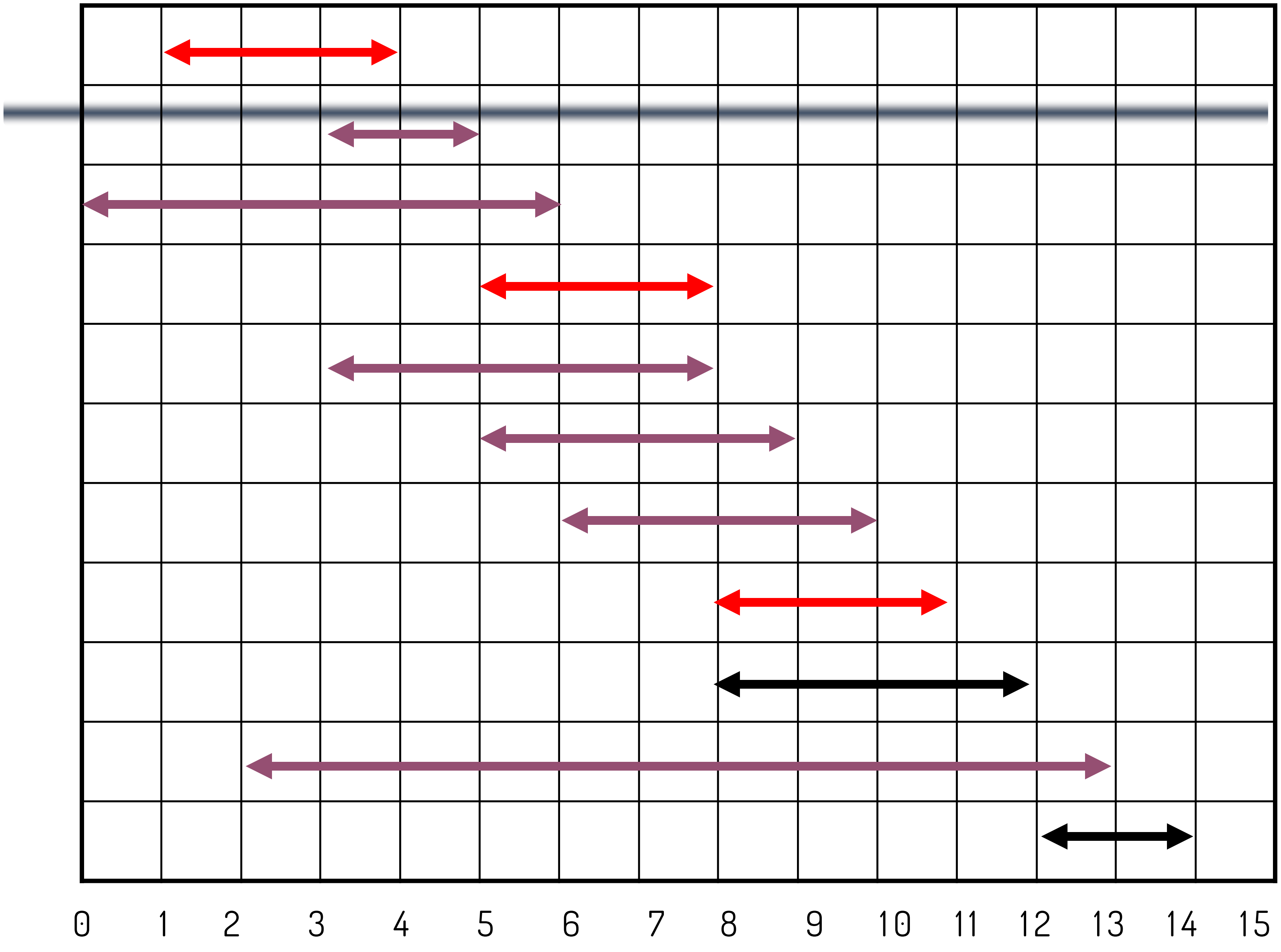
- Select the activity with the earliest finish
- Eliminate the activities that could not be scheduled
- Repeat!

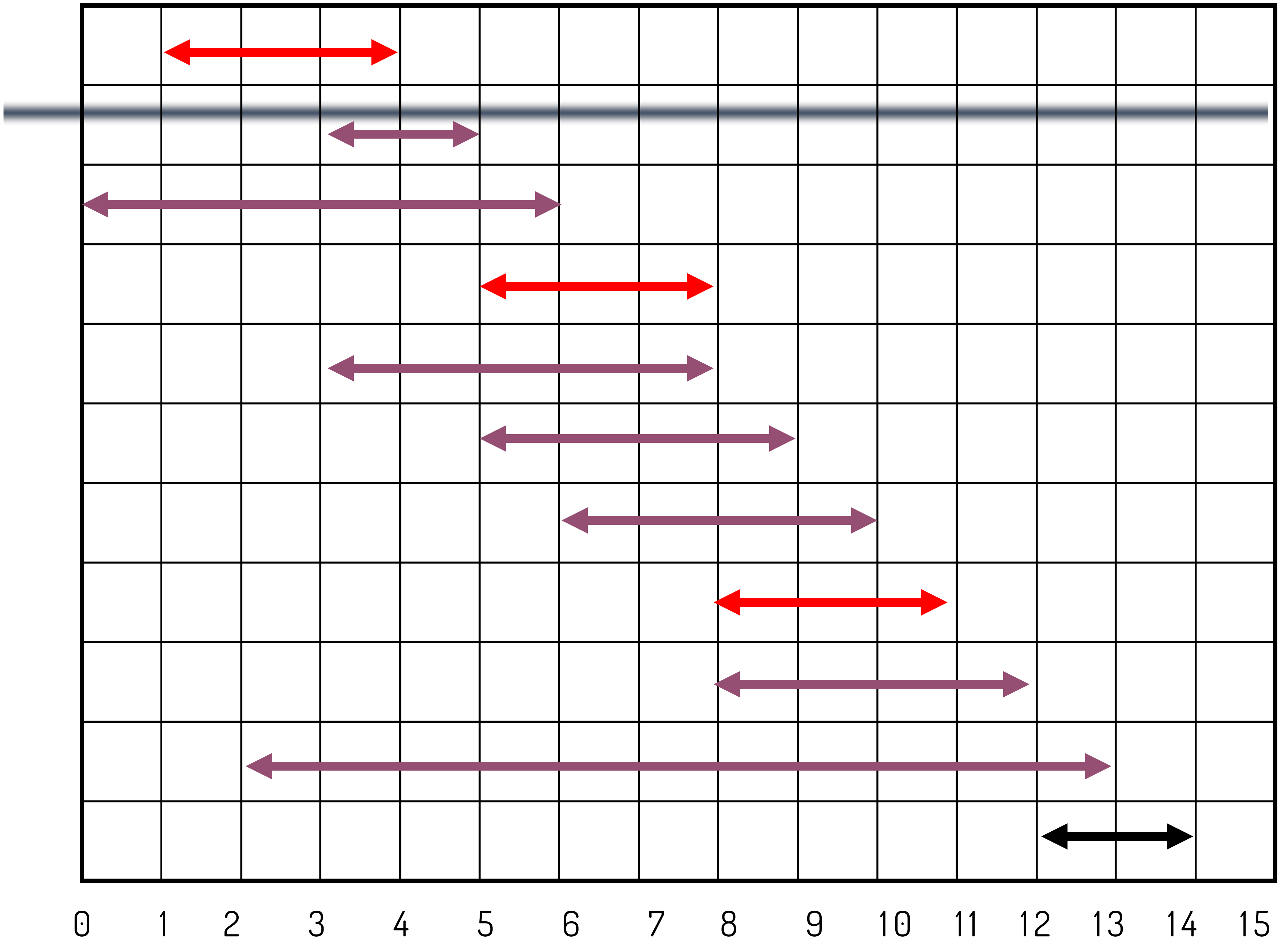

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

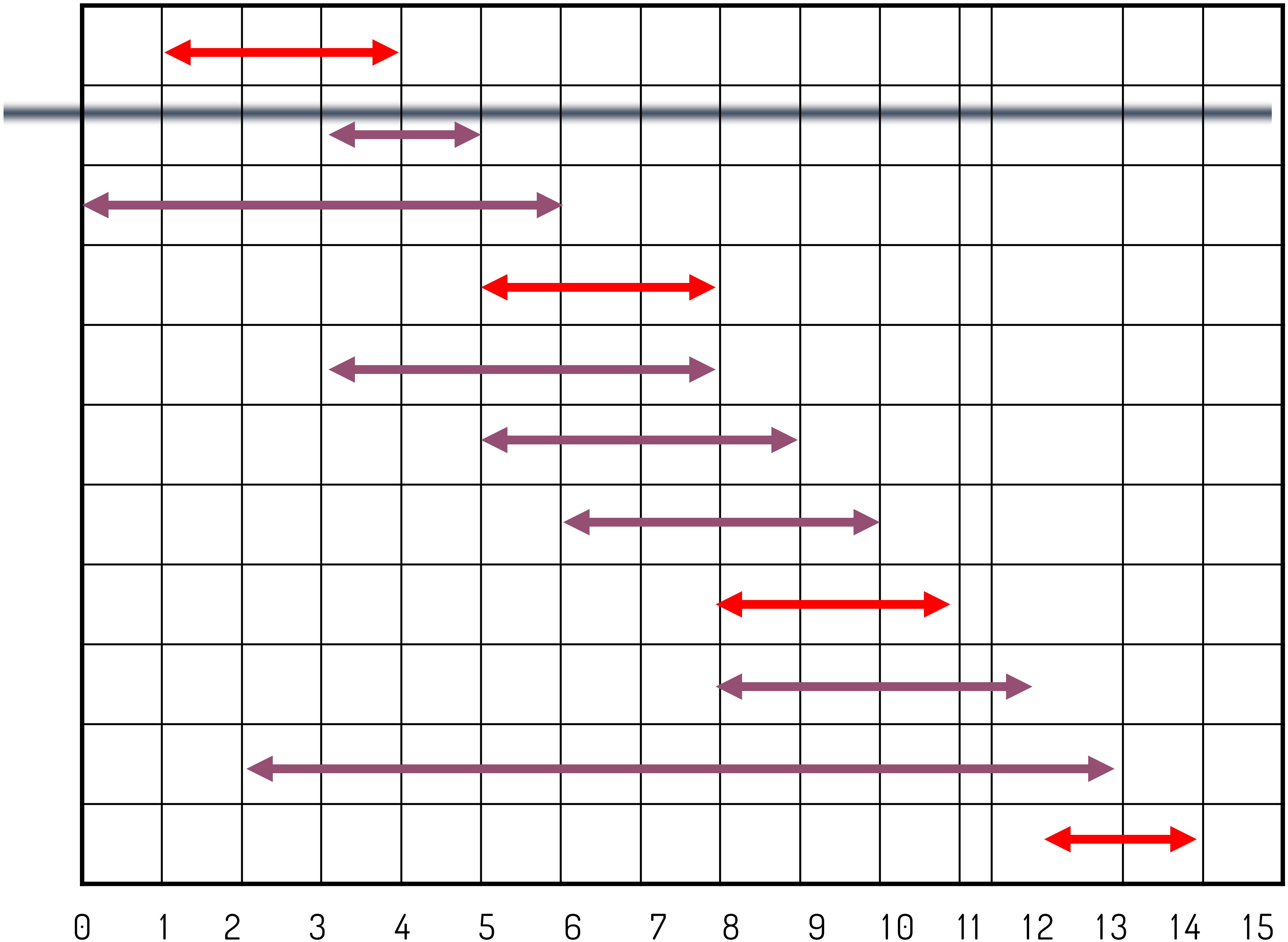

0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15


0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15


0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15


0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15


0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15


0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

Assuming activities are sorted by finish time

GREEDY-ACTIVITY-SELECTOR(s, f)

$n \leftarrow length[s]$
$A \leftarrow \{a_1\}$
$i \leftarrow 1$
for $m \leftarrow 2$ **to** n
    do if $s_m \geq f_i$
        then $A \leftarrow A \cup \{a_m\}$
            $i \leftarrow m$
return A

Why it is Greedy?

- Greedy in the sense that it leaves as much opportunity as possible for the remaining activities to be scheduled
- The greedy choice is the one that maximizes the amount of unscheduled time remaining

Huffman Codes

- Widely used technique for data compression
- Assume the data to be a sequence of characters
- Looking for an effective way of storing the data
- ***Binary character code***
    - Uniquely represents a character by a binary string
- Lossless Compression Technique.
- https://www.youtube.com/watch?v=co4_ahEDCho

Fixed-Length Codes

E.g.: Lets say, Data file containing 100,000 characters

	a	b	c	d	e	f
Frequency (thousands)	45	13	12	16	9	5

- 3 bits needed
- a = 000, b = 001, c = 010, d = 011, e = 100, f = 101
- Requires: $100{,}000 \cdot 3 = 300{,}000$ bits

Huffman Codes

- Idea:
  - Use the frequencies of occurrence of characters to build a optimal way of representing each character

	a	b	c	d	e	f
Frequency (thousands)	45	13	12	16	9	5

Variable-Length Codes

E.g.: Data file containing 100,000 characters

	a	b	c	d	e	f
Frequency (thousands)	45	13	12	16	9	5

- Assign short codewords to frequent characters and long codewords to infrequent characters
- a = 0, b = 101, c = 100, d = 111, e = 1101, f = 1100
- $(45 \cdot 1 + 13 \cdot 3 + 12 \cdot 3 + 16 \cdot 3 + 9 \cdot 4 + 5 \cdot 4) \cdot 1{,}000$

  = 224,000 bits

Encoding with Binary Character Codes

- Encoding
    - Concatenate the codewords representing each character in the file
- *E.g.:*
    - a = 0, b = 101, c = 100, d = 111, e = 1101, f = 1100
    - abc = 0 · 101 · 100 = 0101100

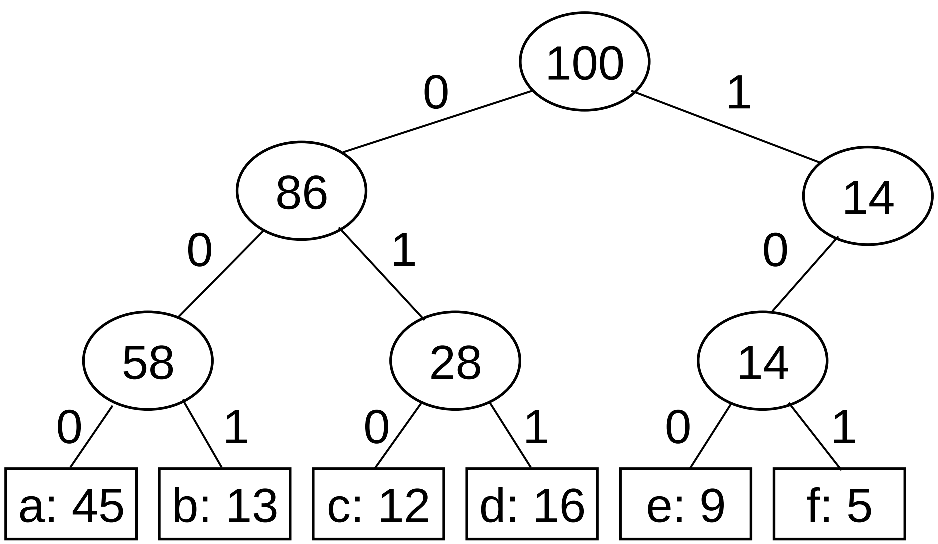
Decoding with Binary Character Codes

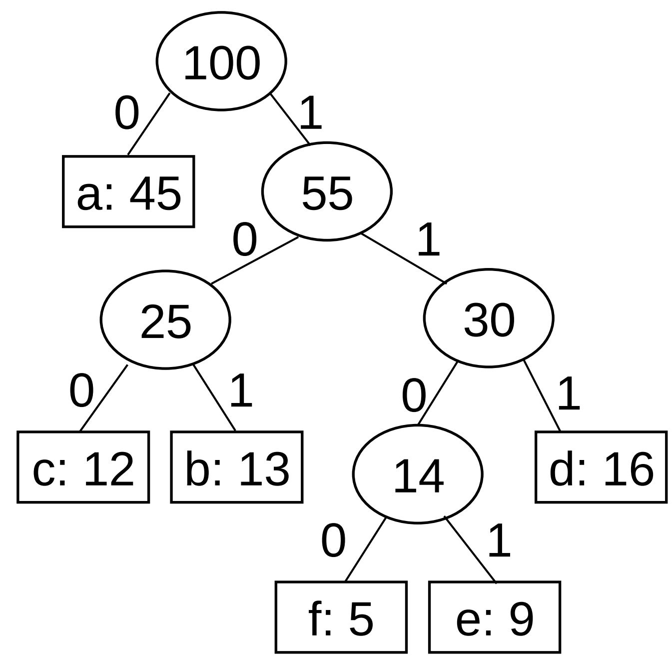
- Prefix codes simplify decoding
  - No codeword is a prefix of another $\Rightarrow$ the codeword that begins an encoded file is unambiguous
- Approach
  - Identify the initial codeword
  - Translate it back to the original character
  - Repeat the process on the remainder of the file
- *E.g.:*
  - a = 0, b = 101, c = 100, d = 111, e = 1101, f = 1100
  - 001011101 =

$$0 \cdot 0 \cdot 101 \cdot 1101 = \text{aabe}$$

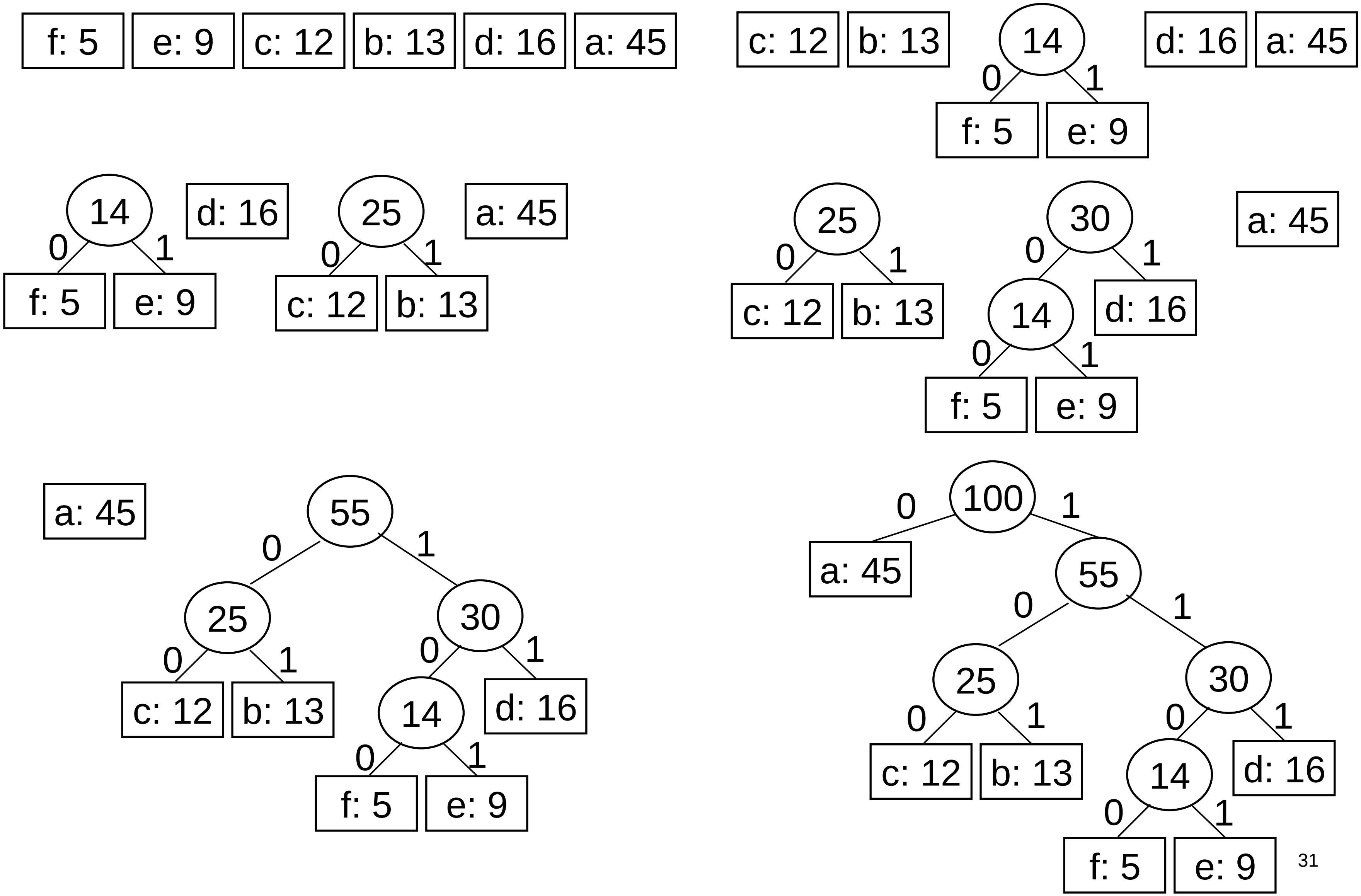
Prefix Code Representation

- Binary tree whose leaves are the given characters
- Binary codeword
  - the path from the root to the character, where 0 means "go to the left child" and 1 means "go to the right child"
- Length of the codeword
  - Length of the path from root to the character leaf (depth of node)





Example



f: 5
e: 9
c: 12
b: 13
d: 16
a: 45
c: 12
b: 13
14
0
1
f: 5
e: 9
d: 16
a: 45
14
0
1
f: 5
e: 9
d: 16
25
0
1
c: 12
b: 13
a: 45
25
0
1
c: 12
b: 13
30
0
1
14
0
1
f: 5
e: 9
d: 16
a: 45
a: 45
55
0
1
25
0
1
c: 12
b: 13
30
0
1
14
0
1
f: 5
e: 9
d: 16
100
0
1
a: 45
55
0
1
25
0
1
c: 12
b: 13
30
0
1
14
0
1
f: 5
e: 9
d: 16