Improving the Energetic Reasoning: How I followed 15-year-old advice from my supervisor

Claude-Guy Quimper
Université Laval, Canada

Alejandro López-Ortiz

1967 - 2017

Purposes of this talk

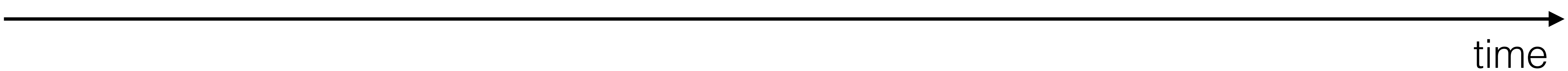
- To reveal some of my supervisor’s greatest advice.
- To show how I still apply his advice when working my students.
- To present a $O(n \log^2 n)$ checker for the energetic reasoning

Yanick Ouellet

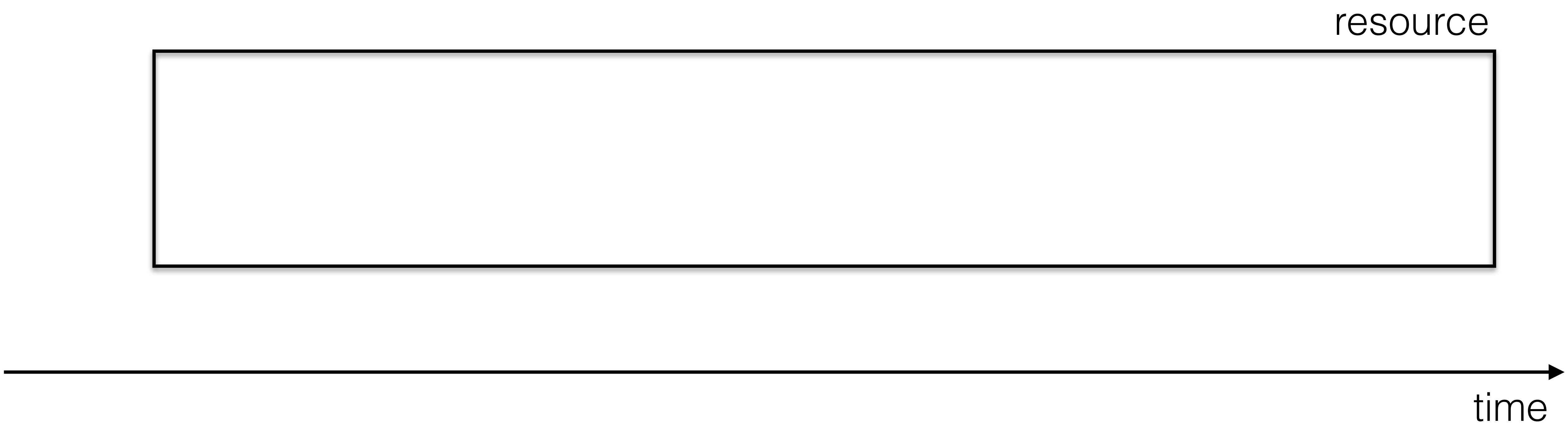
Outline

- The CUMULATIVE constraint
- The energetic check
- Our new checker
  - The computation of energy (Advice #1)
  - Monge matrices (Advice #2)
  - Experiments (Advice #3)
- A last advice (Advice #4)
- Conclusion

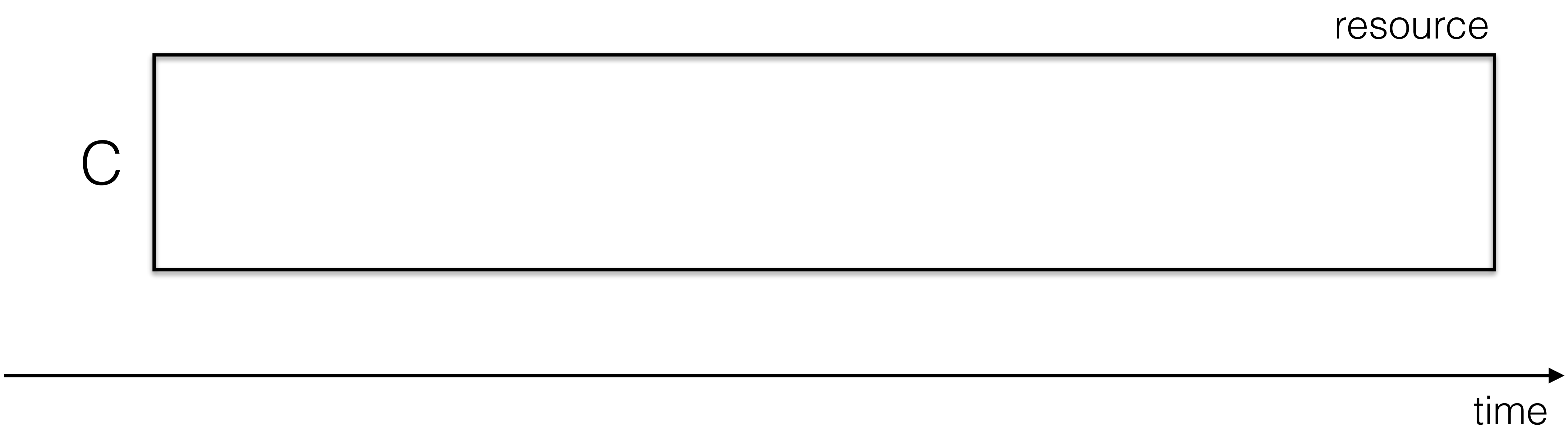
Definitions



Definitions

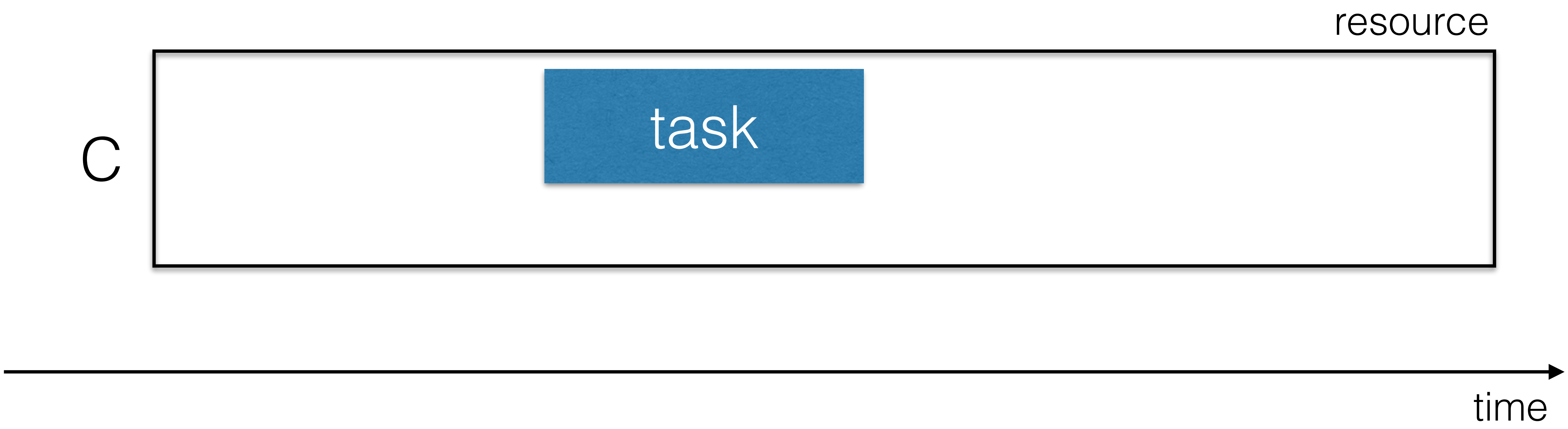


Definitions



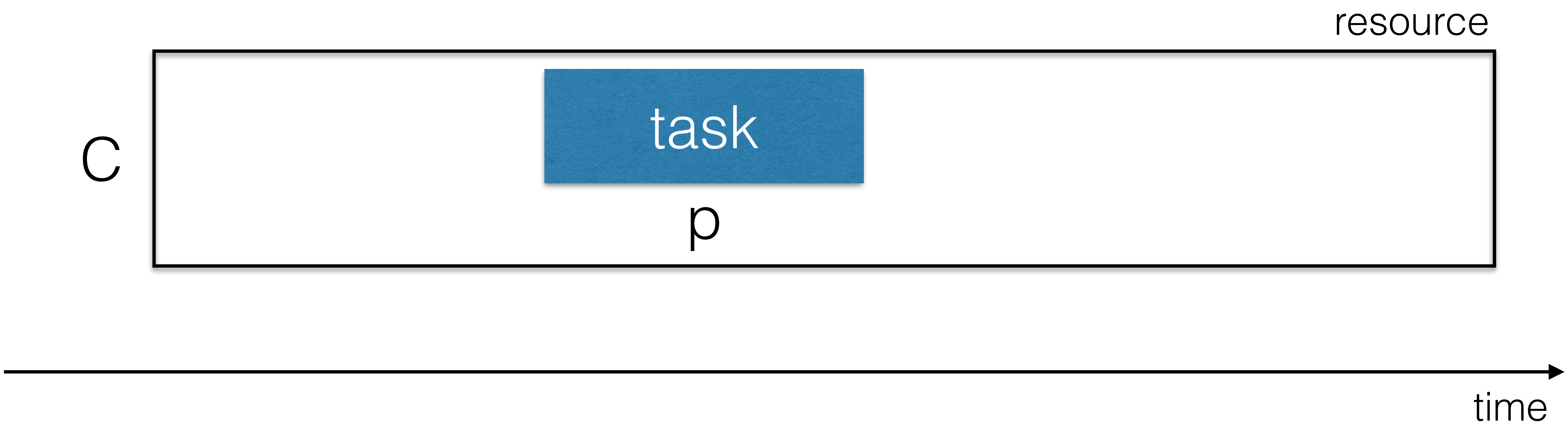
- C: capacity of the resource

Definitions



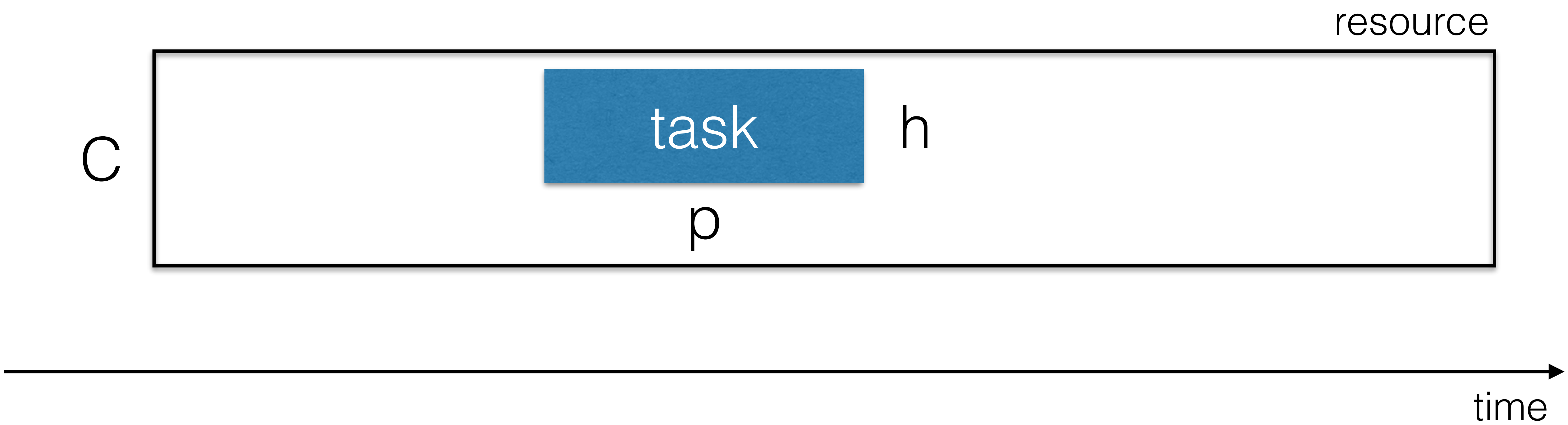
- C: capacity of the resource

Definitions



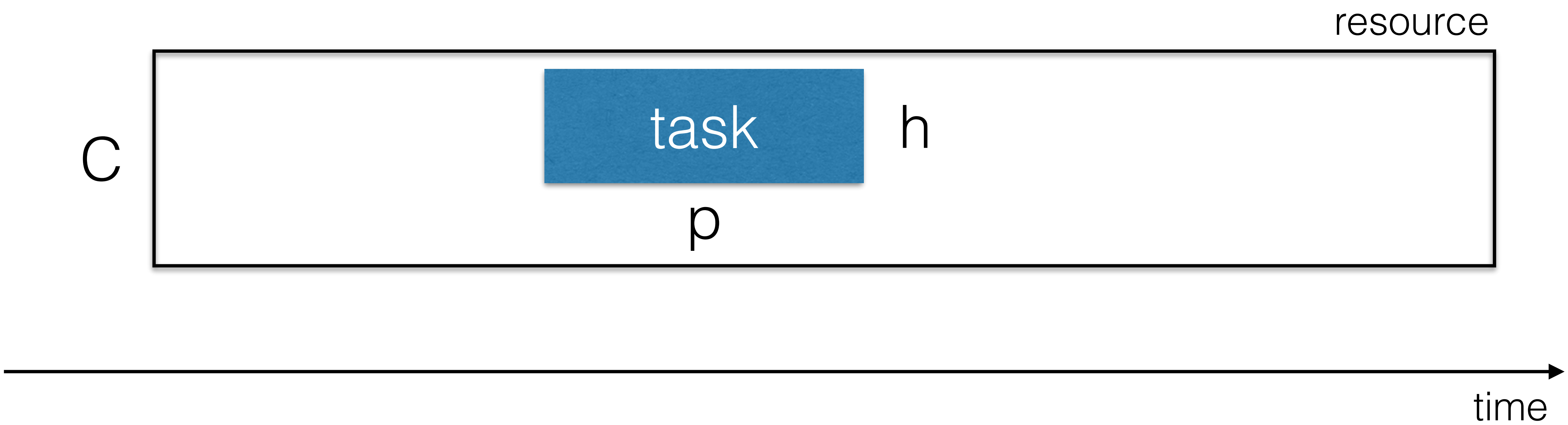
- C: capacity of the resource
- p: processing time

Definitions



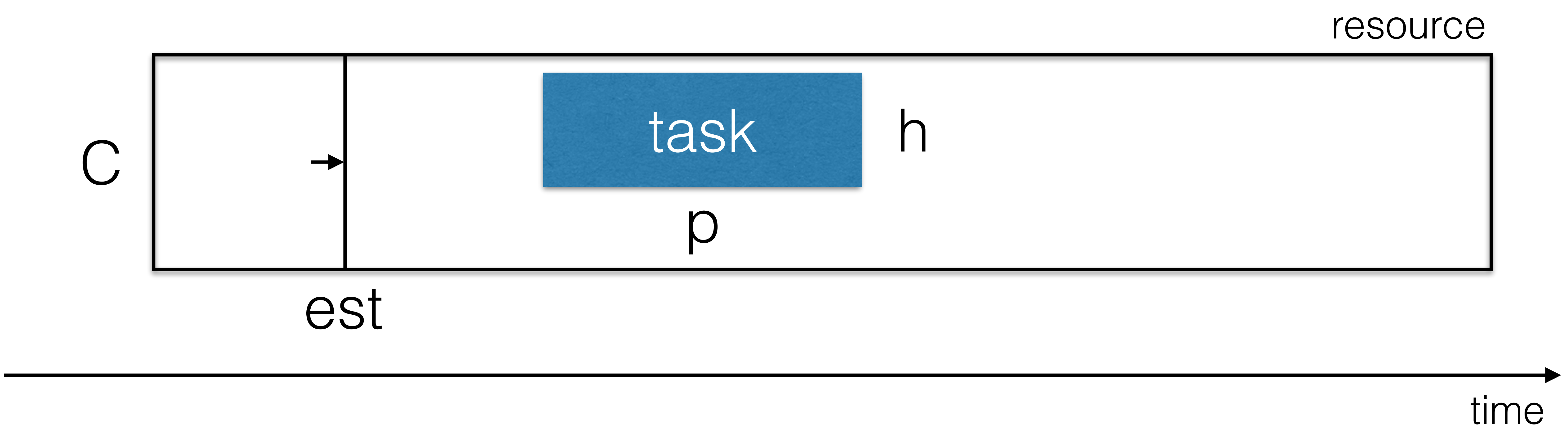
- C: capacity of the resource
- p: processing time
- h: height

Definitions



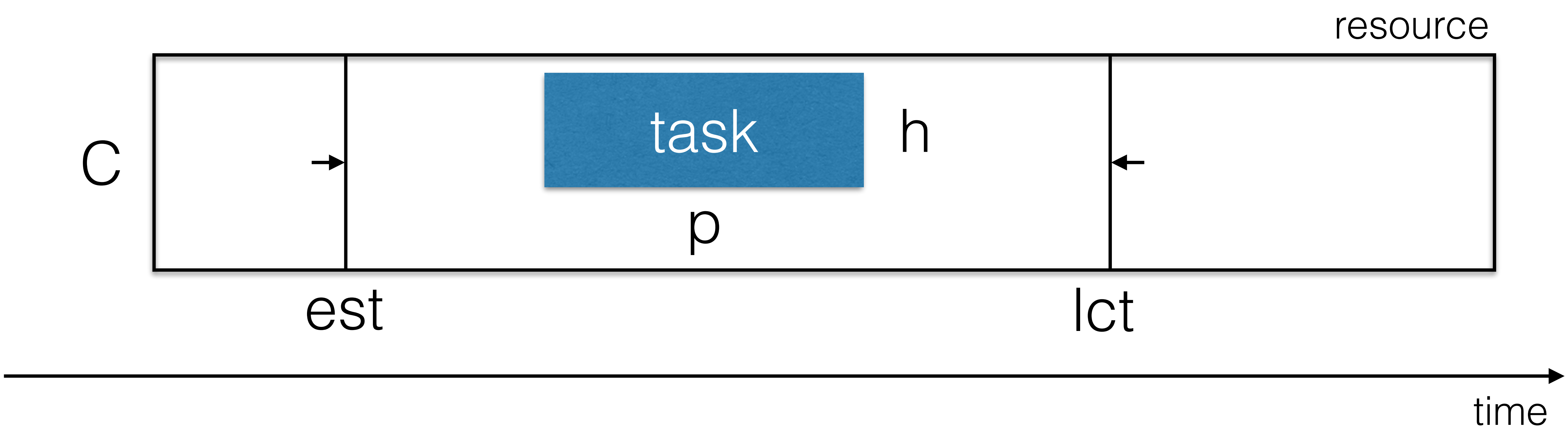
- C: capacity of the resource
- p: processing time
- h: height
- e: energy ($e = p \times h$)

Definitions



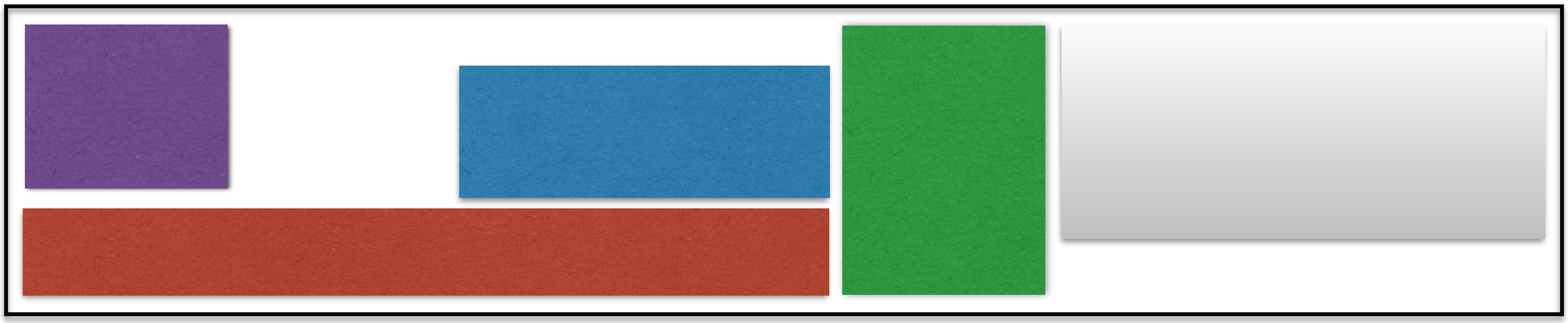
- C: capacity of the resource
- p: processing time
- h: height
- e: energy ($e = p \times h$)
- est: earliest starting time

Definitions



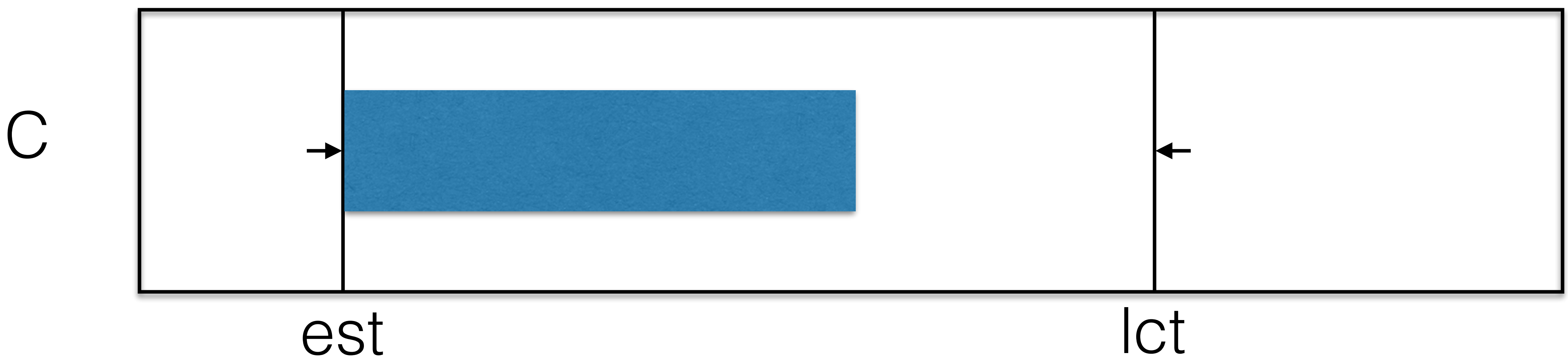
- C: capacity of the resource
- p: processing time
- h: height
- e: energy ($e = p \times h$)
- est: earliest starting time
- lct: latest completion time

The CUMULATIVE constraint

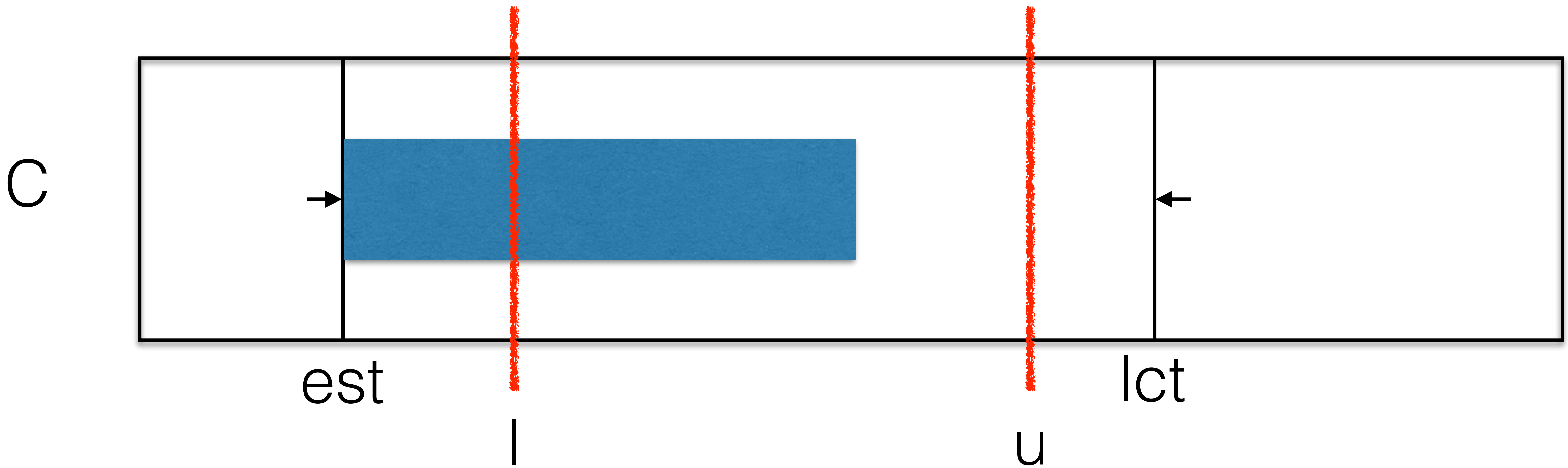


- Tasks must be scheduled between their est and lct.
- No overlap.
- The capacity of the resource is not exceeded.

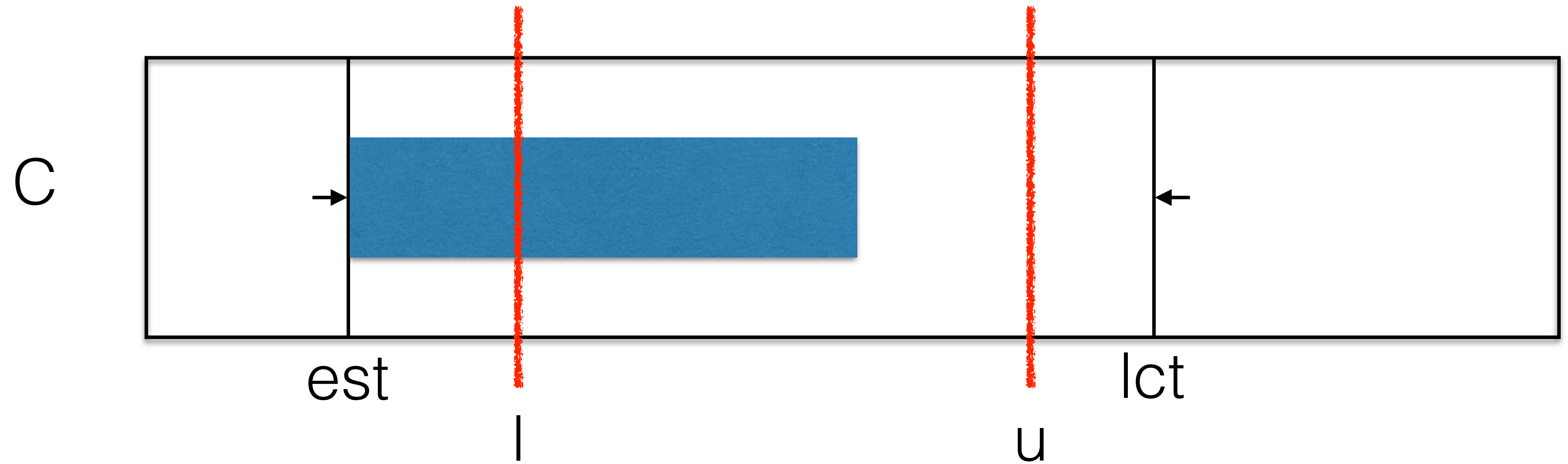
The energetic check



The energetic check

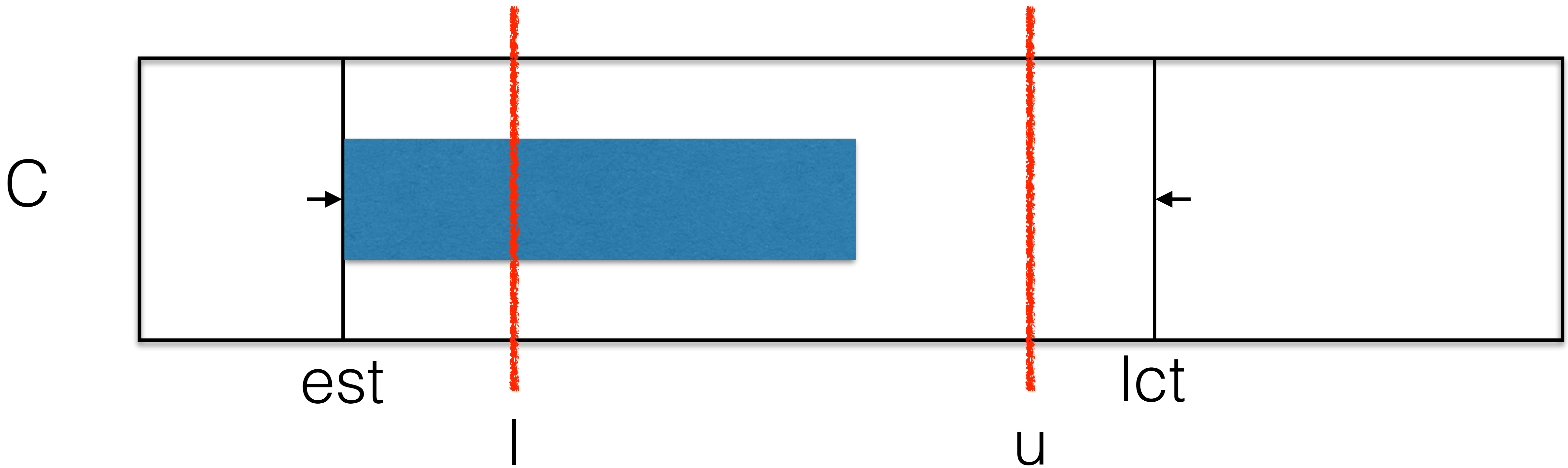


The energetic check



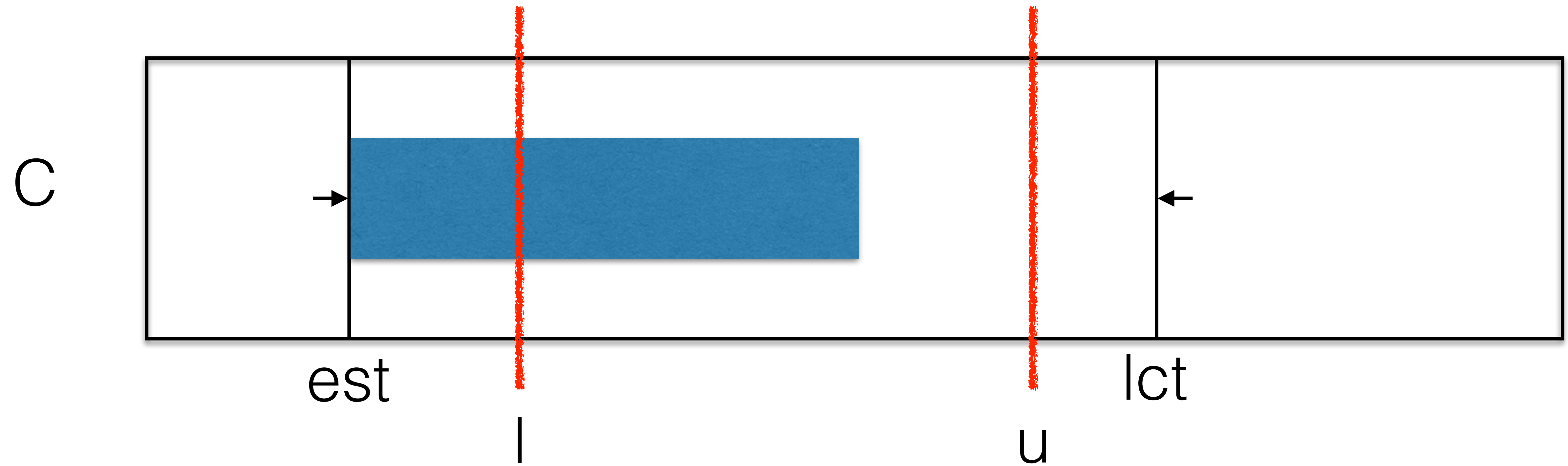
$$E(i, l, u) = h_i \cdot \max(0, \min(\qquad\qquad))$$

The energetic check



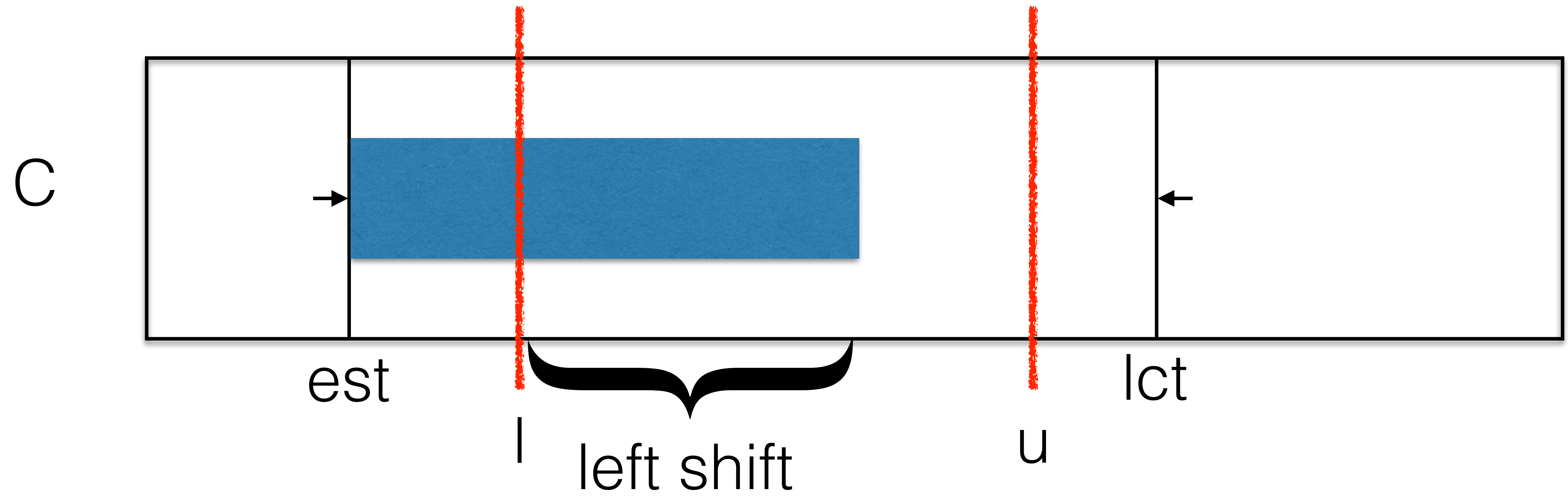
$$E(i,l,u) = h_i \cdot \max(0, \min(u-l, \qquad\qquad\qquad\qquad))$$

The energetic check



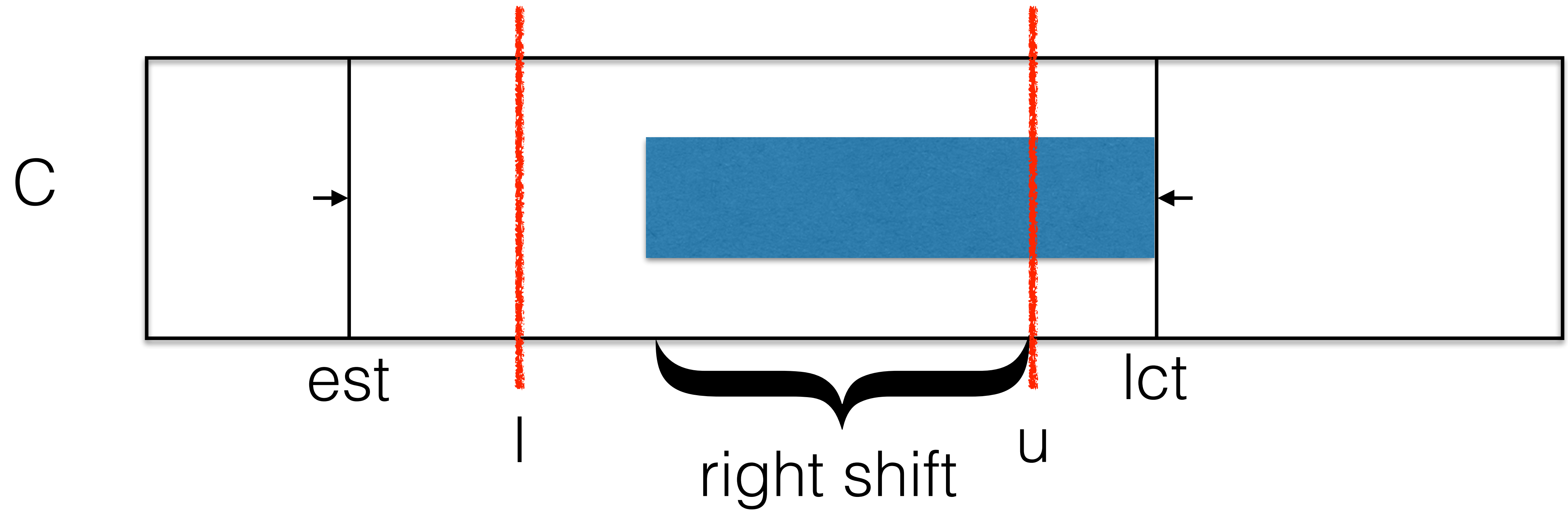
$$E(i,l,u) = h_i \cdot \max(0, \min(u-l, p_i, \quad))$$

The energetic check



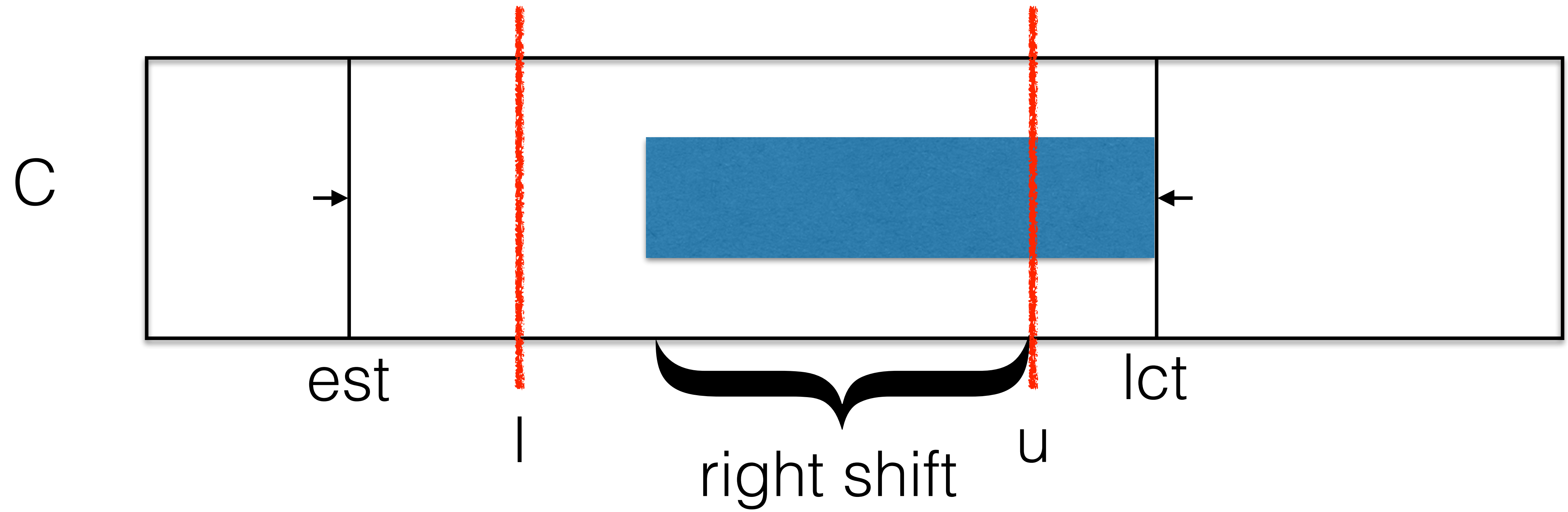
$$E(i, l, u) = h_i \cdot \max(0, \min(u - l, p_i, \text{est}_i + p_i - l \qquad))$$

The energetic check



$$E(i,l,u) = h_i \cdot \max(0, \min(u - l, p_i, \mathrm{est}_i + p_i - l, u - (\mathrm{lct}_i - p_i)))$$

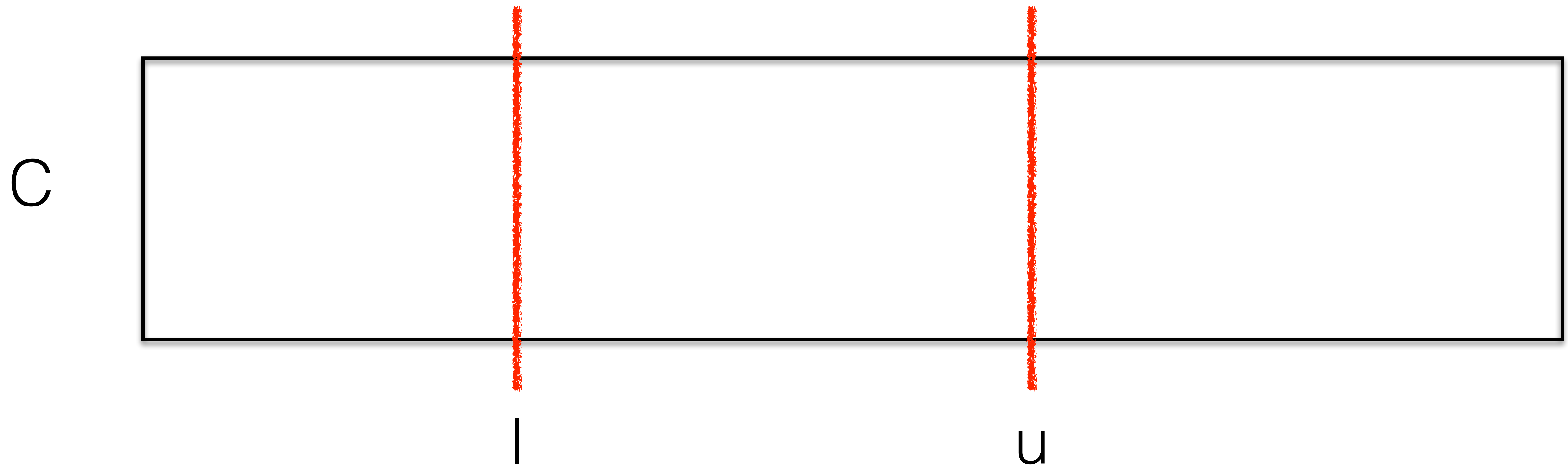
The energetic check



$$E(i,l,u) = h_i \cdot \max(0, \min(u-l, p_i, \mathrm{est}_i + p_i - l, u - (\mathrm{lct}_i - p_i)))$$

$$S(l,u) = C \cdot (u-l) - \sum_i E(i,l,u)$$

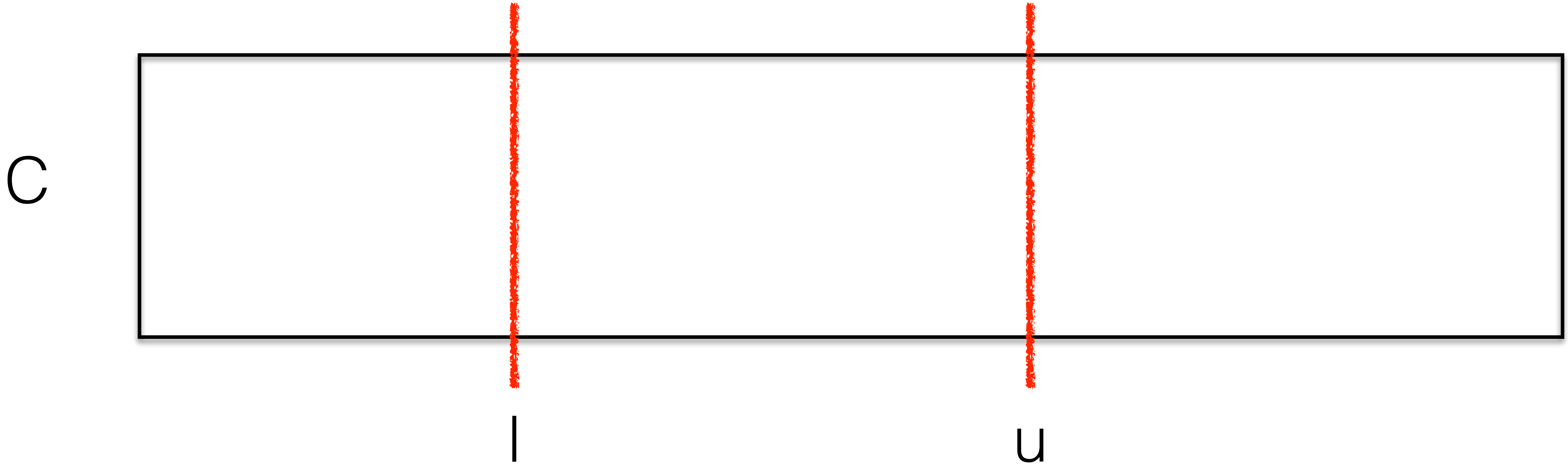
The energetic check



$$E(i,l,u) = h_i \cdot \max(0, \min(u - l, p_i, \mathrm{est}_i + p_i - l, u - (\mathrm{lct}_i - p_i)))$$

$$S(l,u) = C \cdot (u - l) - \sum_i E(i,l,u)$$

The energetic check



$$E(i,l,u) = h_i \cdot \max(0, \min(u-l, p_i, \mathrm{est}_i + p_i - l, u - (\mathrm{lct}_i - p_i)))$$

$$S(l,u) = C \cdot (u-l) - \sum_i E(i,l,u) \geq 0$$

Existing checkers

- Baptiste, Le Pape, and Nuijten showed that it is sufficient to test $O(n^2)$ intervals.

Existing checkers

- Baptiste, Le Pape, and Nuijten showed that it is sufficient to test $O(n^2)$ intervals.
- The slack $S(l, u)$ is computed in constant time, using the previous computation of $S(l, u-1)$.

Existing checkers

- Baptiste, Le Pape, and Nuijten showed that it is sufficient to test $O(n^2)$ intervals.
- The slack $S(l, u)$ is computed in constant time, using the previous computation of $S(l, u-1)$.
- Running time complexity: $O(n^2)$

Existing checkers

- Baptiste, Le Pape, and Nuijten showed that it is sufficient to test $O(n^2)$ intervals.
- The slack $S(l, u)$ is computed in constant time, using the previous computation of $S(l, u-1)$.
- Running time complexity: $O(n^2)$
- Derrien and Petit reduced the multiplicative constant by a factor of 7.

Goal

- To perform the energetic check in sub-quadratic time.
  - We need to test fewer than $O(n^2)$ intervals
  - We will need to compute the slack, upon request, for any time interval [l, u)

Goal

- To perform the energetic check in sub-quadratic time.
    - We need to test fewer than $O(n^2)$ intervals
    - We will need to compute the slack, upon request, for any time interval $[l, u)$

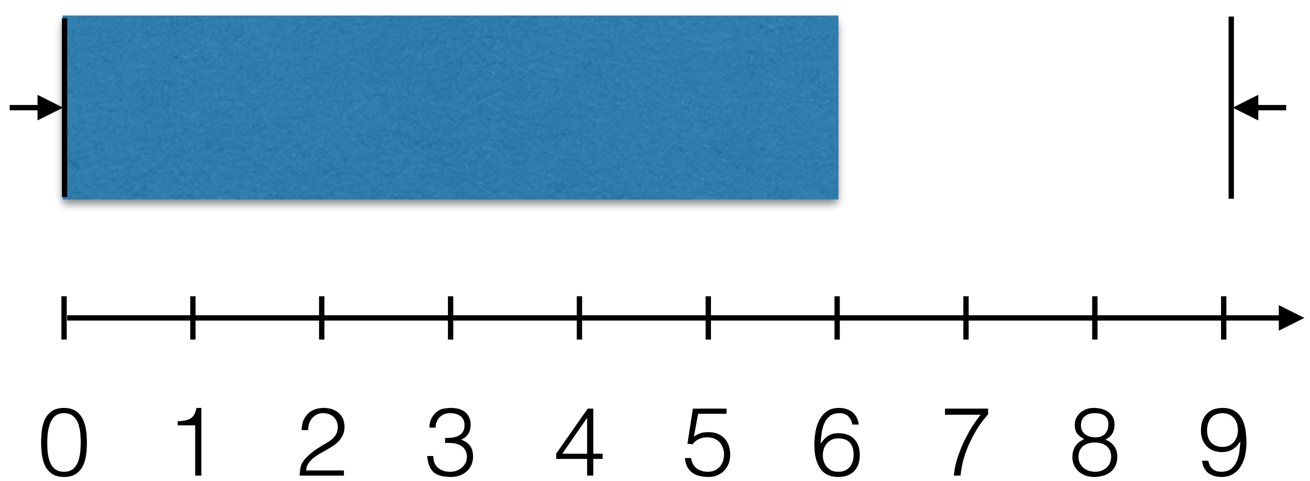
Let's start by solving this problem

Advice #1

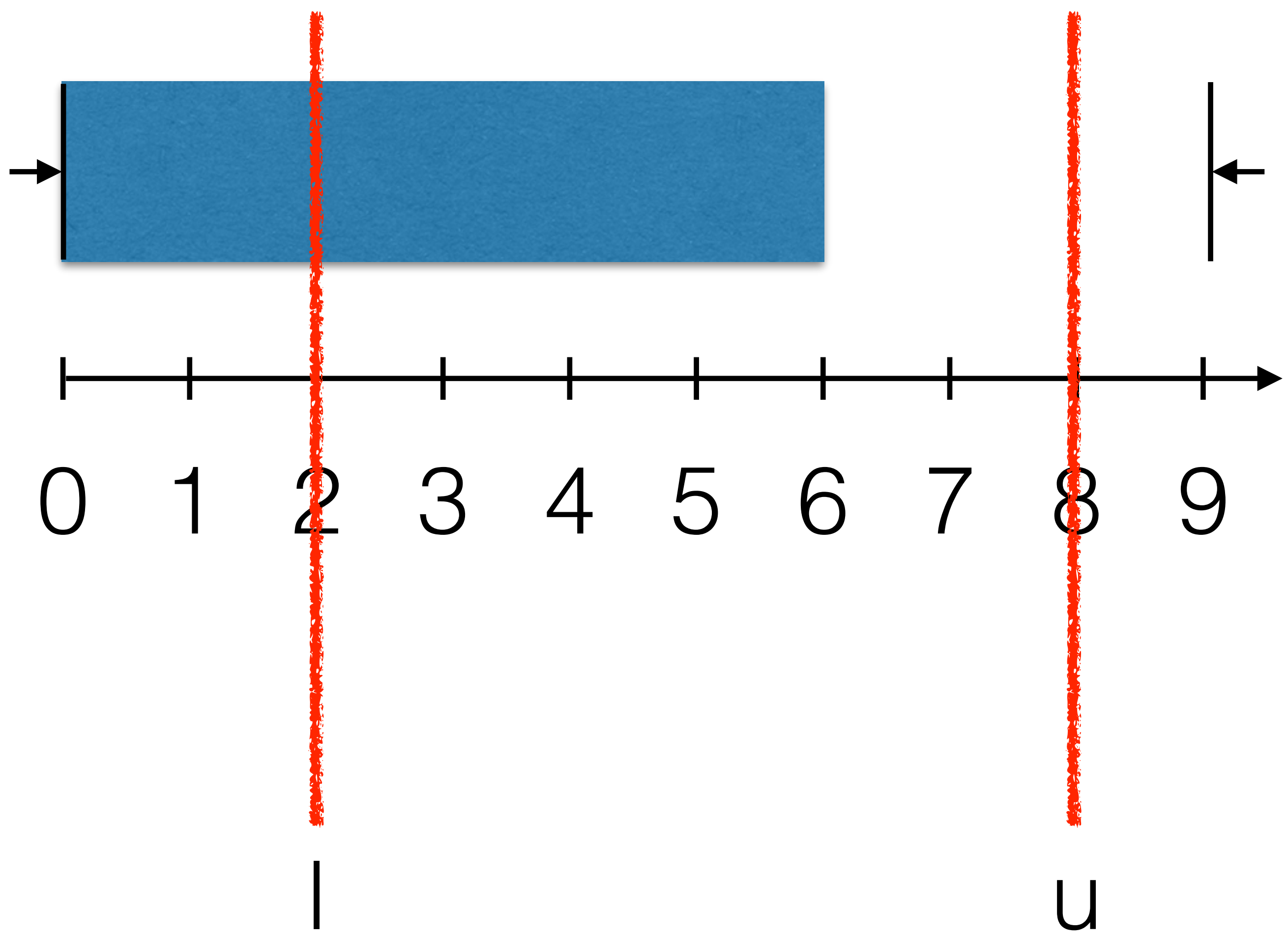
Reduce your problem to one that is already solved.

- Reductions can provide solutions out of the box.
- If not, they give a direction how to adapt a solution to your problem.
- Take time to reformulate your problem using different abstractions: graphs, points/vectors, …

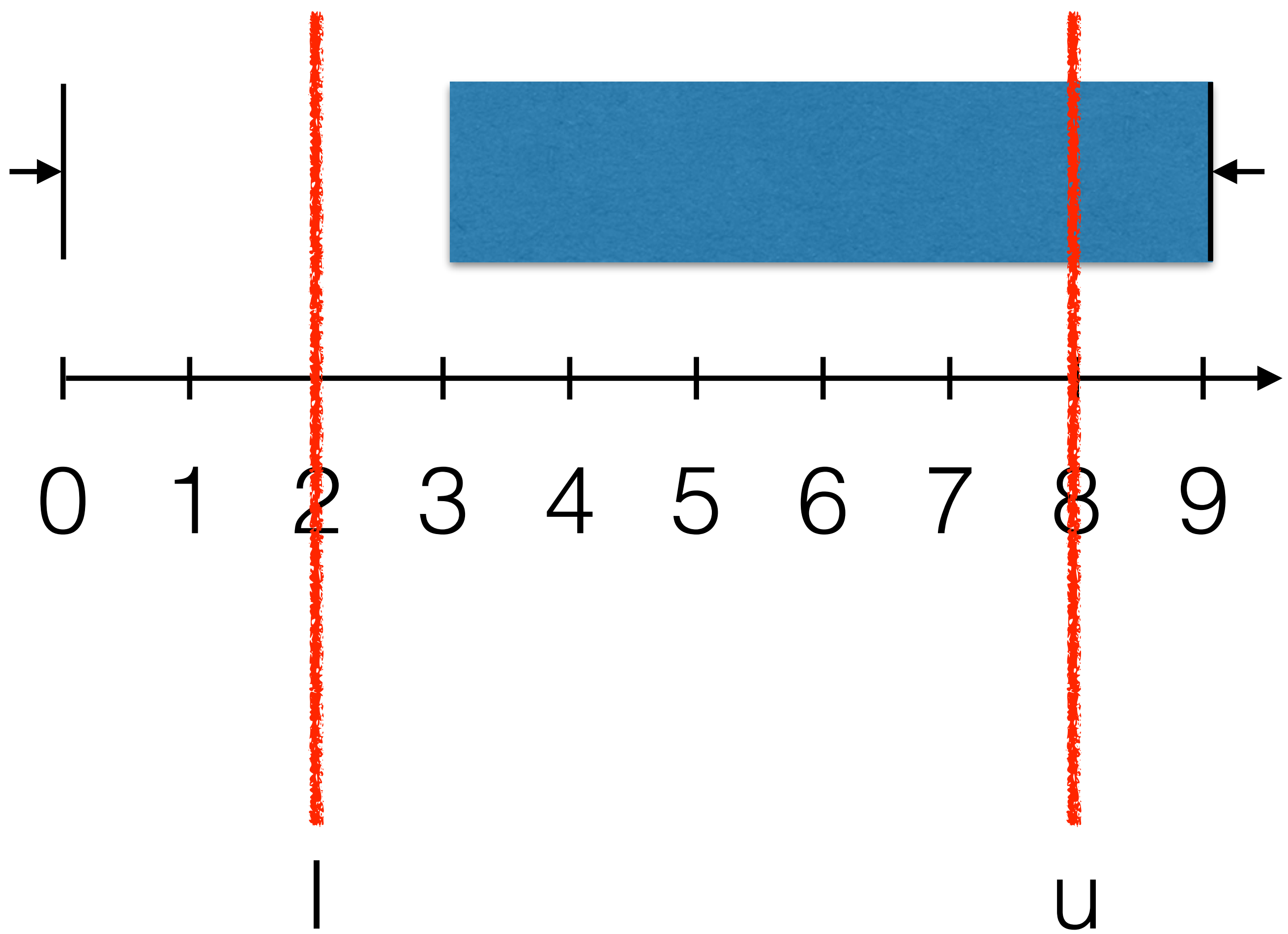
Reduction



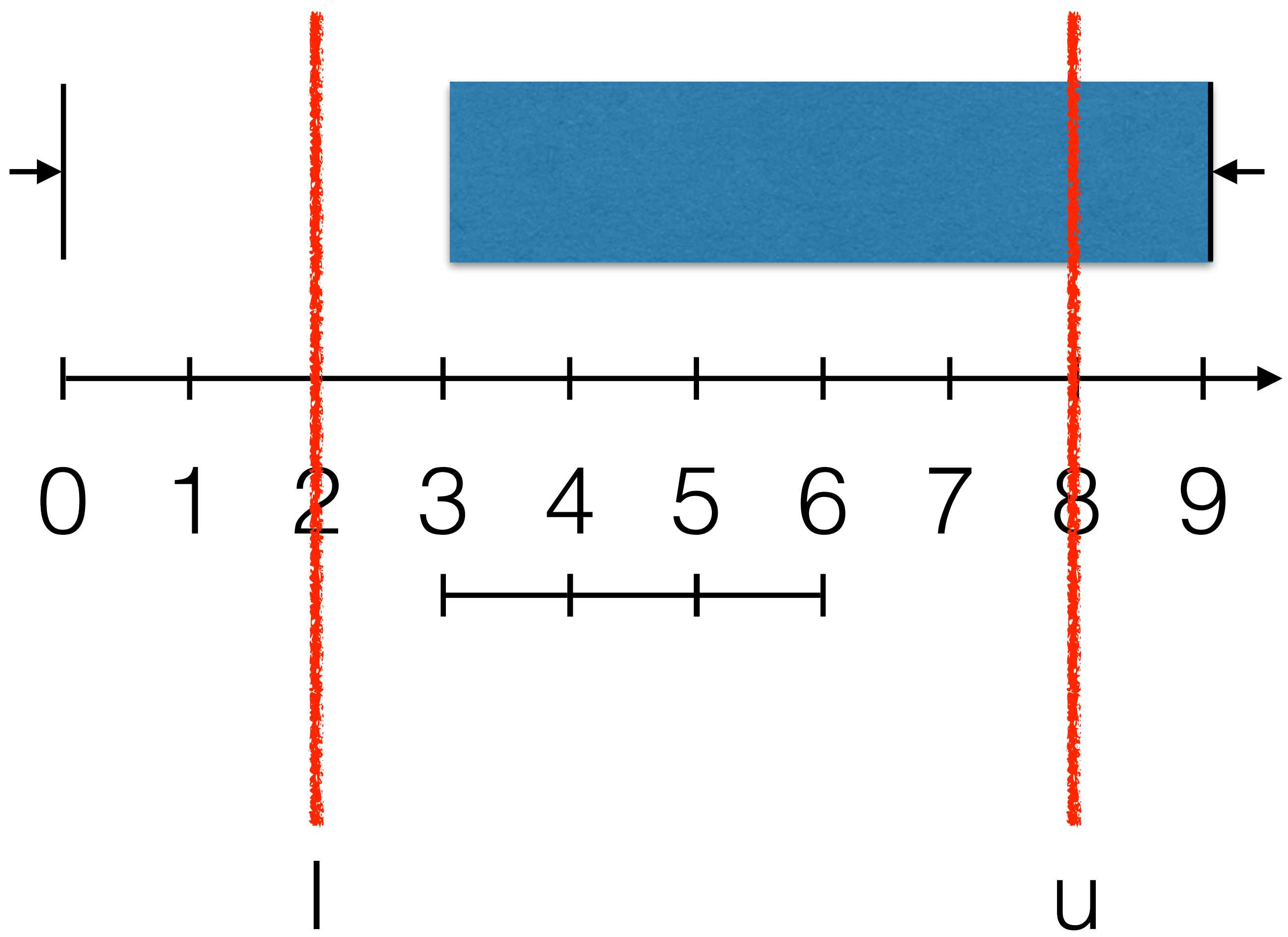
Reduction



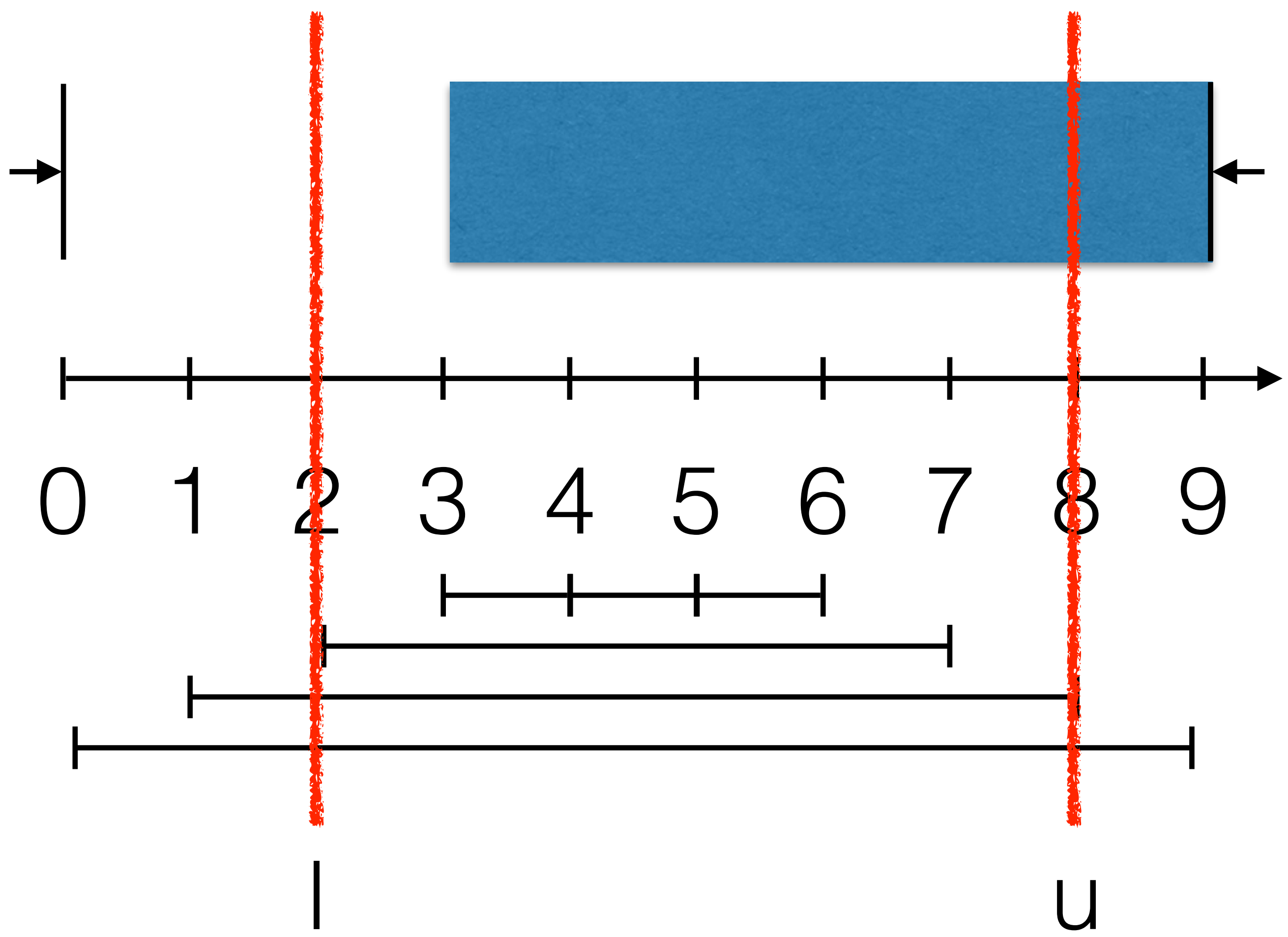
Reduction



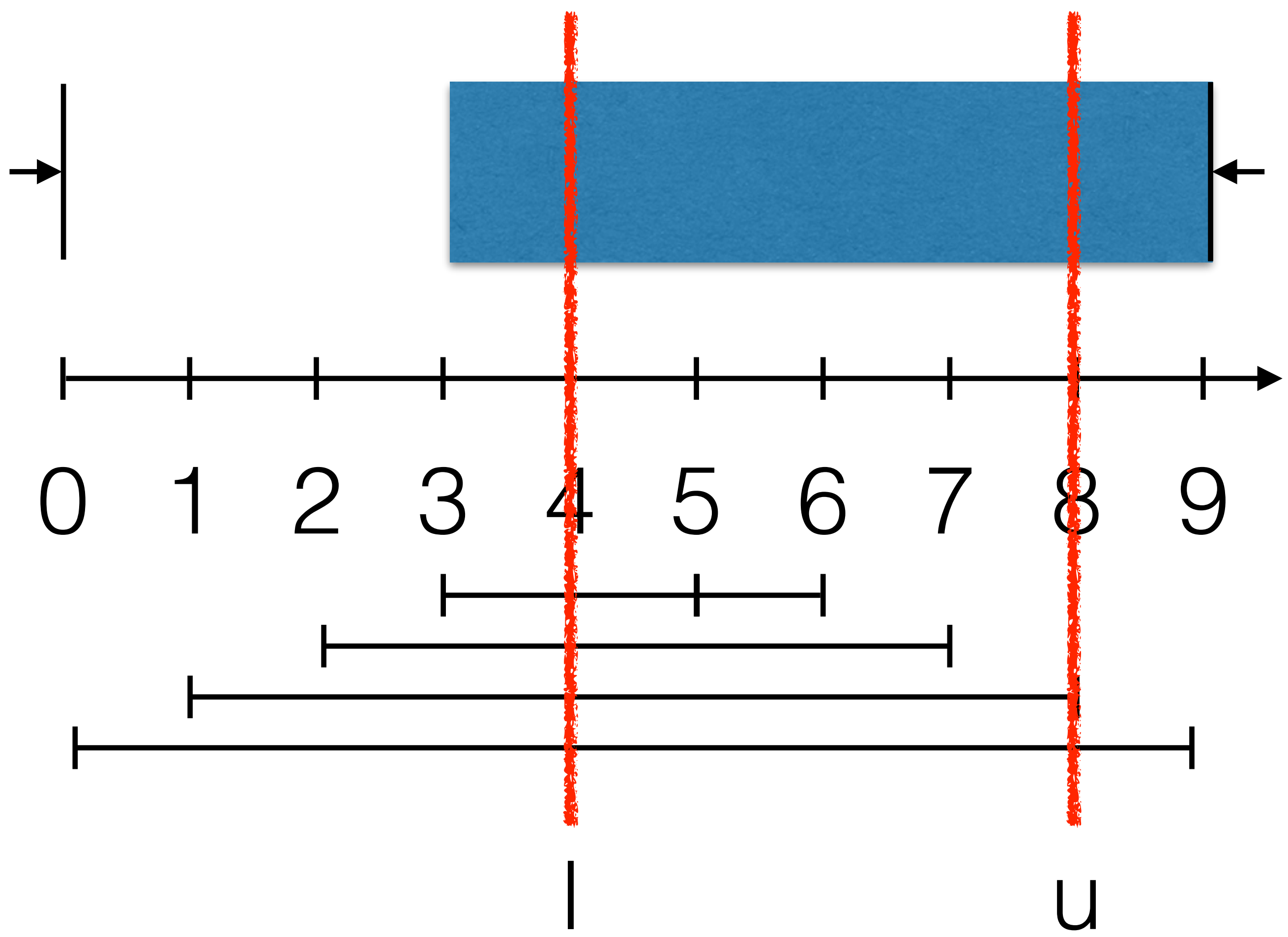
Reduction



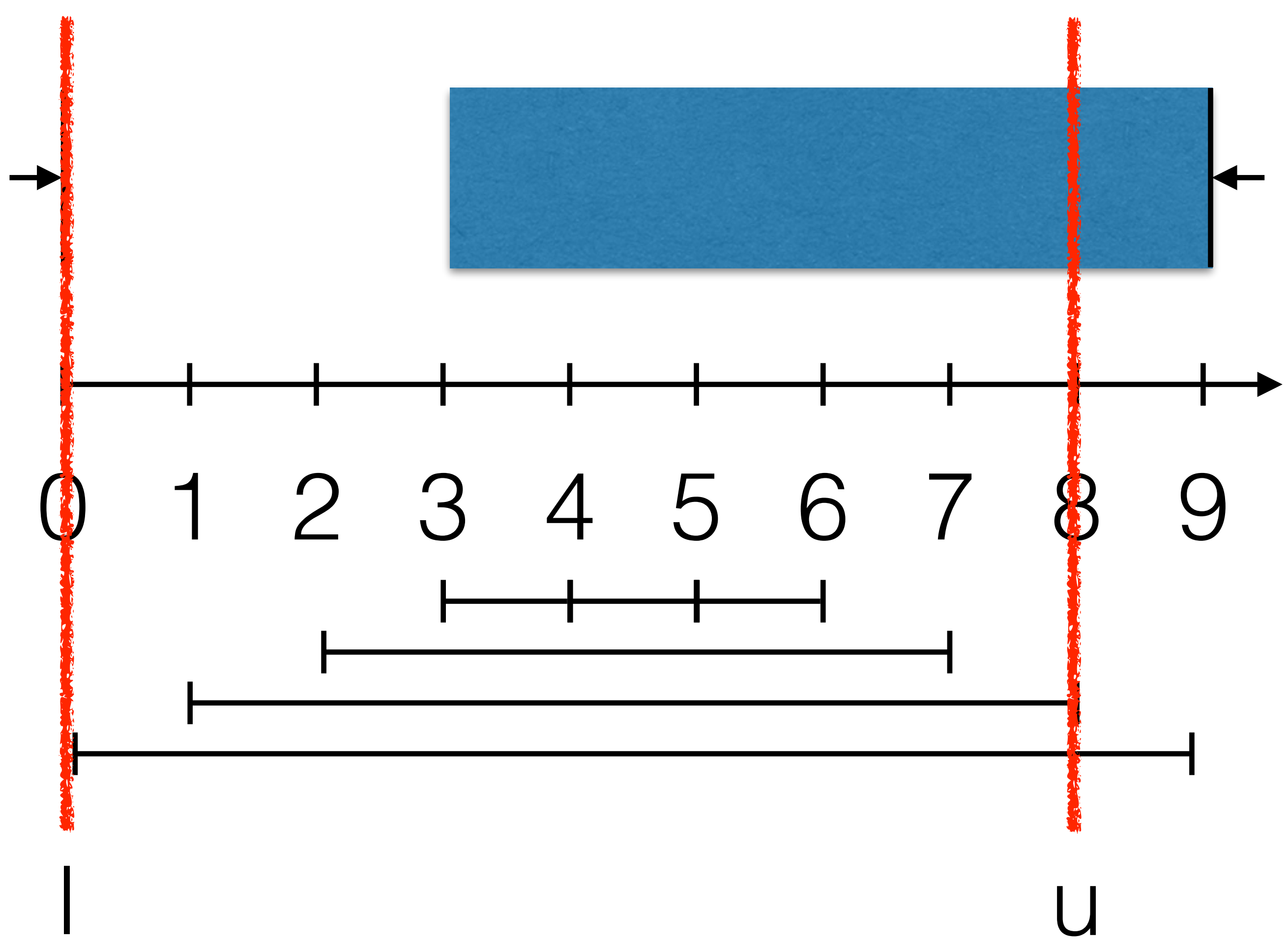
Reduction



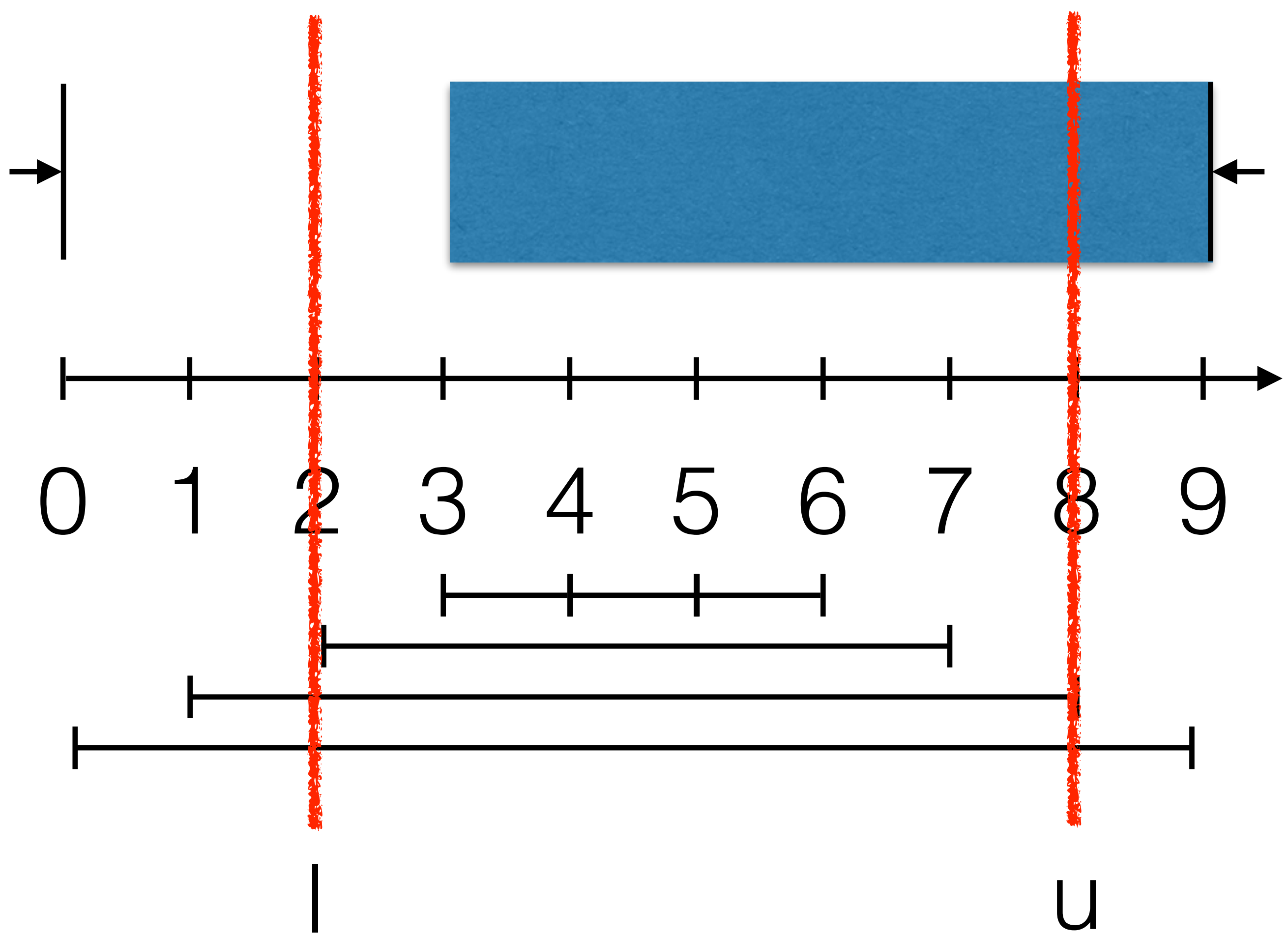
Reduction



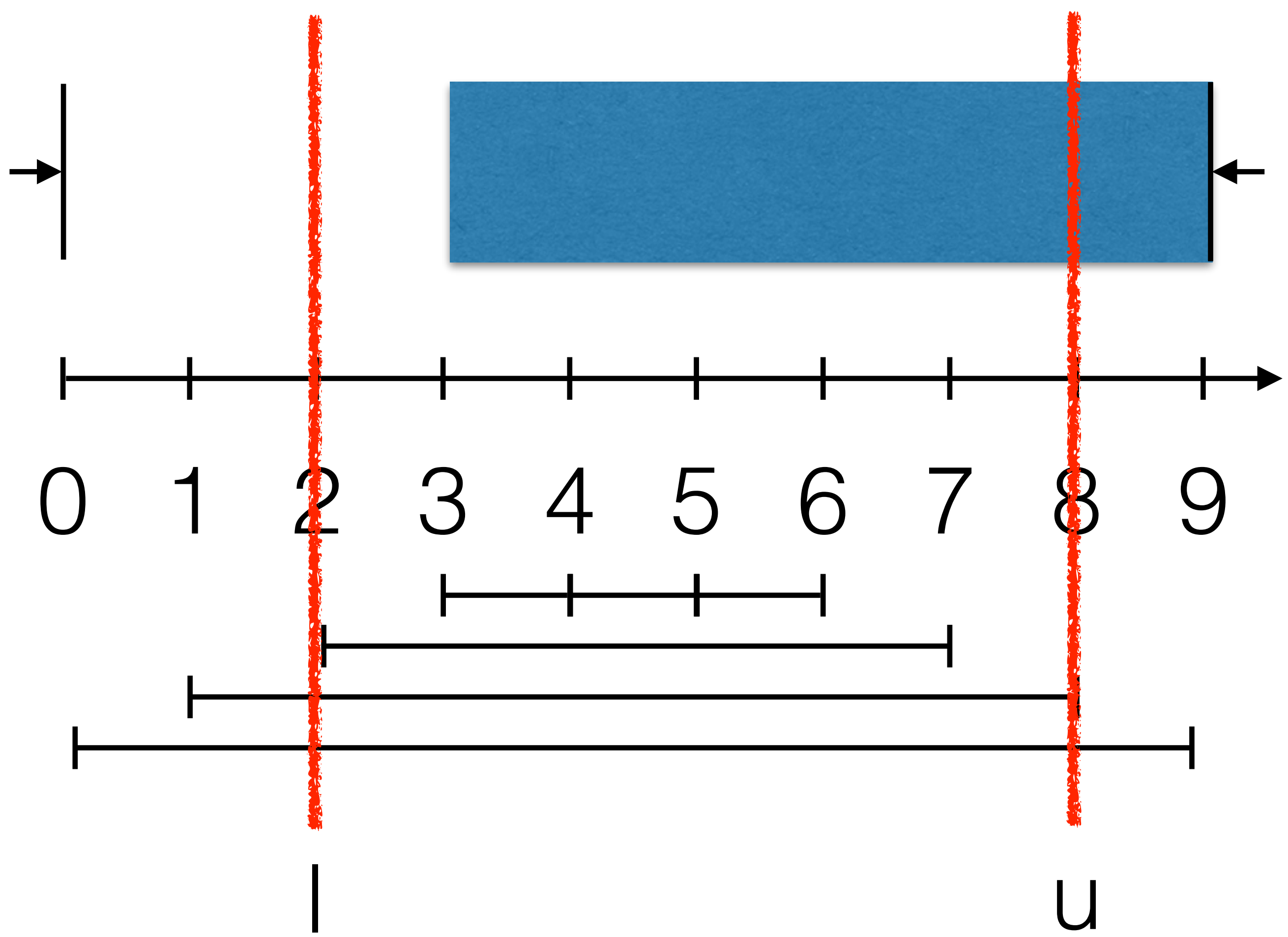
Reduction

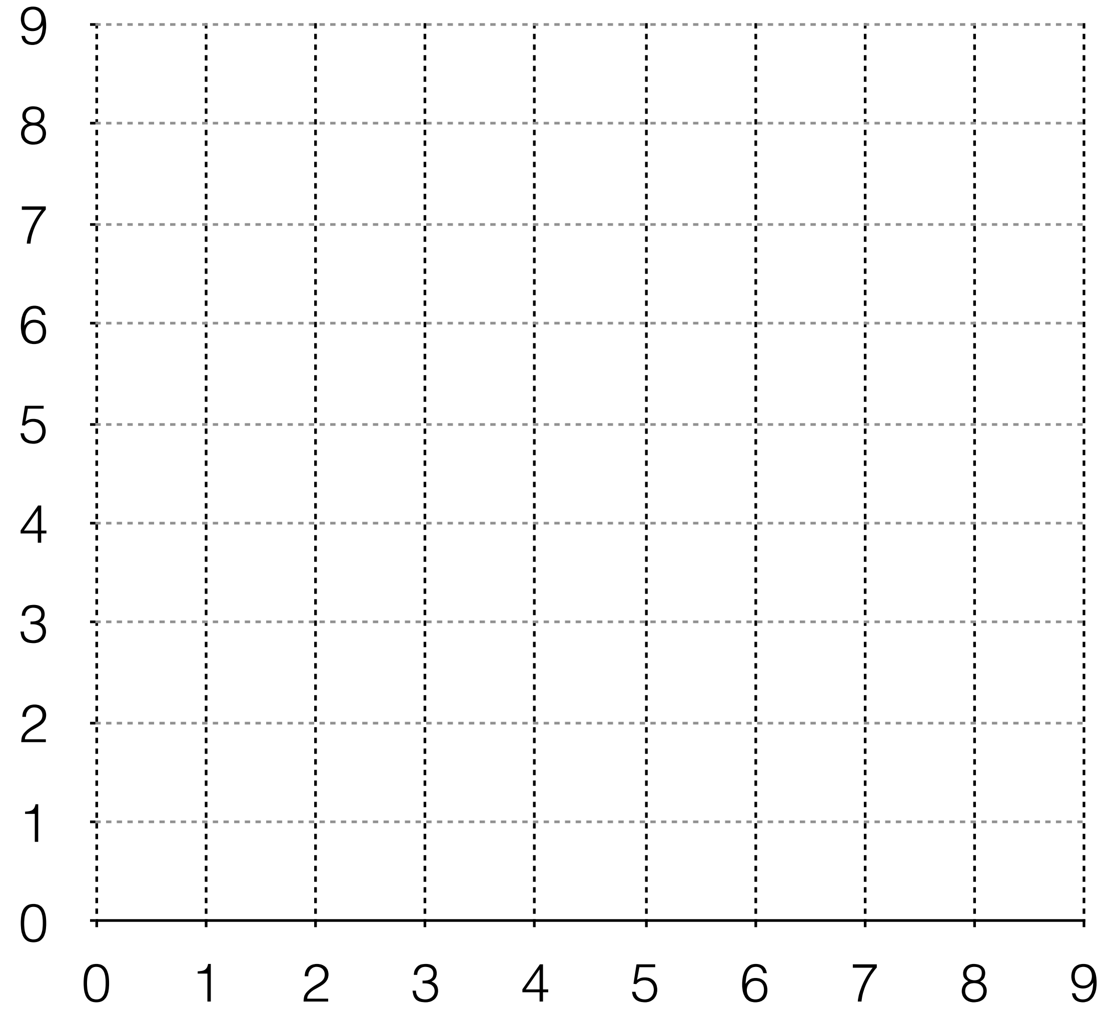


Reduction

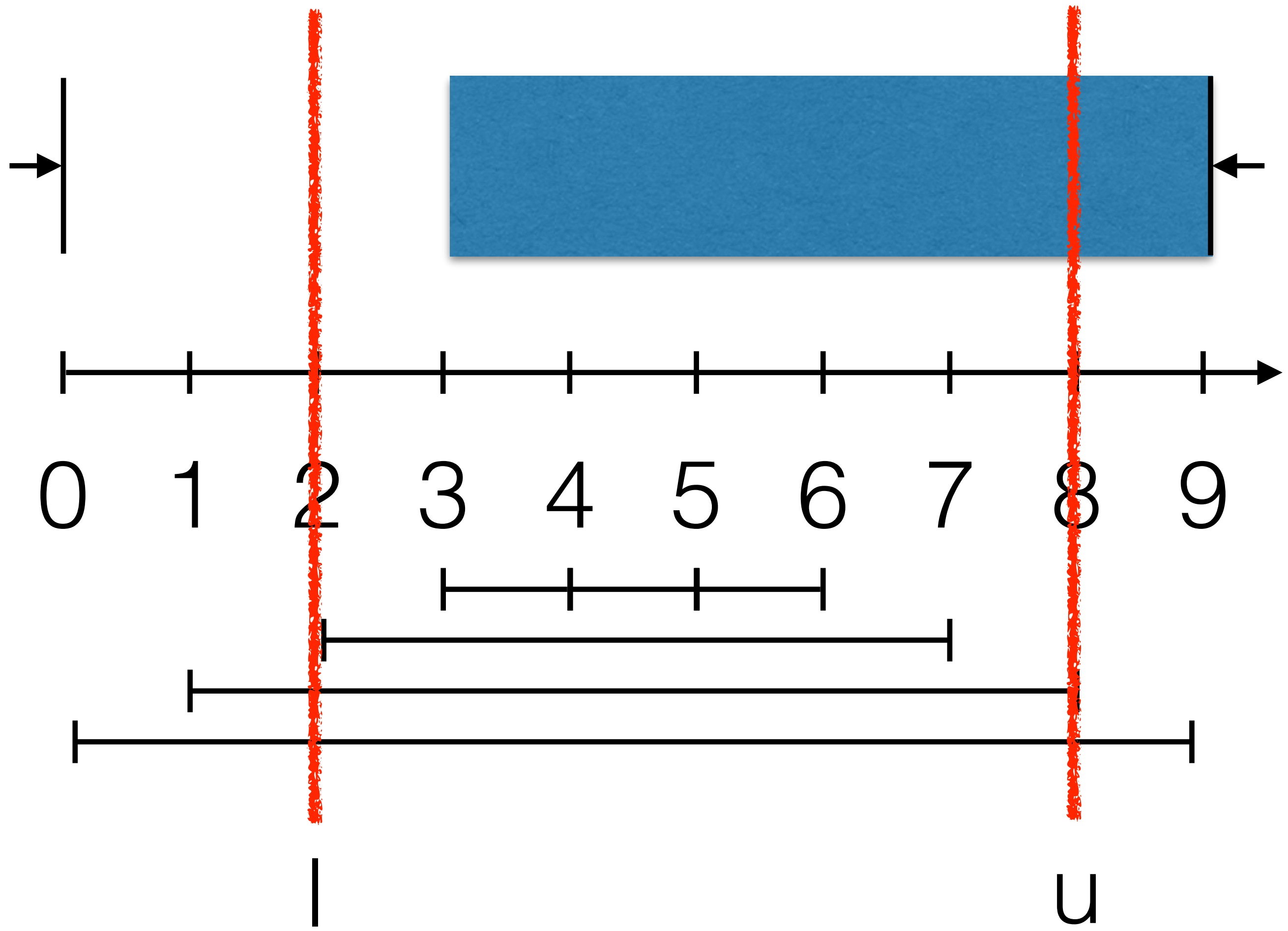


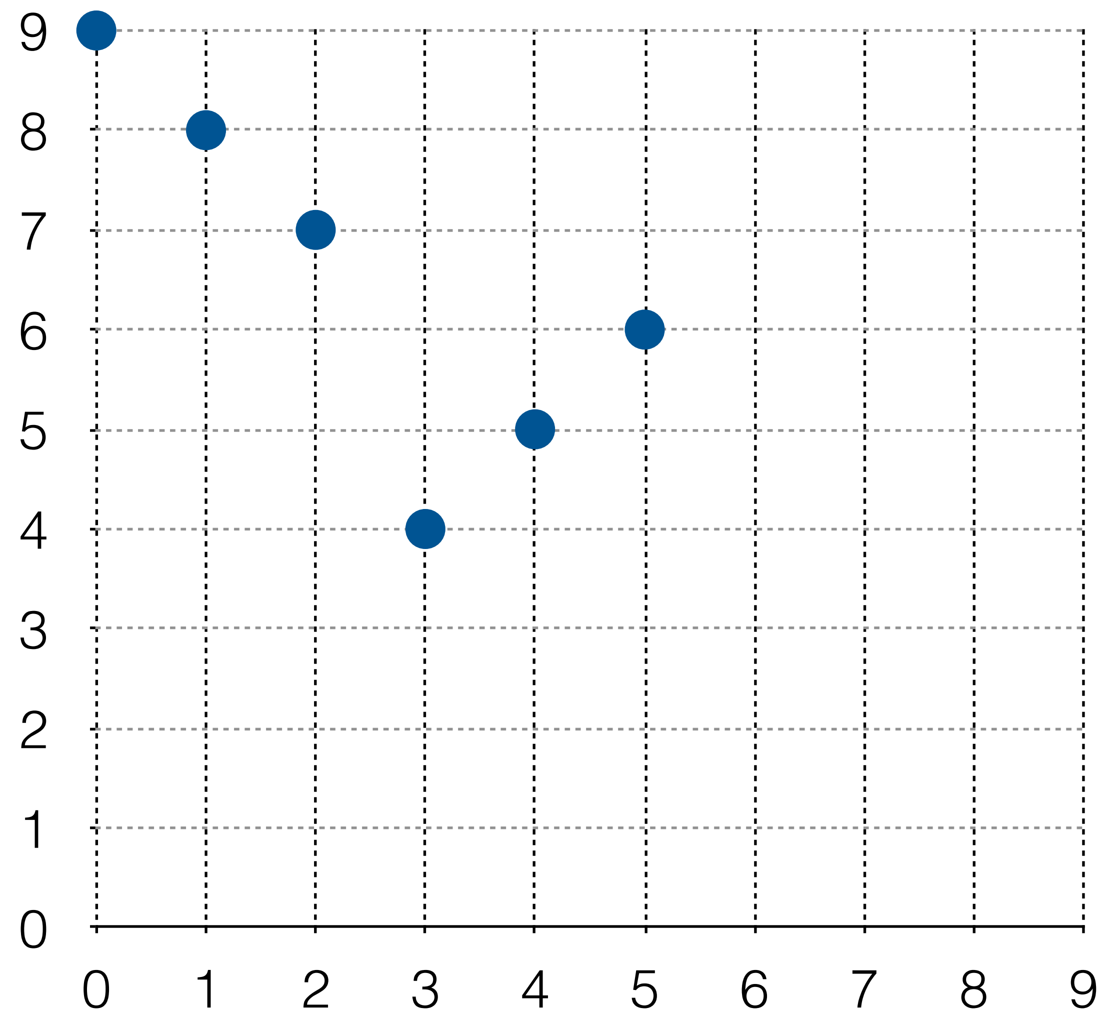
Reduction



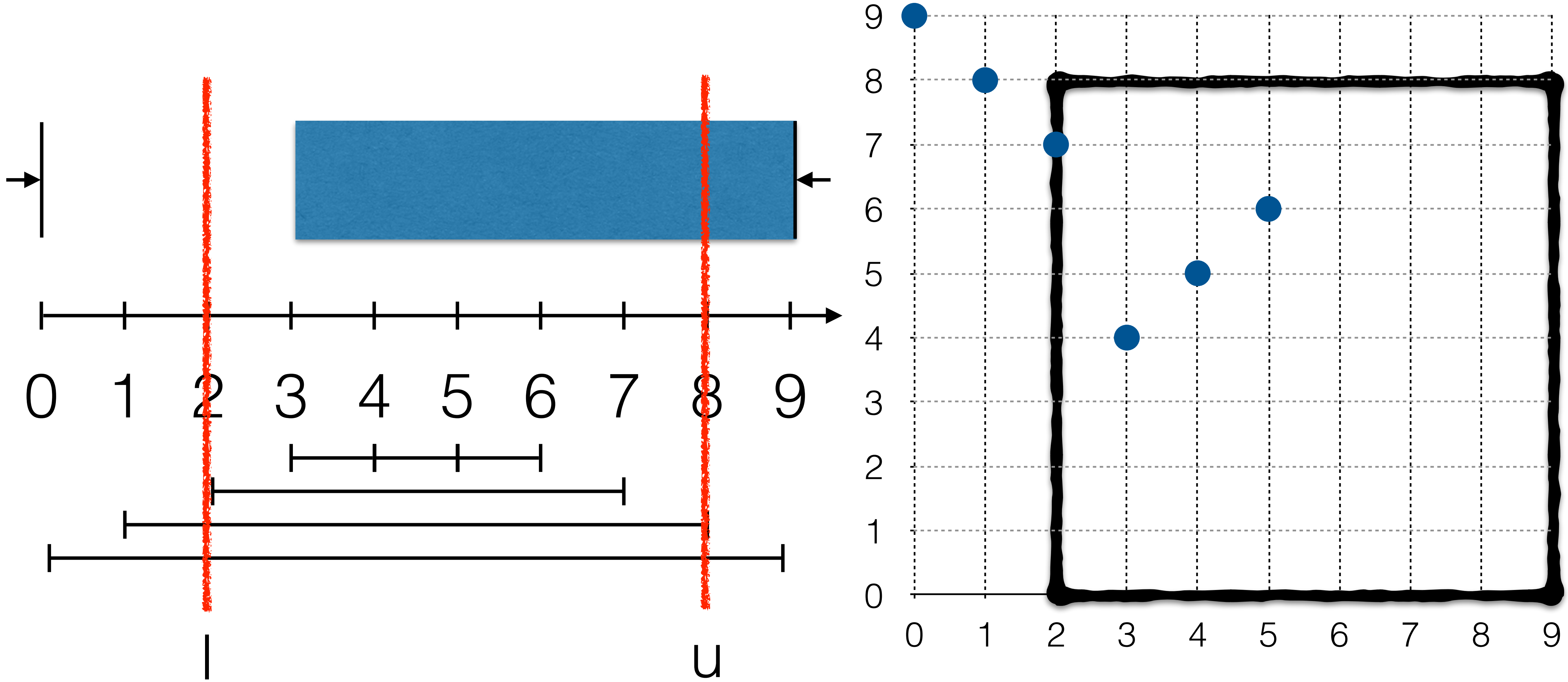


Reduction

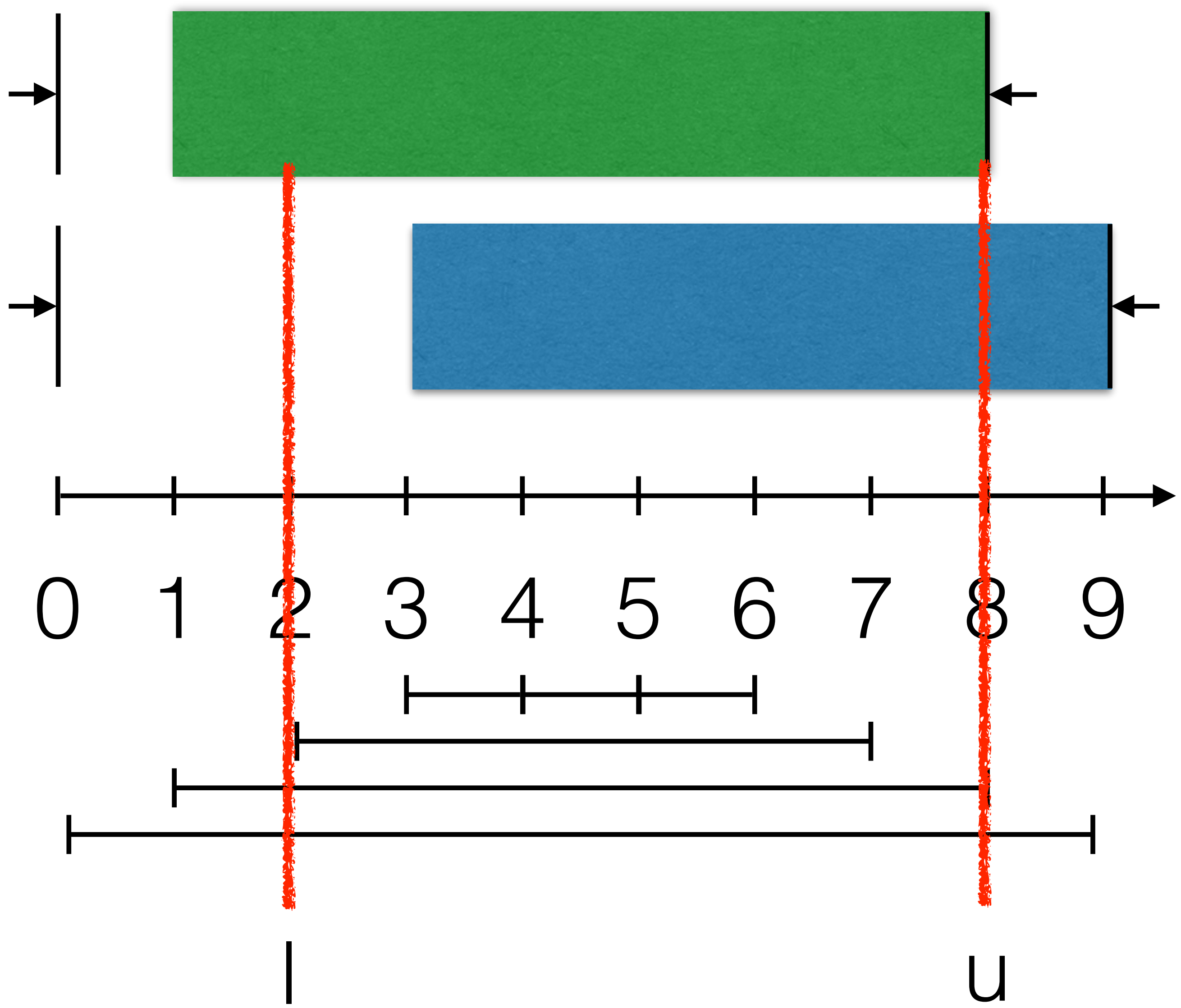


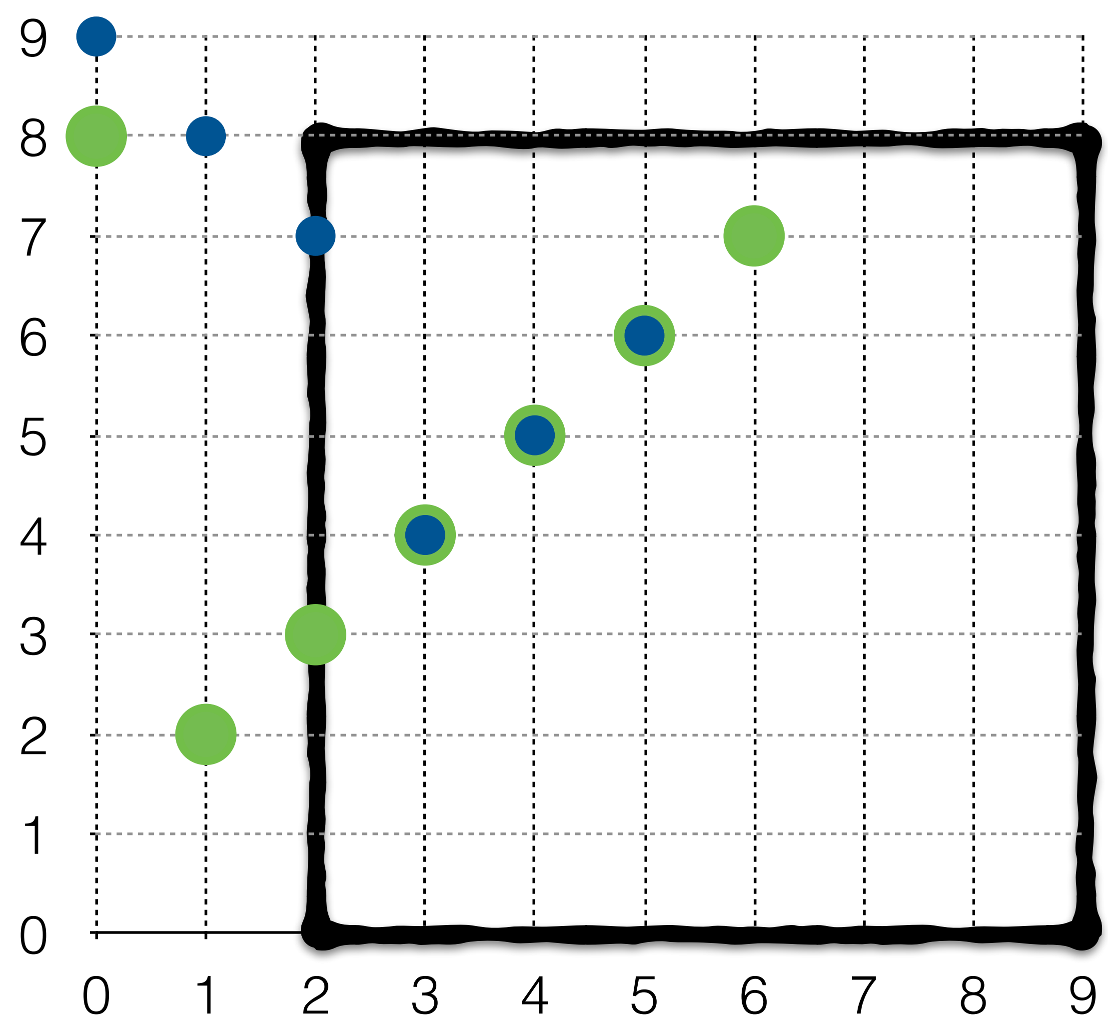


Reduction



Reduction





Range trees

- Given n points
- Build a range tree in $O(n \log n)$ space and time
- Count number of points in any given box in $O(\log n)$ time
- Since there are multiple points associated to a single task, adaptations are required to remain strongly polynomial in space and time.

How to check fewer than O(n^2) intervals?

- Recall that the slack is computed as follows.

$$S(l, u) = C \cdot (u - l) - \sum_i E(i, l, u)$$

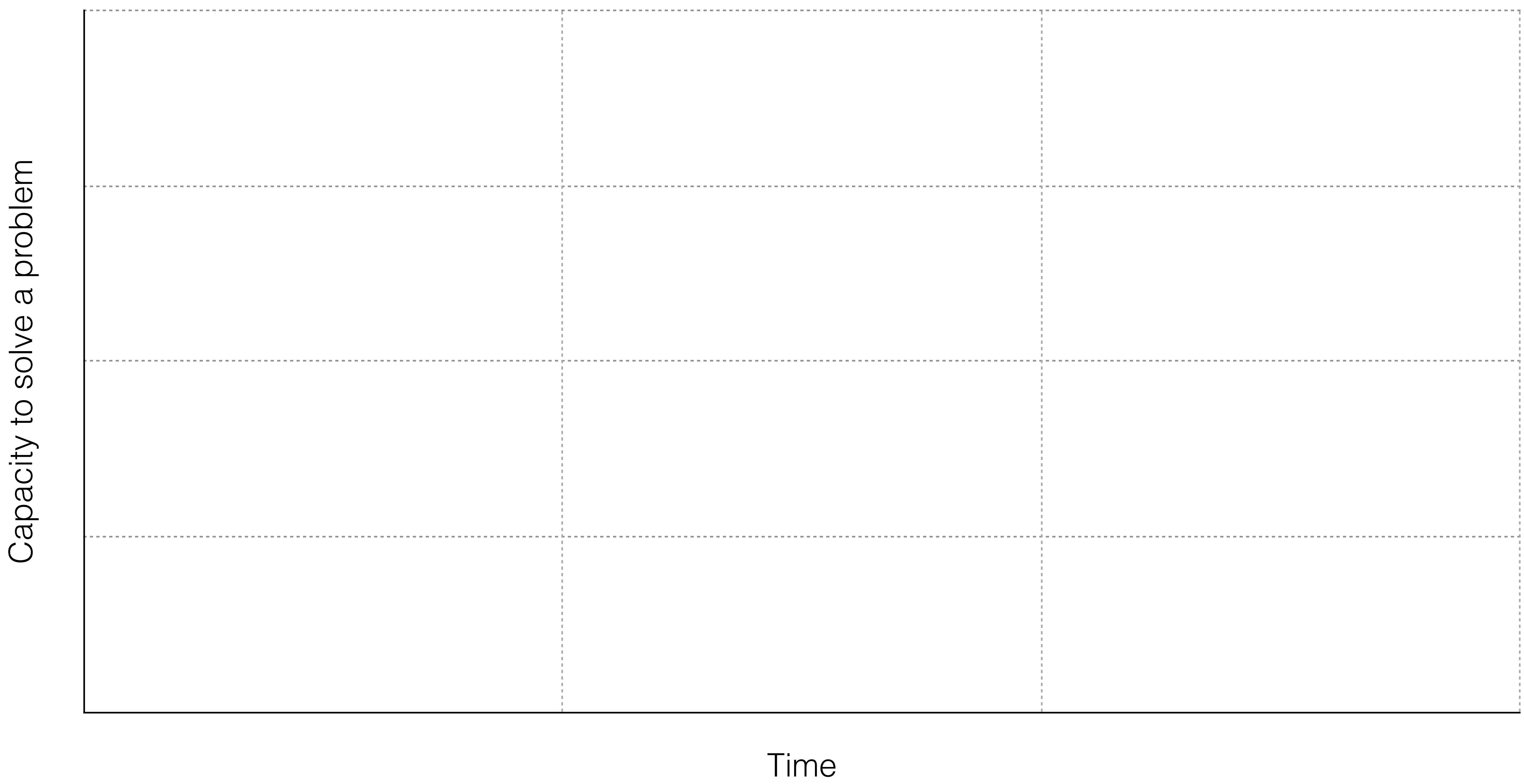
- **Goal**: Find a time interval [*l*, *u*] such that *S*(*l*, *u*) < 0 while sampling fewer than O(n^2) intervals or guaranty that there is no such interval.

Advice #2

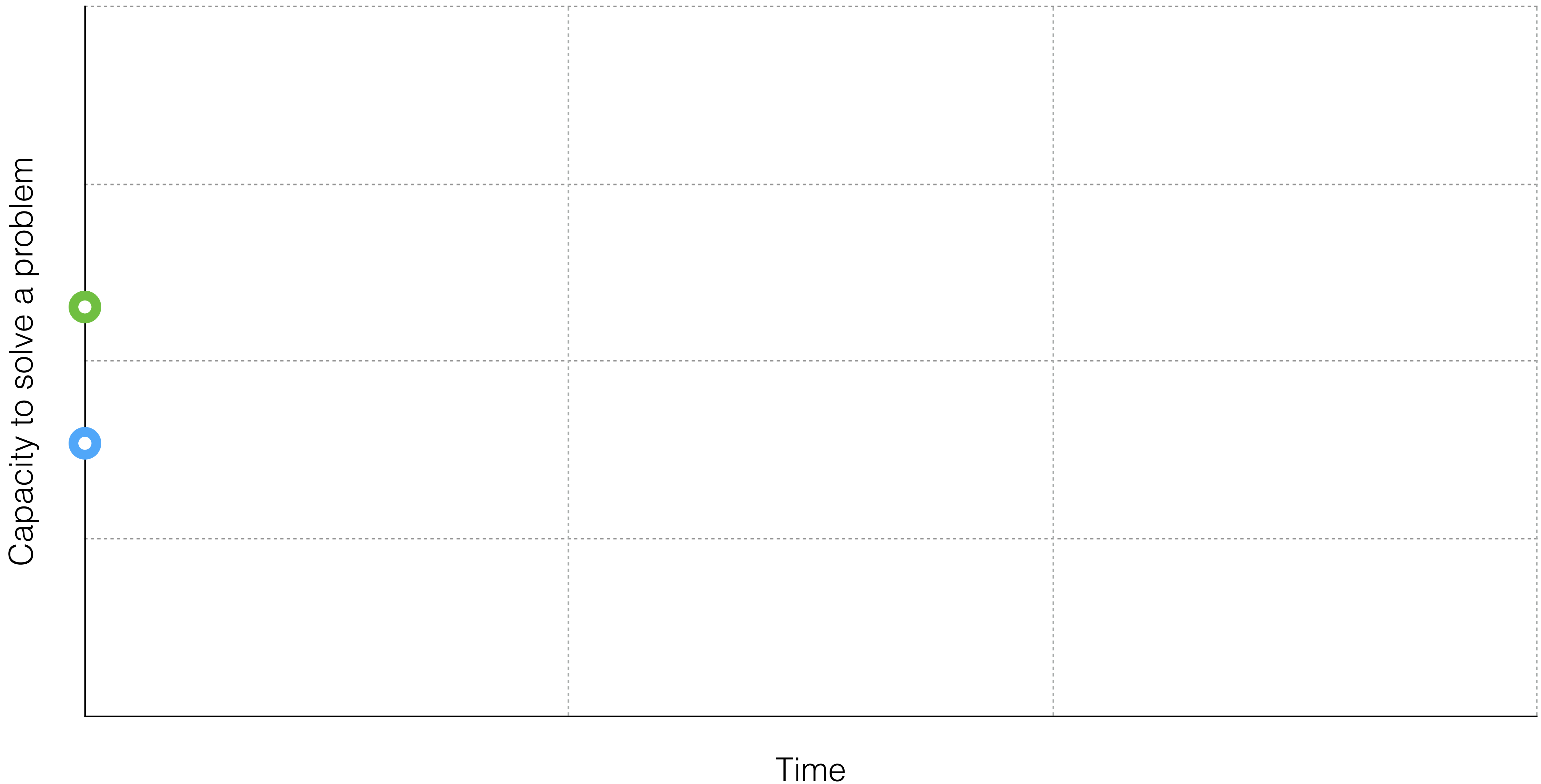
Increase your capacity to solve problems: learn new things every day.

How to improve your capacity to solve problems

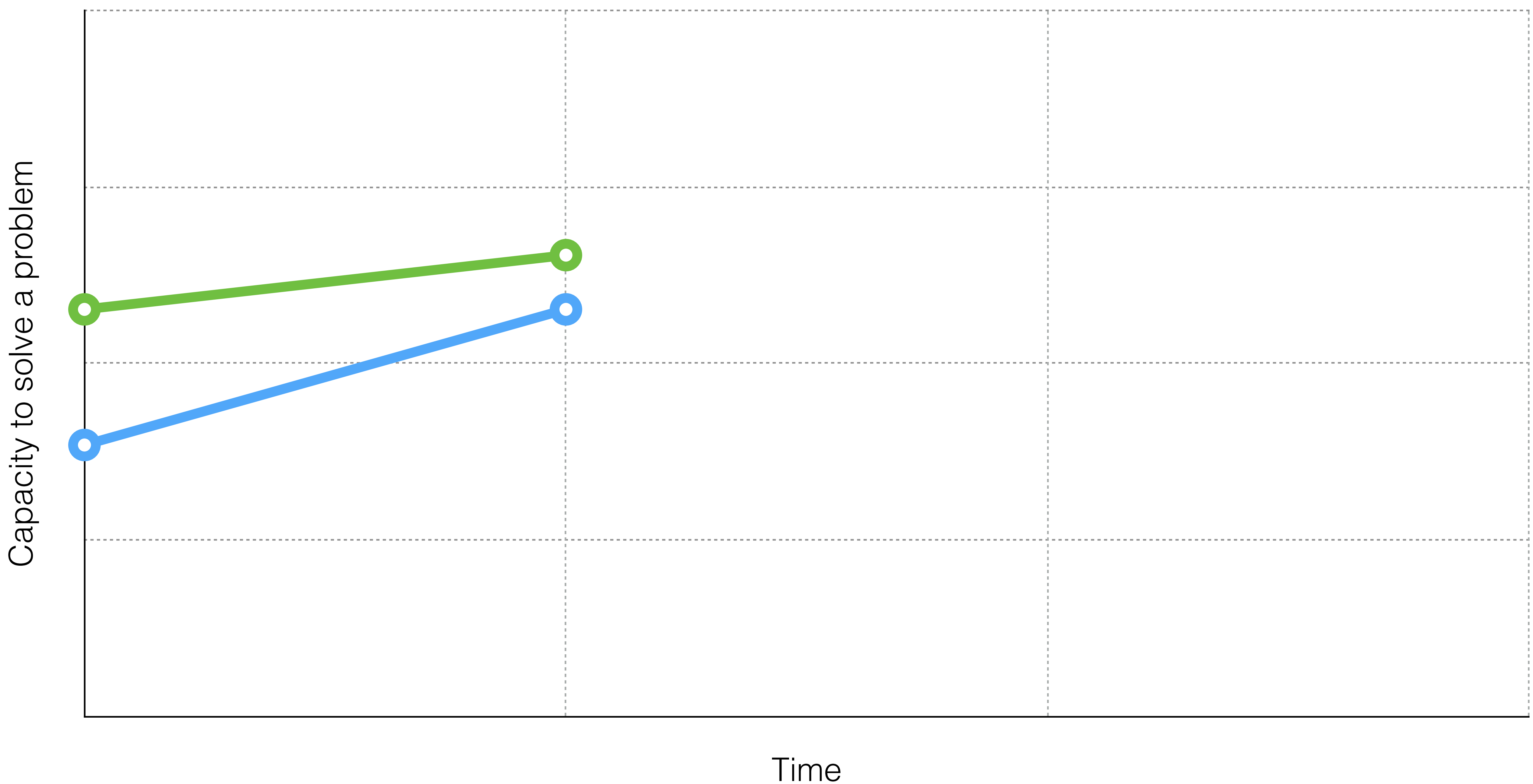
How to improve your capacity to solve problems



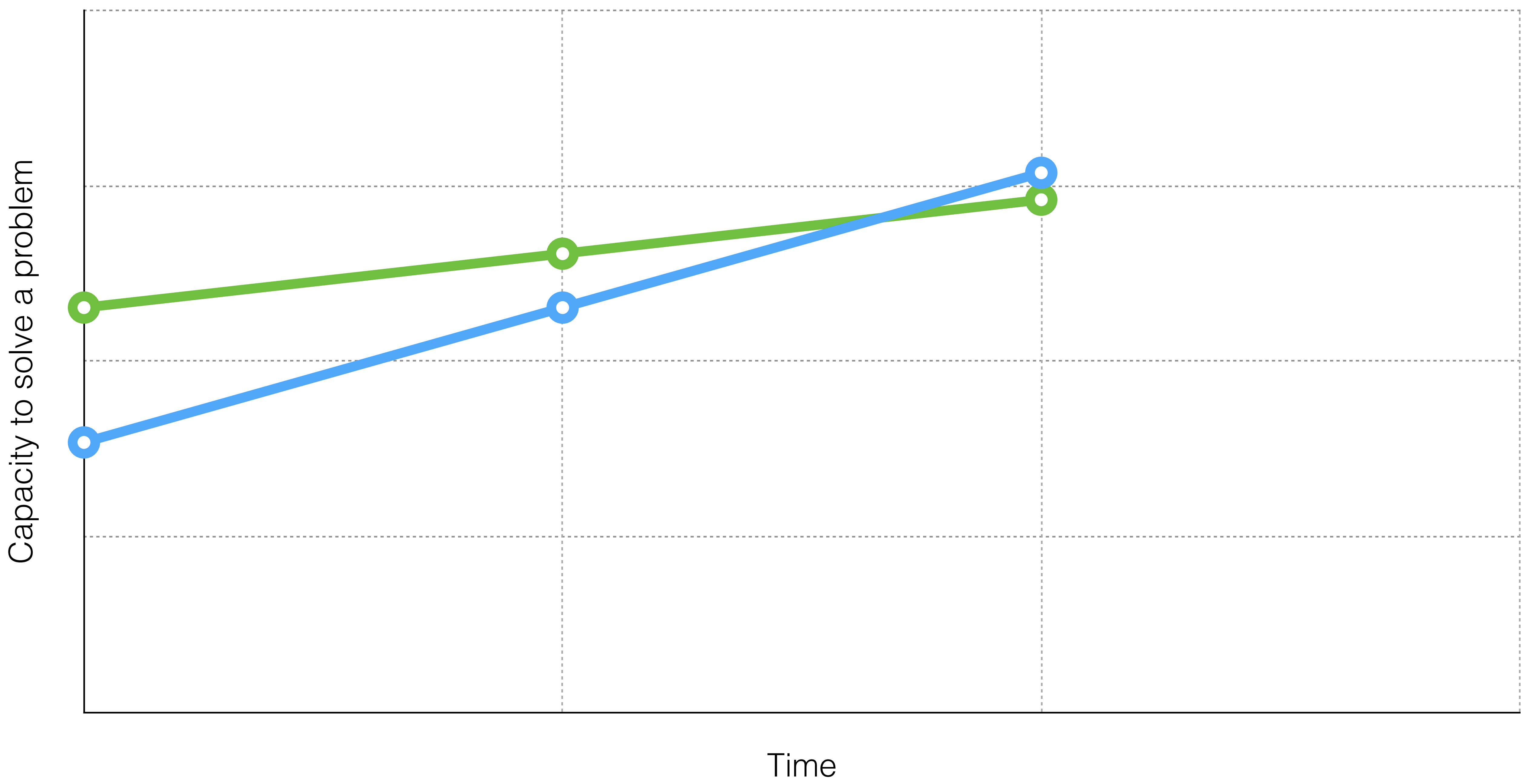
How to improve your capacity to solve problems



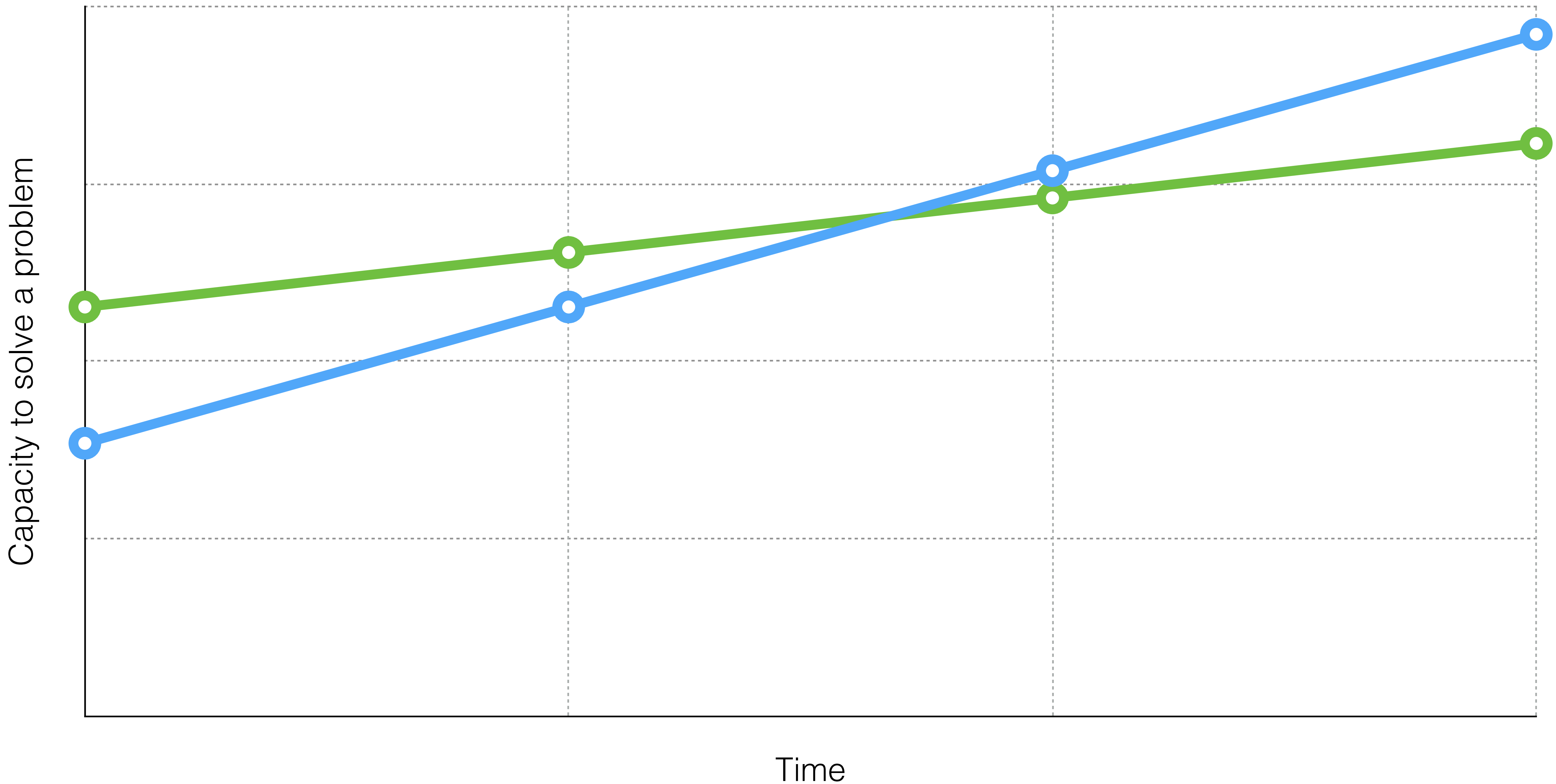
How to improve your capacity to solve problems



How to improve your capacity to solve problems



How to improve your capacity to solve problems



The slack matrix

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
0	0	2	4	6	8	10	8	8	7	6	6	8	10	12
1		0	2	4	6	8	6	6	5	4	4	6	8	10
2			0	2	4	6	4	4	3	2	2	4	6	8
3				0	2	4	2	2	1	0	0	2	4	6
4					0	2	0	0	-1	0	1	3	5	7
5						0	0	0	1	2	3	5	7	9
6							0	0	1	2	3	5	7	9
7								0	1	2	4	6	8	10
8									0	1	3	5	7	9
9										0	2	4	6	8
10											0	2	4	6
11												0	2	4
12													0	2
13														0

The slack matrix

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
0	0	2	4	6	8	10	8	8	7	6	6	8	10	12
1		0	2	4	6	8	6	6	5	4	4	6	8	10
2			0	2	4	6	4	4	3	2	2	4	6	8
3				0	2	4	2	2	1	0	0	2	4	6
4					0	2	0	0	-1	0	1	3	5	7
5						0	0	0	1	2	3	5	7	9
6							0	0	1	2	3	5	7	9
7								0	1	2	4	6	8	10
8									0	1	3	5	7	9
9										0	2	4	6	8
10											0	2	4	6
11												0	2	4
12													0	2
13														0

Monge matrix

- A matrix M has the Monge property if and only if

$$M[i+1, j+1] - M[i+1, j] \geq M[i, j+1] - M[i, j]$$

Monge matrix

- A matrix M has the Monge property if and only if

$$M[i+1, j+1] - M[i+1, j] \geq M[i, j+1] - M[i, j]$$

- I did not get that out from a hat. I learned it **15 years** ago!

Monge matrix

- A matrix M has the Monge property if and only if

$$M[i+1, j+1] - M[i+1, j] \geq M[i, j+1] - M[i, j]$$

- I did not get that out from a hat. I learned it **15 years** ago!
- The Monge property can make many problems easier:

Monge matrix

- A matrix M has the Monge property if and only if

$$M[i+1, j+1] - M[i+1, j] \geq M[i, j+1] - M[i, j]$$

- I did not get that out from a hat. I learned it **15 years** ago!
- The Monge property can make many problems easier:
  - Finding the smallest element on every row of a Monge matrix can be done in linear time!

Monge matrix

- A matrix M has the Monge property if and only if

$$M[i+1, j+1] - M[i+1, j] \geq M[i, j+1] - M[i, j]$$

- I did not get that out from a hat. I learned it **15 years** ago!
- The Monge property can make many problems easier:
  - Finding the smallest element on every row of a Monge matrix can be done in linear time!
  - Solving the Traveling Salesman Problem can be done linear time!

Inverse Monge matrix

Inverse

- A matrix M has the Monge property if and only if

$$M[i+1,j+1] - M[i+1,j] \geq M[i,j+1] - M[i,j]$$

- I did not get that out from a hat. I learned it **15 years** ago!
- The Monge property can make many problems easier:
  - Finding the smallest element on every row of a Monge matrix can be done in linear time!
  - Solving the Traveling Salesman Problem can be done linear time!

Inverse Monge matrix

$$M[i+1, j+1] - M[i+1, j] \geq M[i, j+1] - M[i, j]$$

- To understand the intuition of Monge matrices, consider the i^{th} row of a matrix as a function f_i.

$$\frac{f_{i+1}(x+1) - f_{i+1}(x)}{(x+1) - x} \geq \frac{f_i(x+1) - f_i(x)}{(x+1) - x}$$

- Function f_{i+1} grows faster than function f_i.
- Consequently, both functions cross each other only once.
- The crossing point (or region) can be computed with a binary search.

The slack matrix

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

The slack matrix

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

The slack matrix

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

The slack matrix

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

The slack matrix

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

The slack matrix

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

The slack matrix

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
0	0	2	4	6	8	10	8	8	7	6	6	8	10	12
1		0	2	4	6	8	6	6	5	4	4	6	8	10
2			0	2	4	6	4	4	3	2	2	4	6	8
3				0	2	4	2	2	1	0	0	2	4	6
4					0	2	0	0	-1	0	1	3	5	7
5						0	0	0	1	2	3	5	7	9
6							0	0	1	2	3	5	7	9
7								0	1	2	4	6	8	10
8									0	1	3	5	7	9
9										0	2	4	6	8
10											0	2	4	6
11												0	2	4
12													0	2
13														0

The slack matrix

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

The slack matrix

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

Complexity

- If the matrix had dimension $n \times n$, there would be

Complexity

- If the matrix had dimension $n \times n$, there would be
  - O(n) binary searches

Complexity

- If the matrix had dimension $n \times n$, there would be
  - $O(n)$ binary searches
  - $O(n \log n)$ comparisons

Complexity

- If the matrix had dimension $n \times n$, there would be
  - $O(n)$ binary searches
  - $O(n \log n)$ comparisons
  - Computing an entry of the matrix takes $O(\log n)$

Complexity

- If the matrix had dimension $n \times n$, there would be
  - $O(n)$ binary searches
  - $O(n \log n)$ comparisons
  - Computing an entry of the matrix takes $O(\log n)$
  - Overall complexity: $O(n \log^2 n)$

Complexity

- If the matrix had dimension $n \times n$, there would be
  - $O(n)$ binary searches
  - $O(n \log n)$ comparisons
  - Computing an entry of the matrix takes $O(\log n)$
  - Overall complexity: $O(n \log^2 n)$
- But the dimension is not $n \times n$ …

Complexity

- If the matrix had dimension $n \times n$, there would be
  - $O(n)$ binary searches
  - $O(n \log n)$ comparisons
  - Computing an entry of the matrix takes $O(\log n)$
  - Overall complexity: $O(n \log^2 n)$
- But the dimension is not $n \times n$ …
- To obtain a complexity of $O(n \log^2 n)$, the algorithm only analyzes a subset of $O(n^2)$ cells characterized by Derrien and Petit.

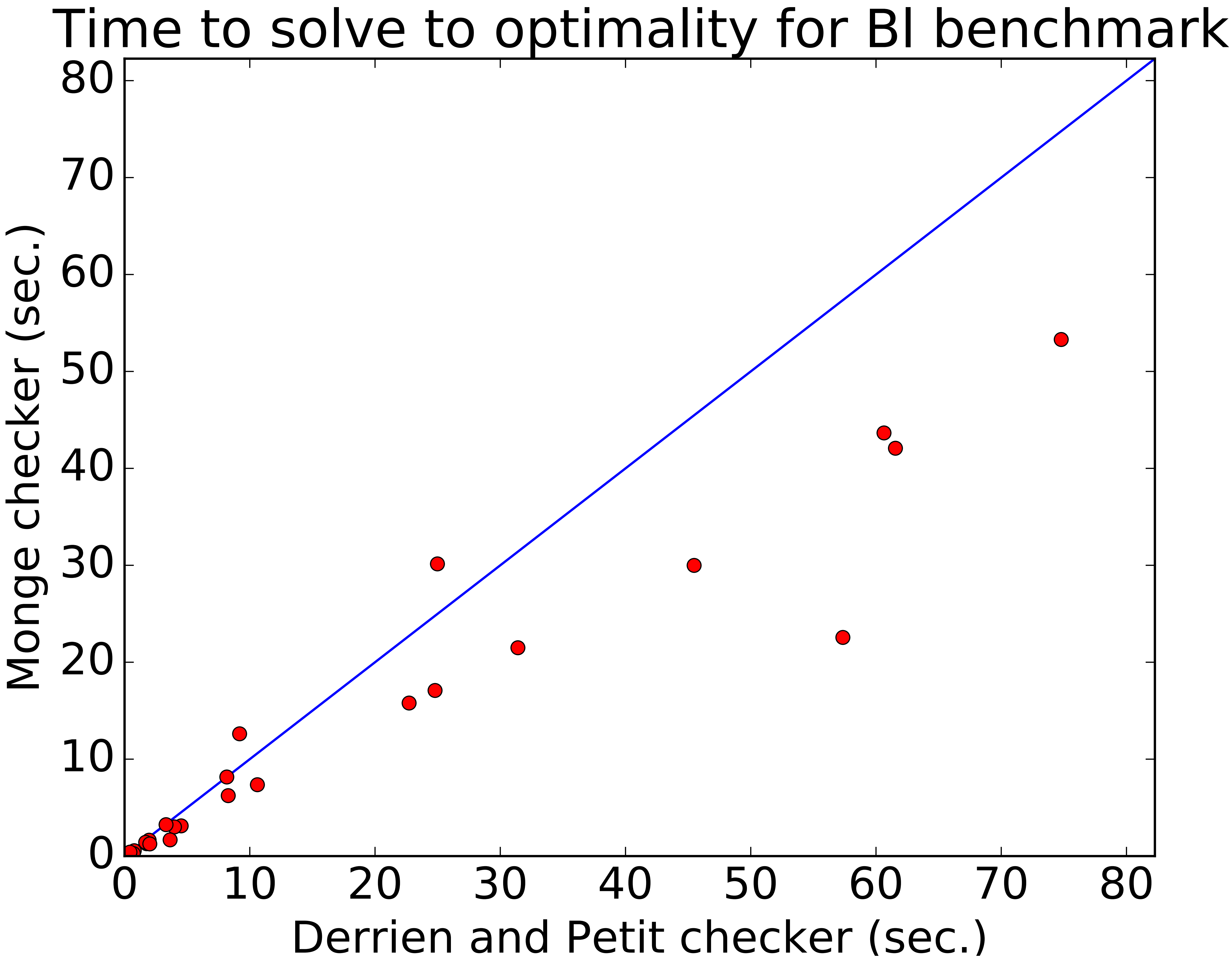
Advice #3

Make your research practical!

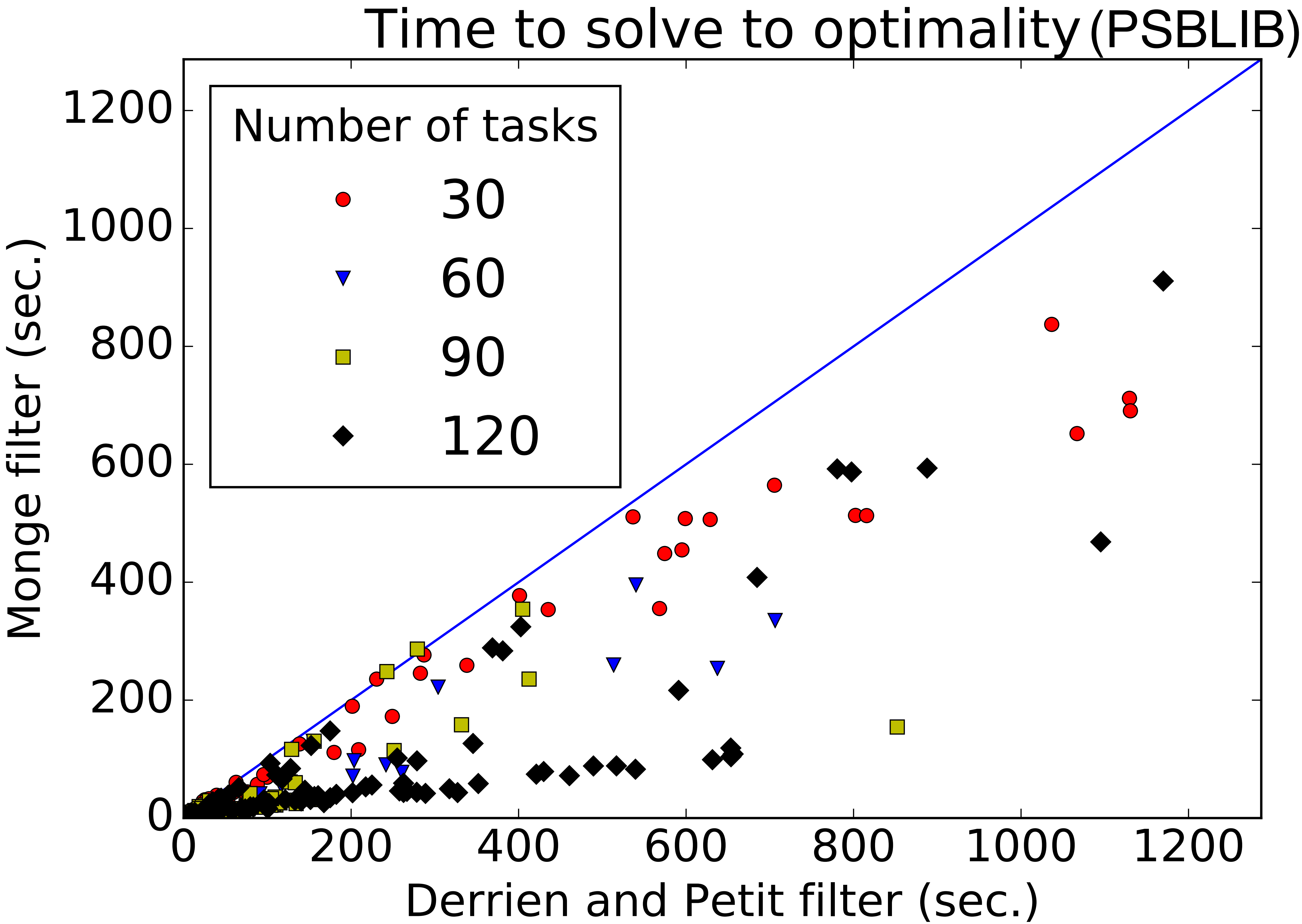
- If you want your graduate studies to leverage your industrial career, work on something practical.
- You prefer an academic career? Having industrial partners will help fund your lab.
- You prefer theory? No problem! Be prepared to justify with applications.
- Implement your ideas!

What we learned when implementing the algorithm

- A large portion of the computation is spent computing entries in the slack matrix.
    - Adding a cache prevents computing twice the same slack and save computation time.
- Derrien et Petit reduced the number of intervals of interests by a factor 7. This makes a huge difference for our checker as well.



Time to solve to optimality for BI benchmark
Monge checker (sec.)
Derrien and Petit checker (sec.)
0
10
20
30
40
50
60
70
80



Time to solve to optimality (PSBLIB)
Number of tasks
30
60
90
120
Monge filter (sec.)
Derrien and Petit filter (sec.)
0
200
400
600
800
1000
1200

Next steps

- Running time analysis of the filtering algorithm

Next steps

- Running time analysis of the filtering algorithm
  - Worst case: $O(n^2 \log^2 n)$

Next steps

- Running time analysis of the filtering algorithm
  - Worst case: $O(n^2 \log^2 n)$

We believe this bound is not tight

Next steps

- Running time analysis of the filtering algorithm
  - Worst case: $O(n^2 \log^2 n)$
  - Average case: $O(n^2 \log n)$

We believe this bound is not tight

Next steps

- Running time analysis of the filtering algorithm
  - Worst case: $O(n^2 \log^2 n)$
  - Average case: $O(n^2 \log n)$

We believe this bound is not tight

We believe this bound is not tight

Next steps

- Running time analysis of the filtering algorithm
  - Worst case: $O(n^2 \log^2 n)$
  - Average case: $O(n^2 \log n)$
- Nogood learning

We believe this bound is not tight

We believe this bound is not tight

Advice #4

Share your ideas

Conclusion

- Range trees are convenient to compute the amount of energy in a given time interval.
- The Monge property appears in scheduling problems and can be exploited.
- Energetic check in $O(n \log^2 n)$ time.

Advice #1

Reduce your problem to one that is already solved.

Advice #2

Increase your capacity to solve problems: learn new things every day.

Advice #3

Make it practical!

Advice #4

Share your ideas