Time-Table-Extended-Edge-Finding for the Cumulative Constraint

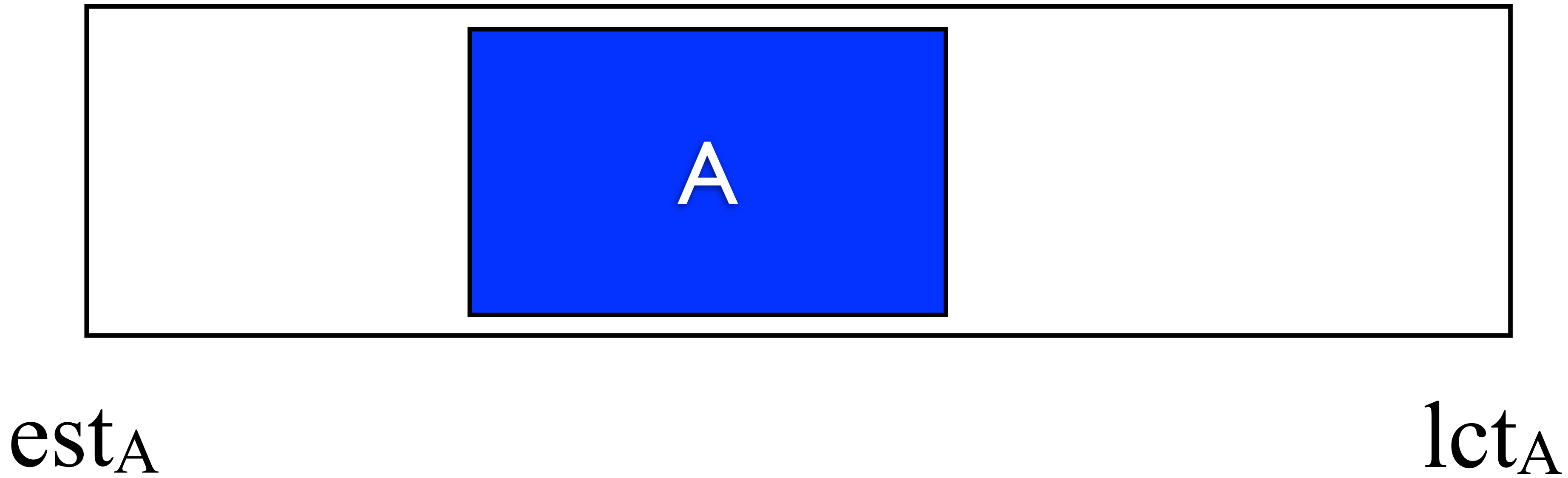
Pierre Ouellet and Claude-Guy Quimper

Université Laval
Québec, Canada

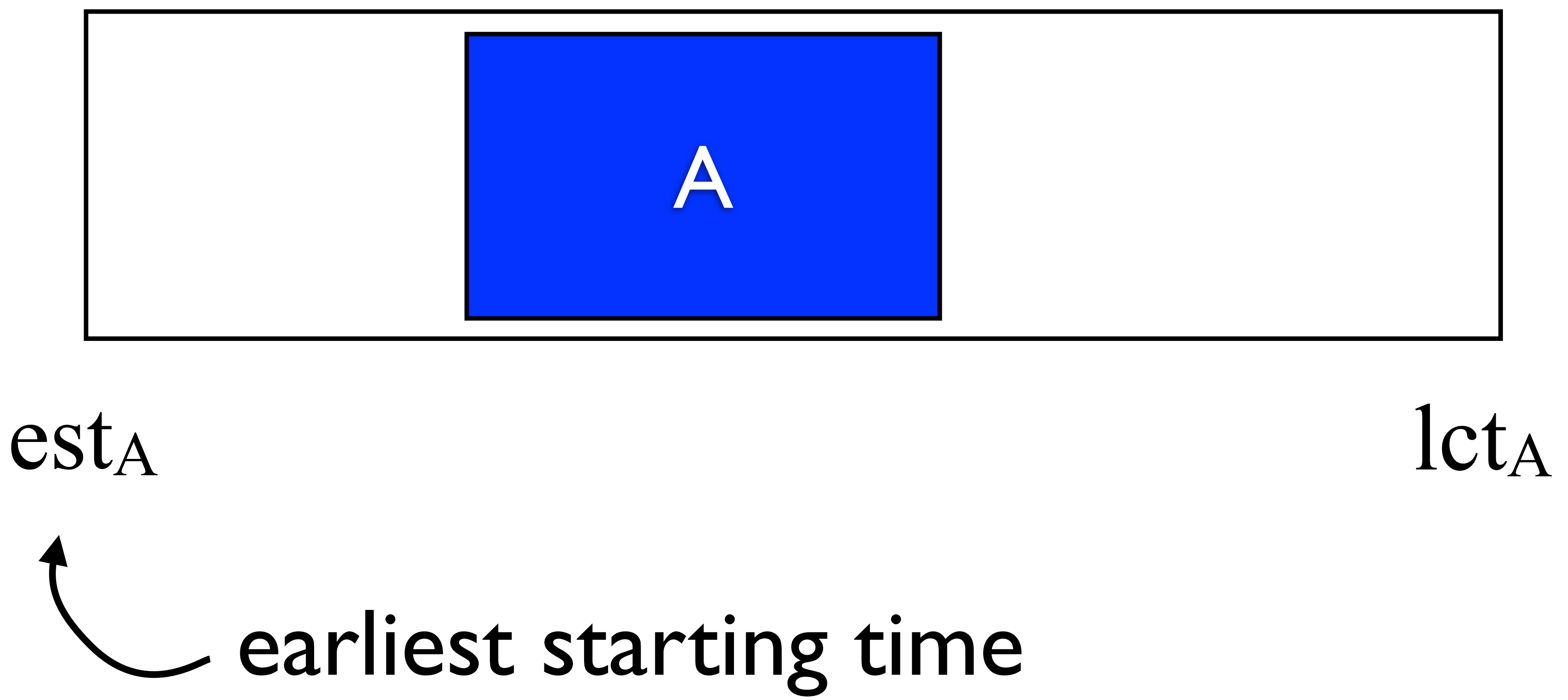
Introduction

- We present new filtering algorithms for the Cumulative constraint.
  - An Extended-Edge-Finder.
  - A Time-Table algorithm.
  - A Time-Table-Extended-Edge-Finder.

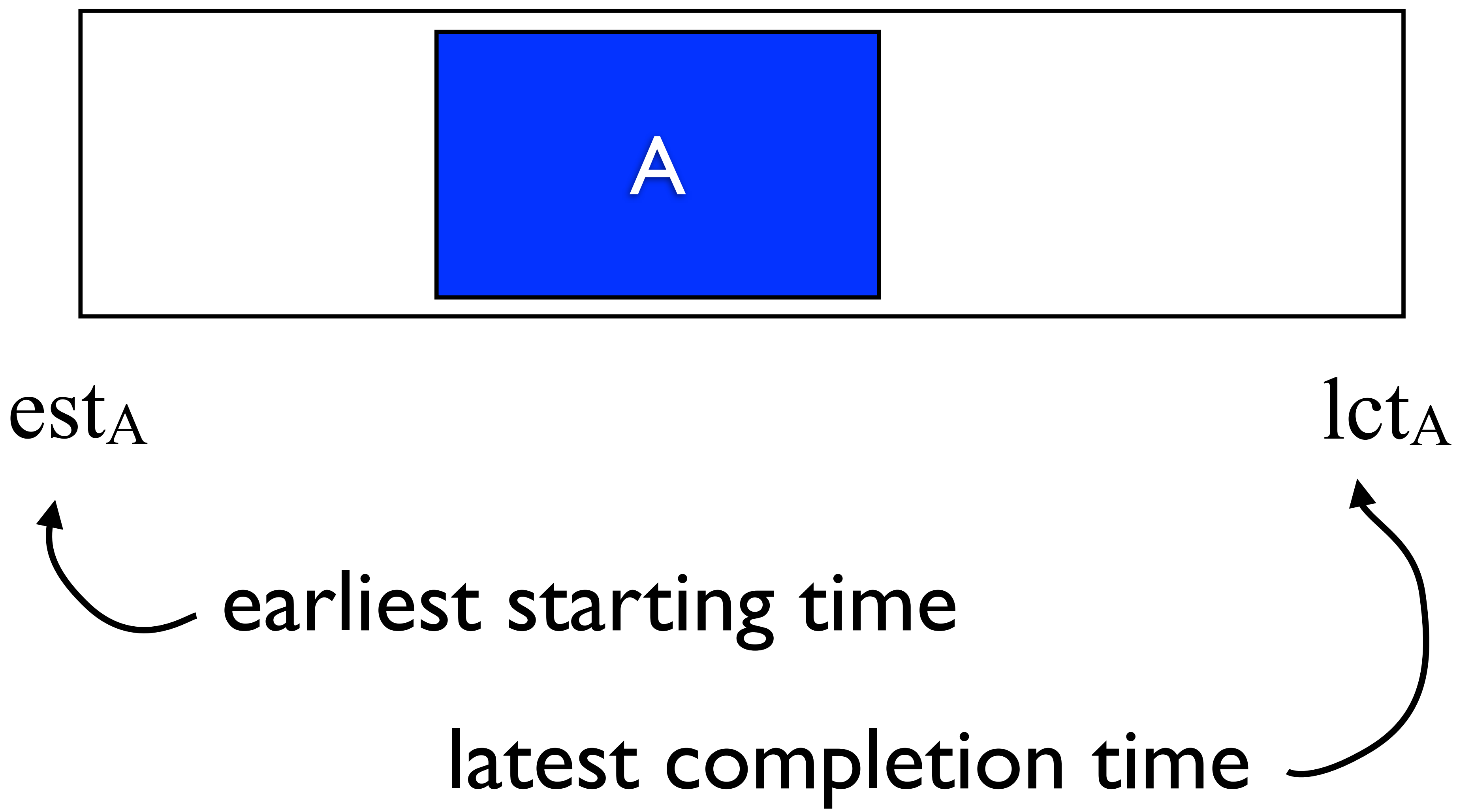
The Task



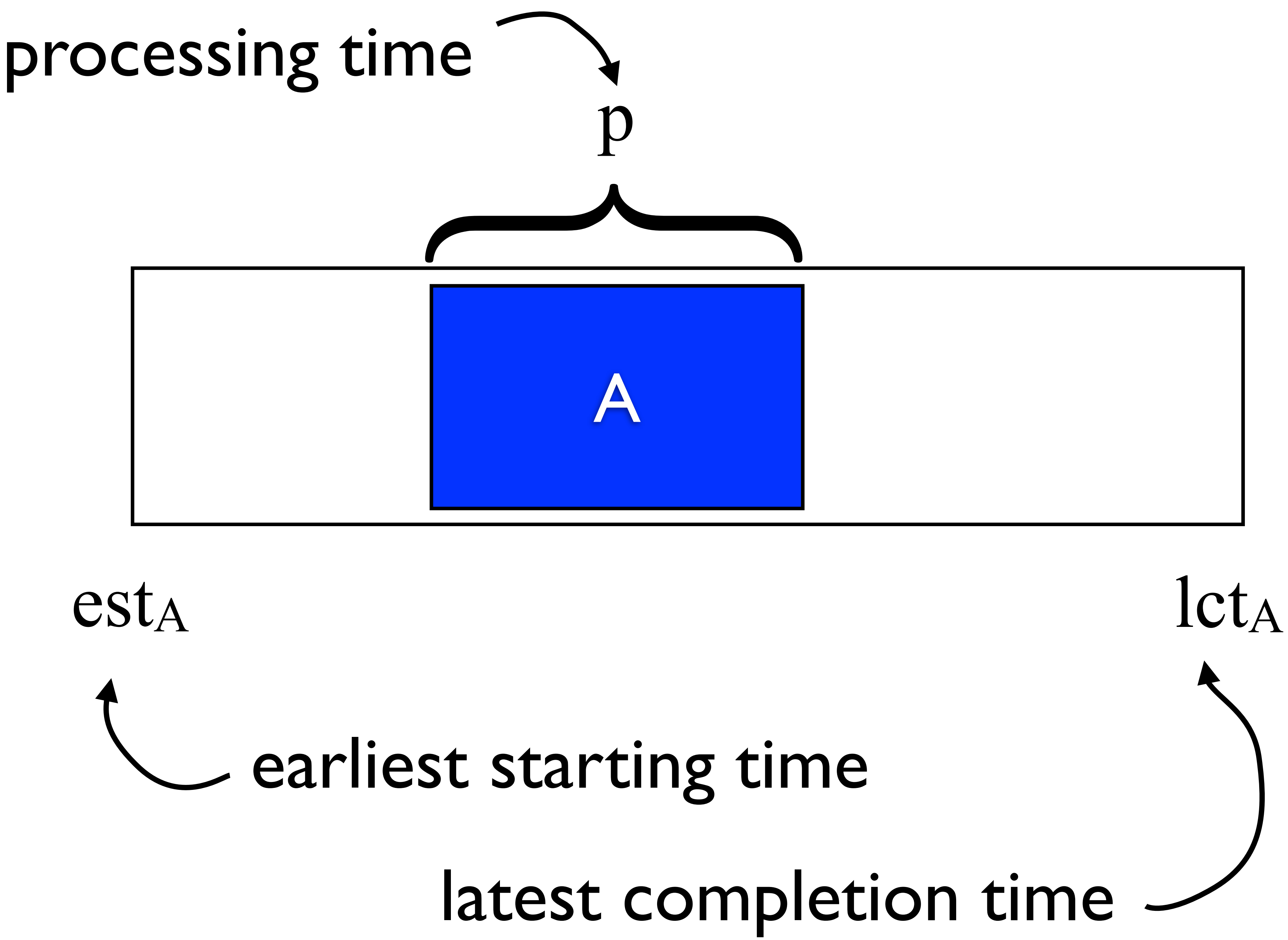
The Task



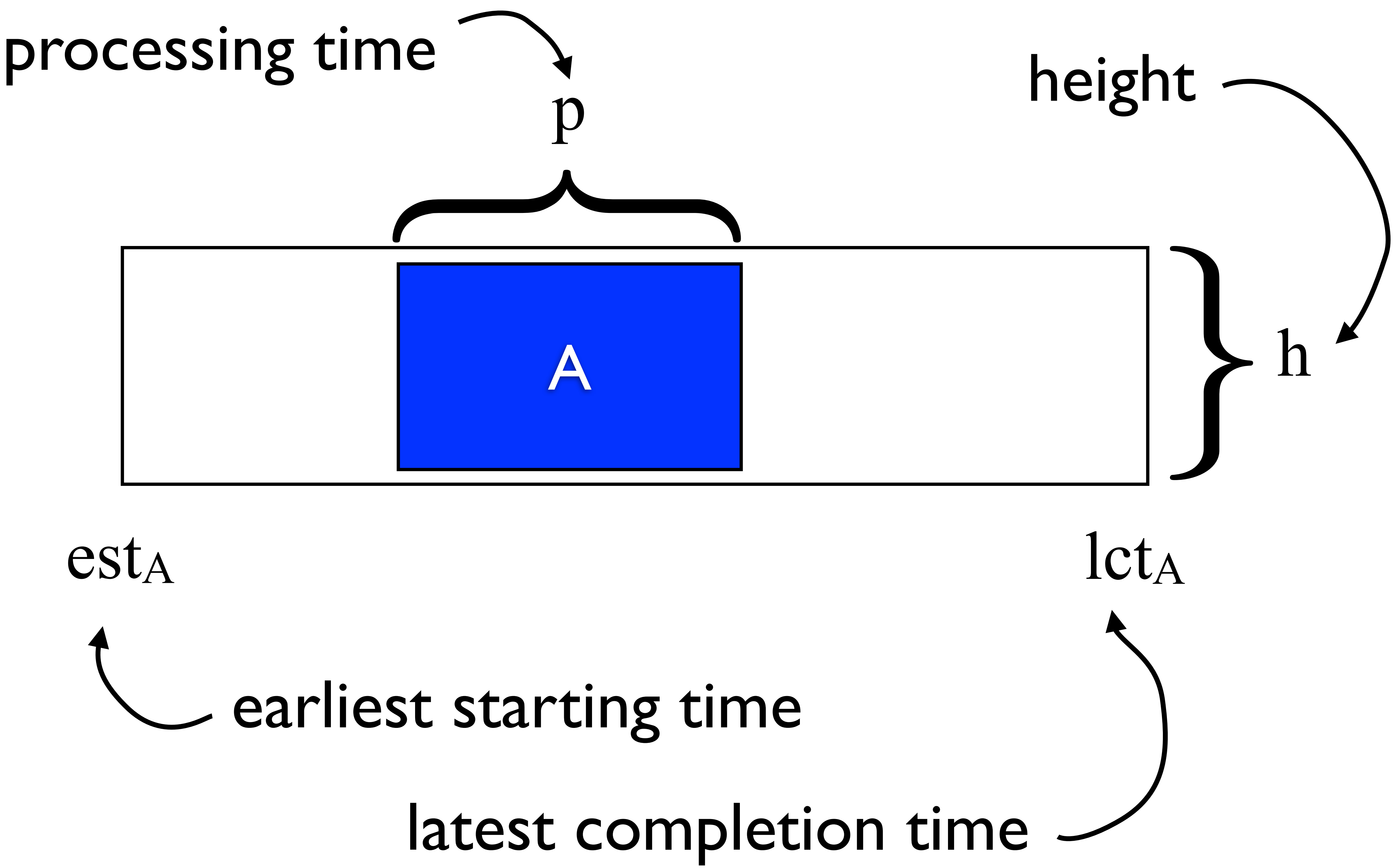
The Task



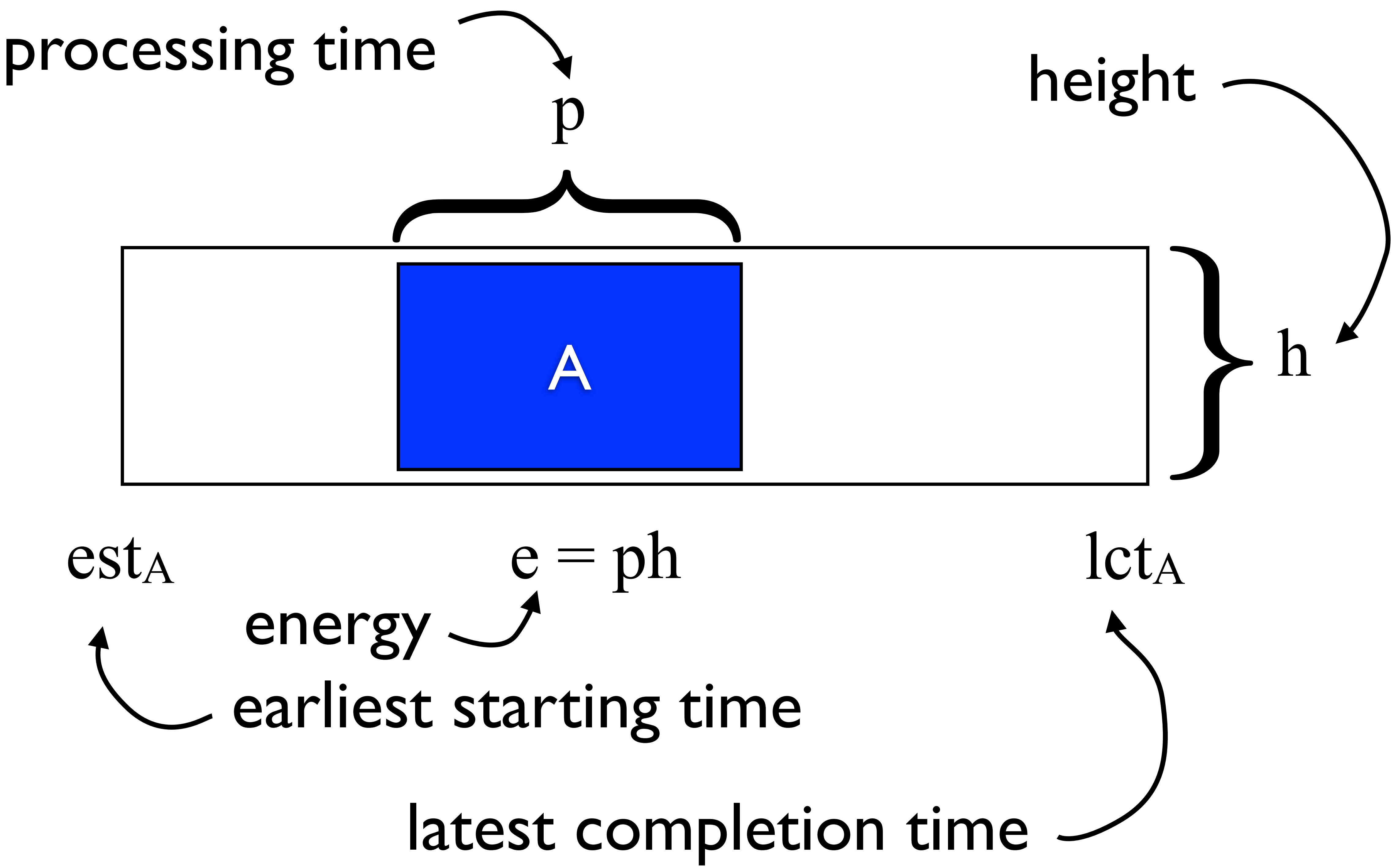
The Task


processing time
p
A
est_A
lct_A
earliest starting time
latest completion time

The Task

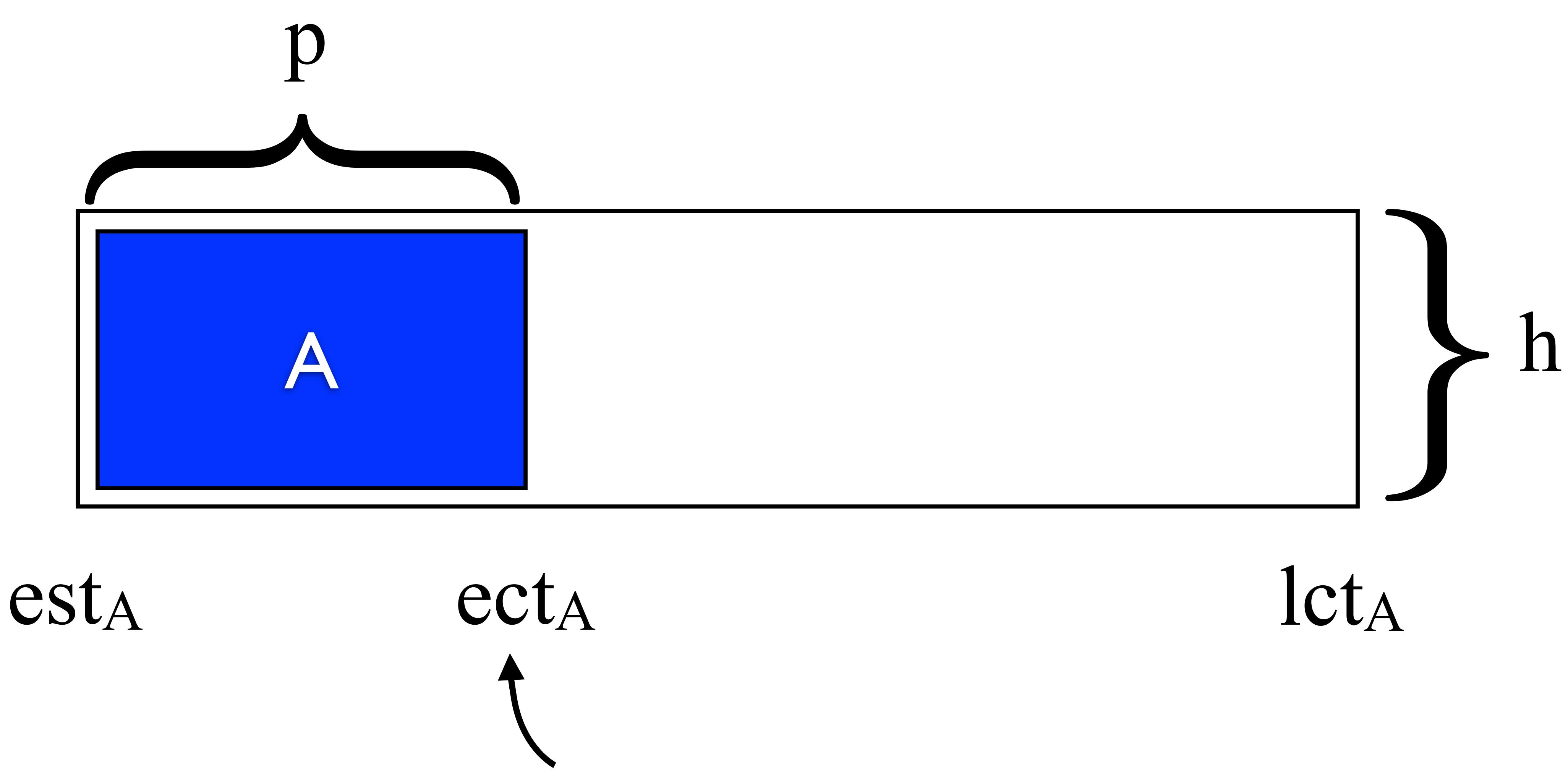

processing time
p
height
A
h
est_A
lct_A
earliest starting time
latest completion time

The Task



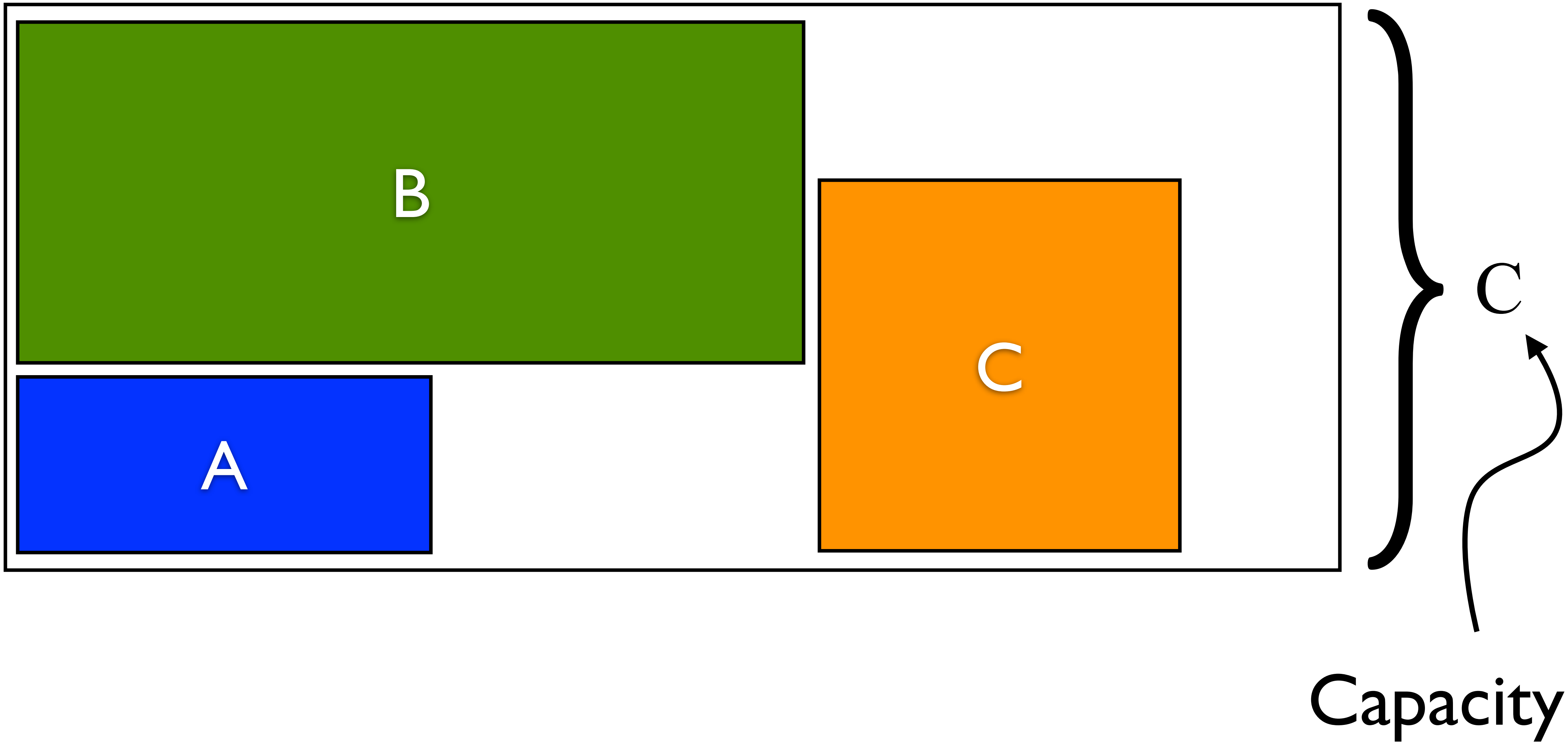
processing time
p
height
A
h
est_A
$e = ph$
lct_A
energy
earliest starting time
latest completion time

The Task

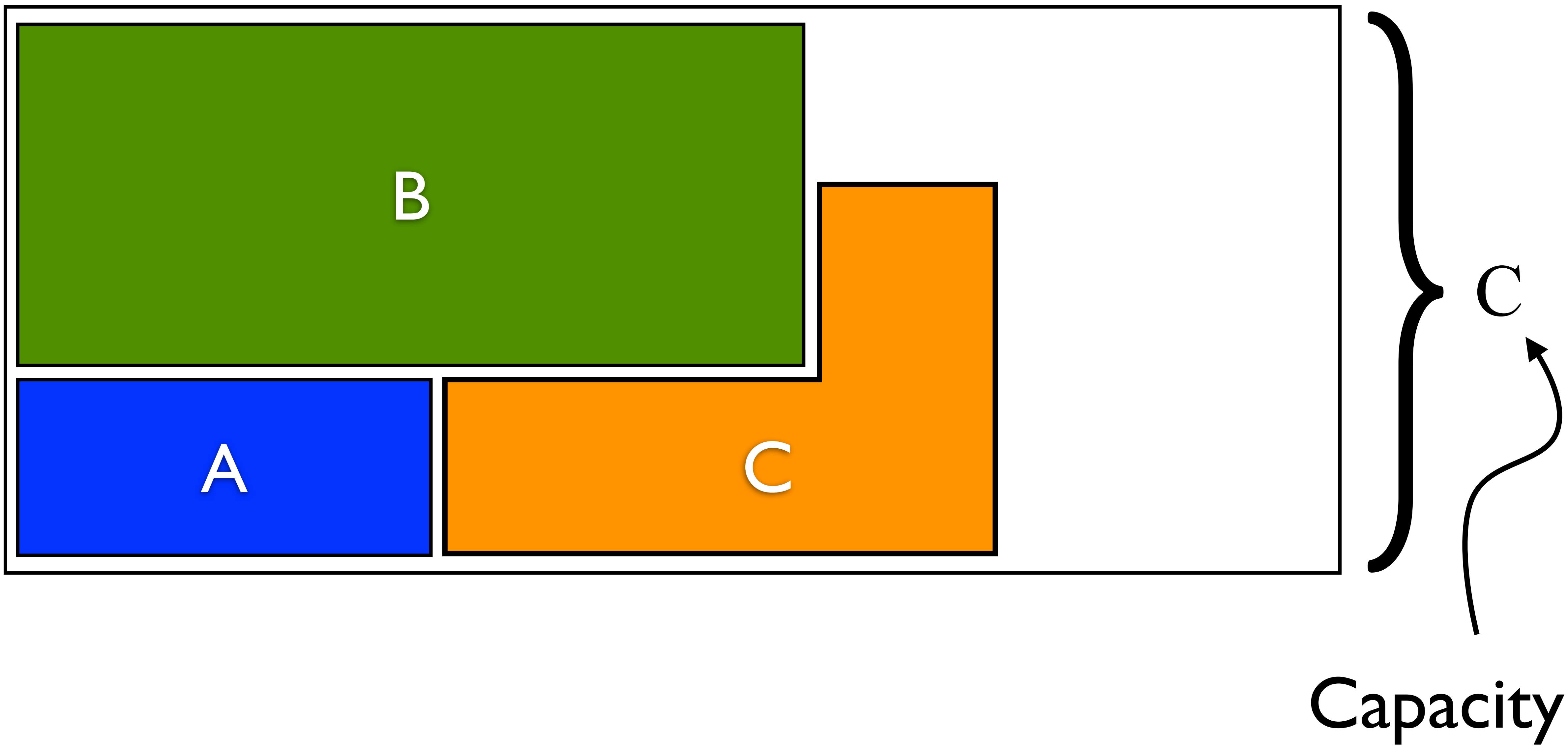


earliest completion time

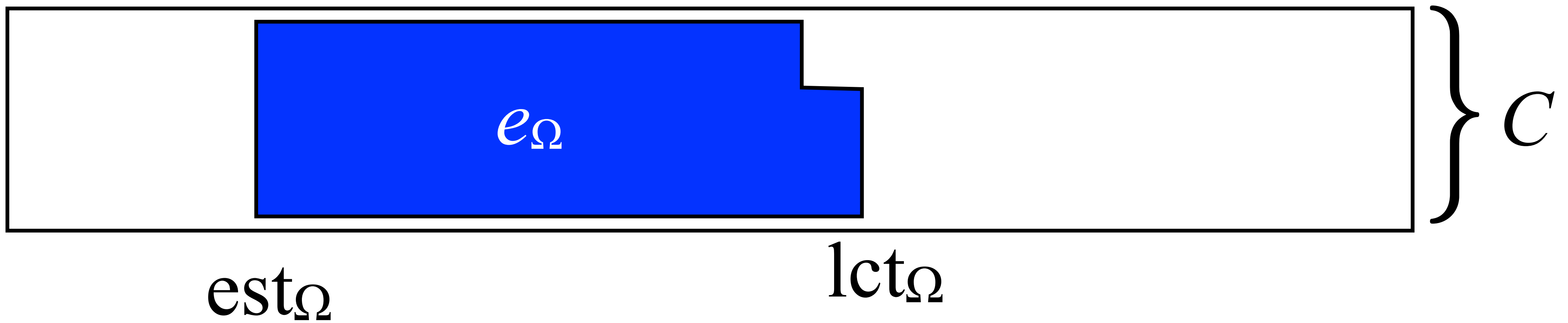
The Resource



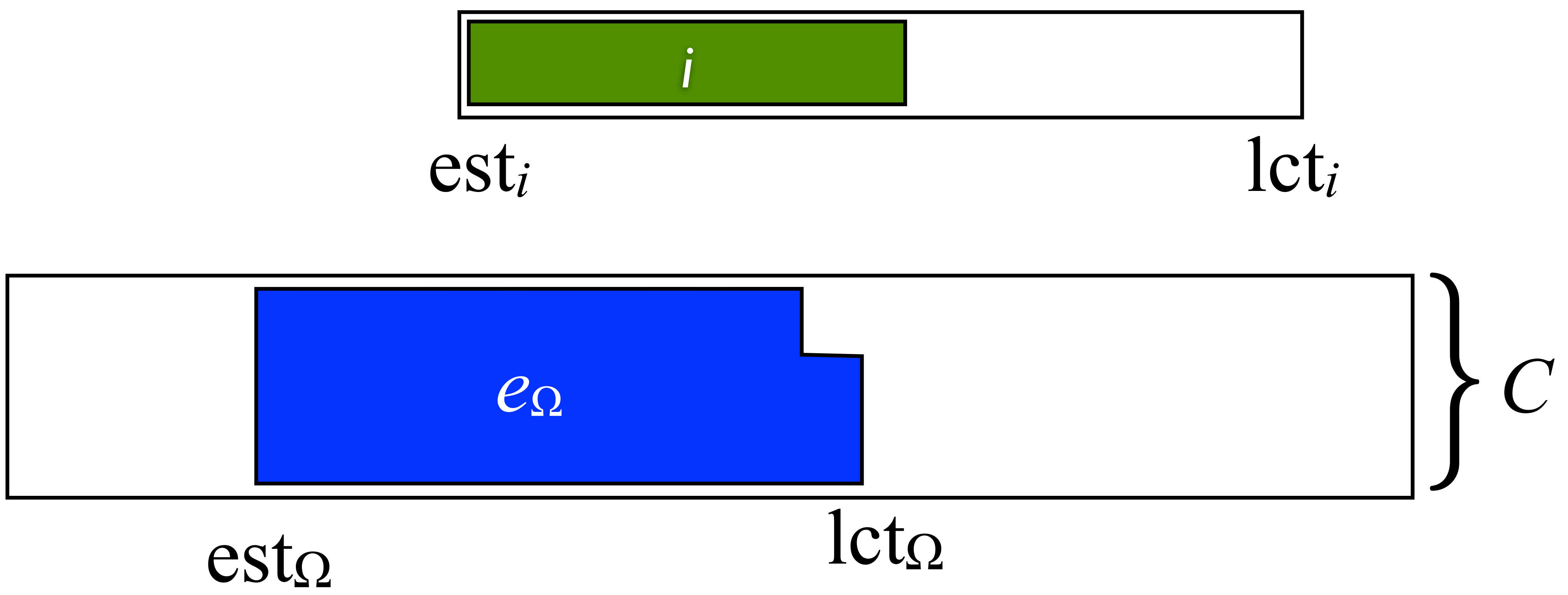
Energetic Relaxation



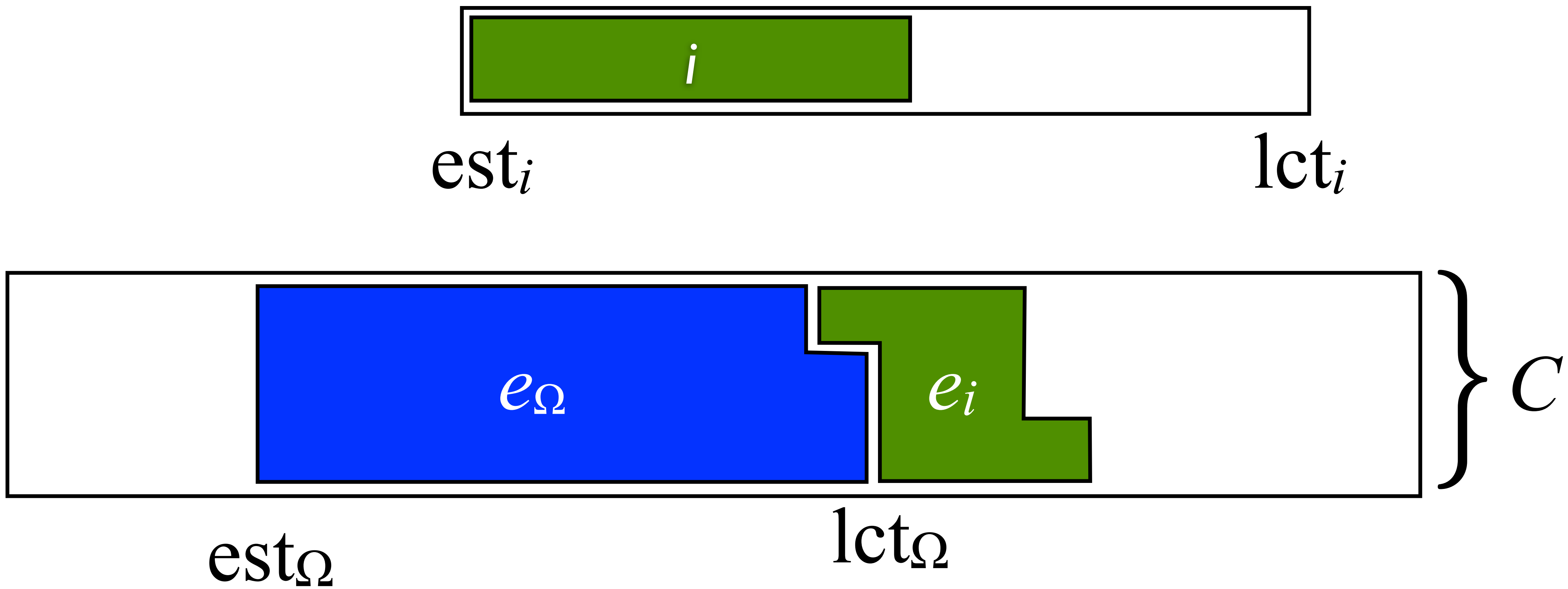
Edge Finder



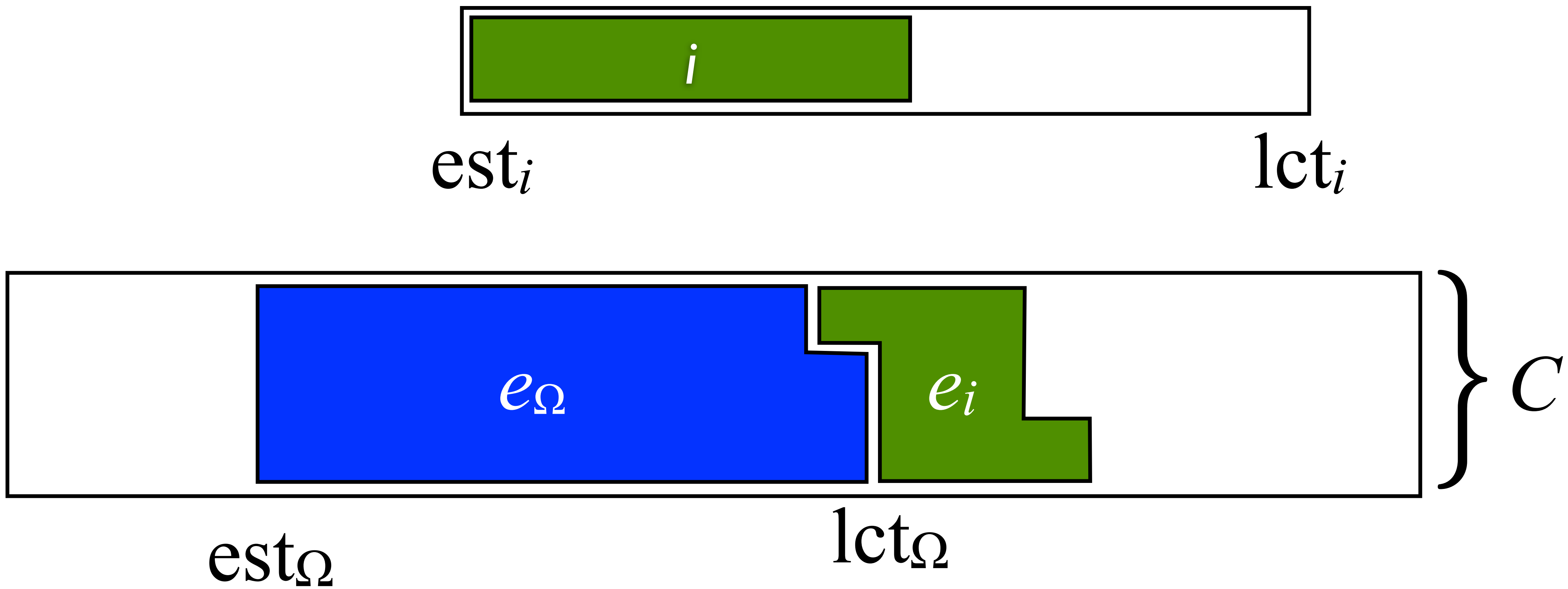
Edge Finder



Edge Finder

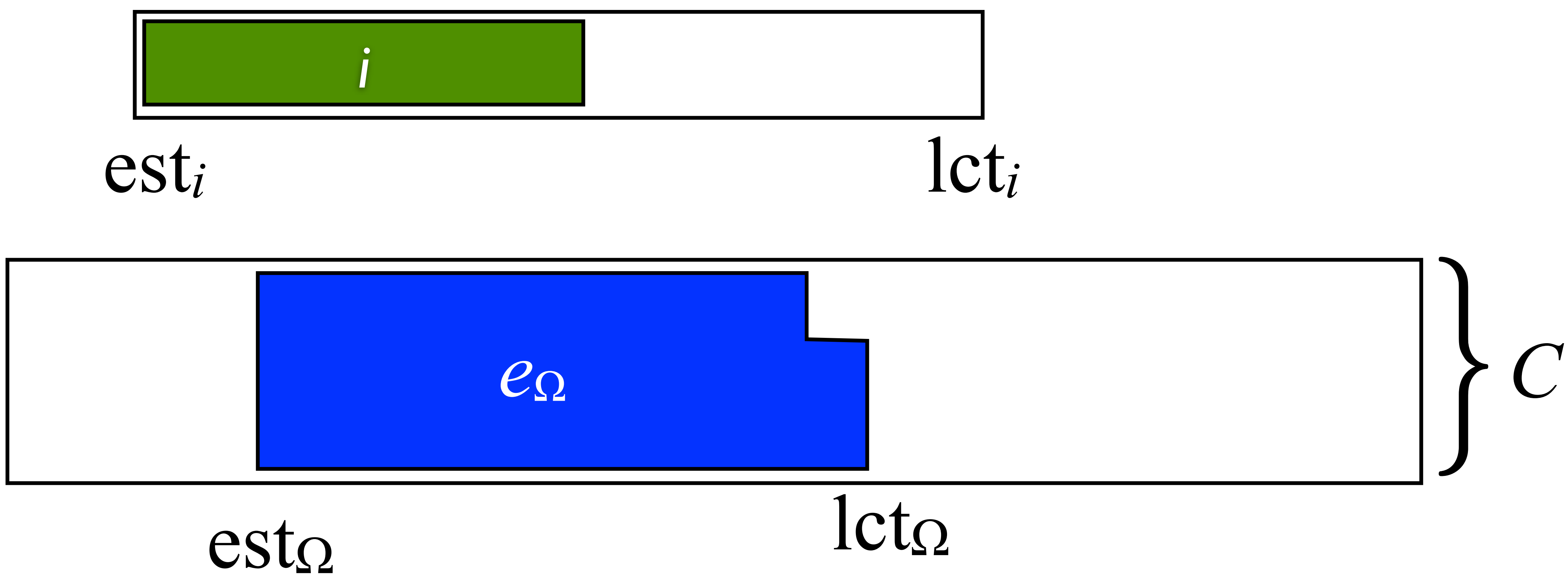


Edge Finder

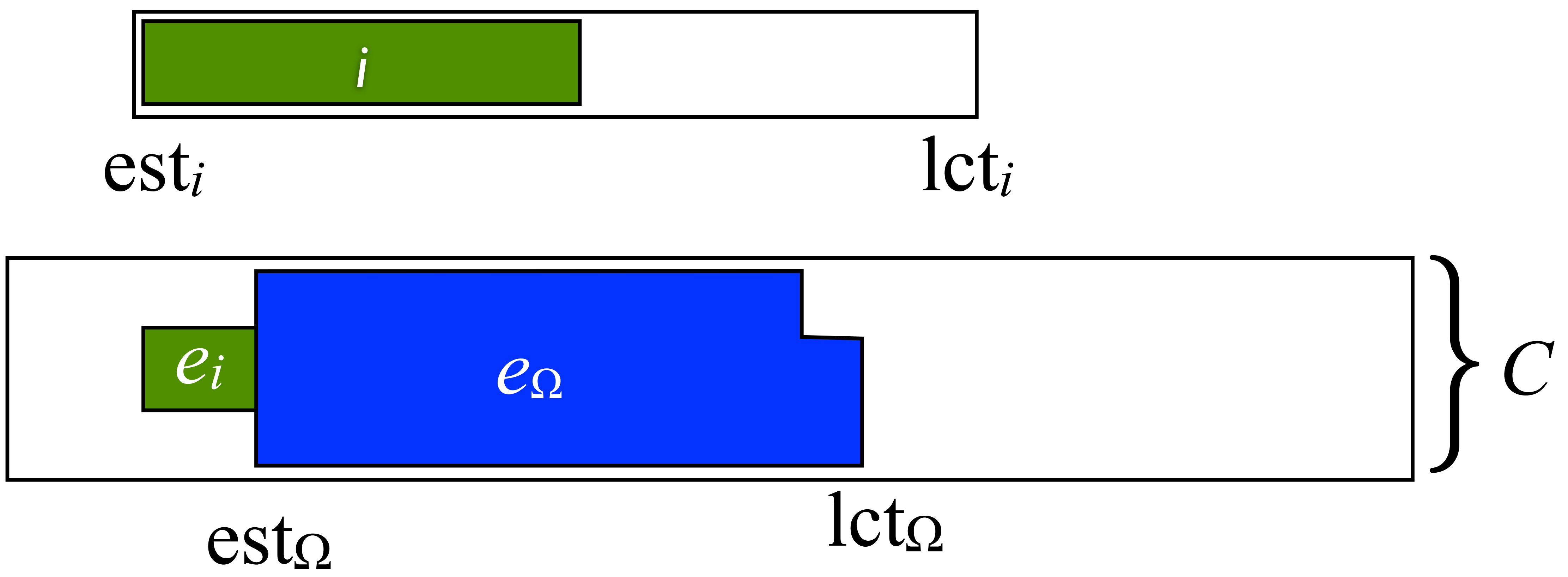


Ω precedes i

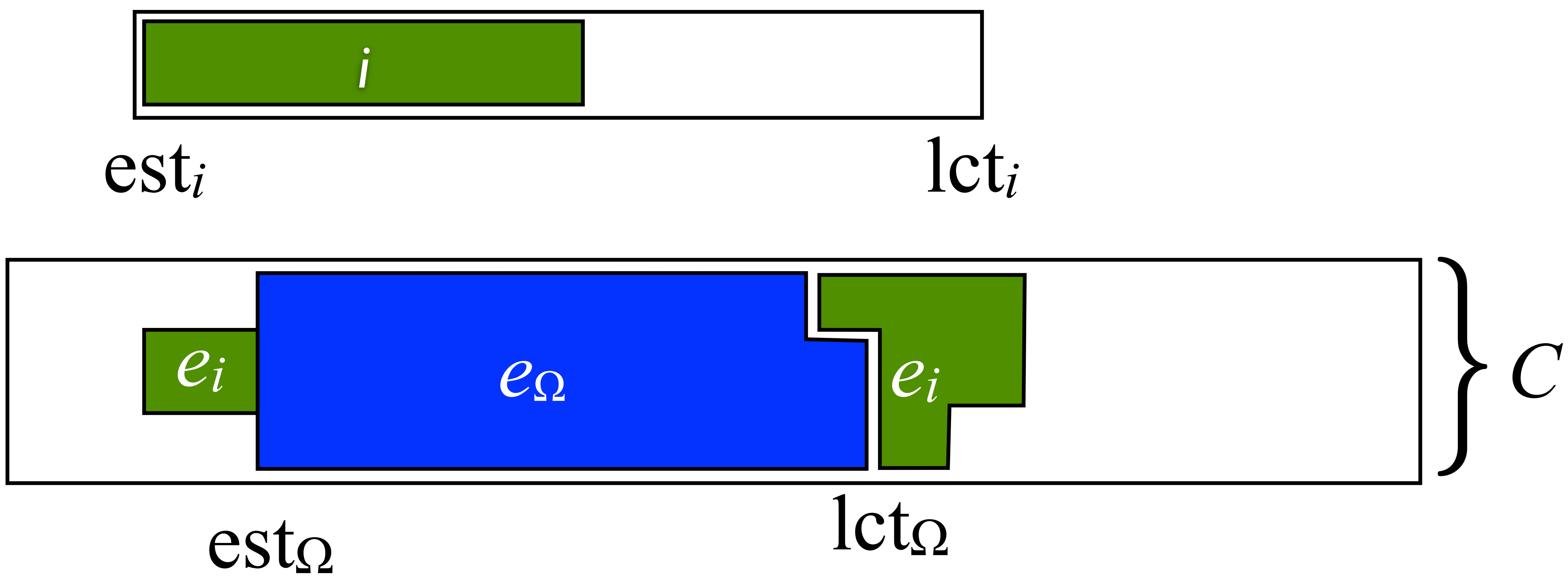
Extended-Edge Finder



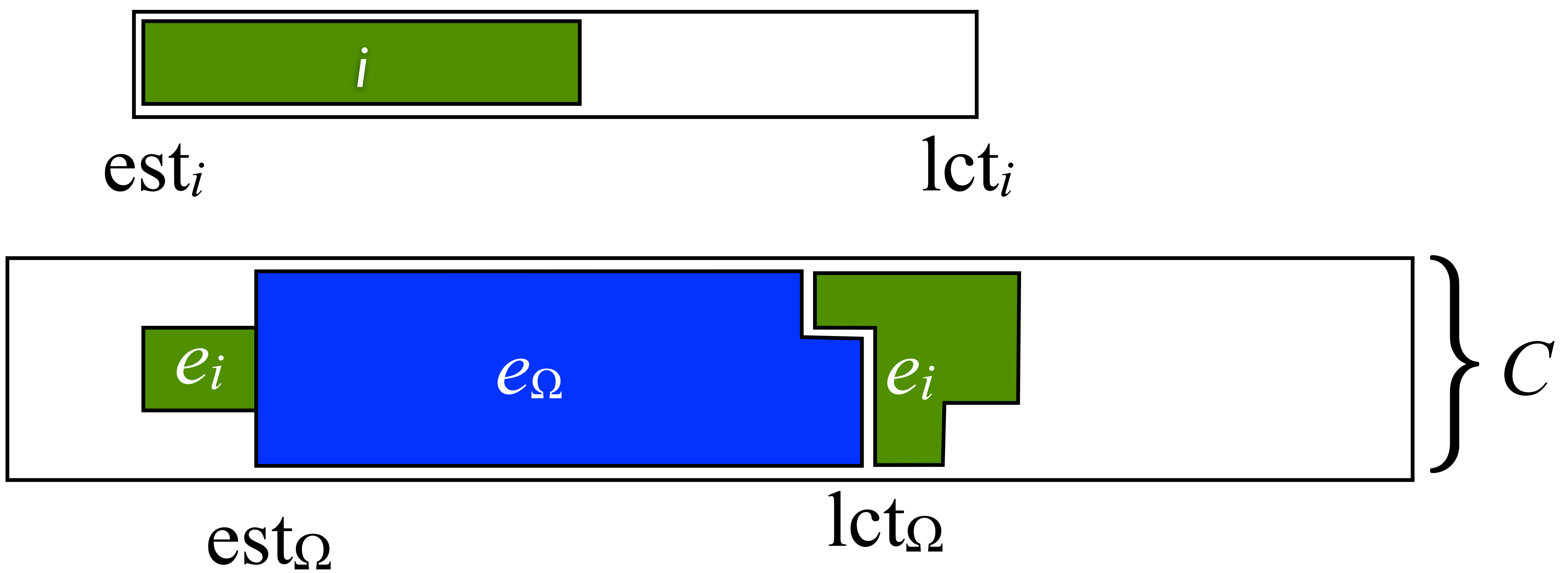
Extended-Edge Finder



Extended-Edge Finder



Extended-Edge Finder



Ω precedes i

Envelop

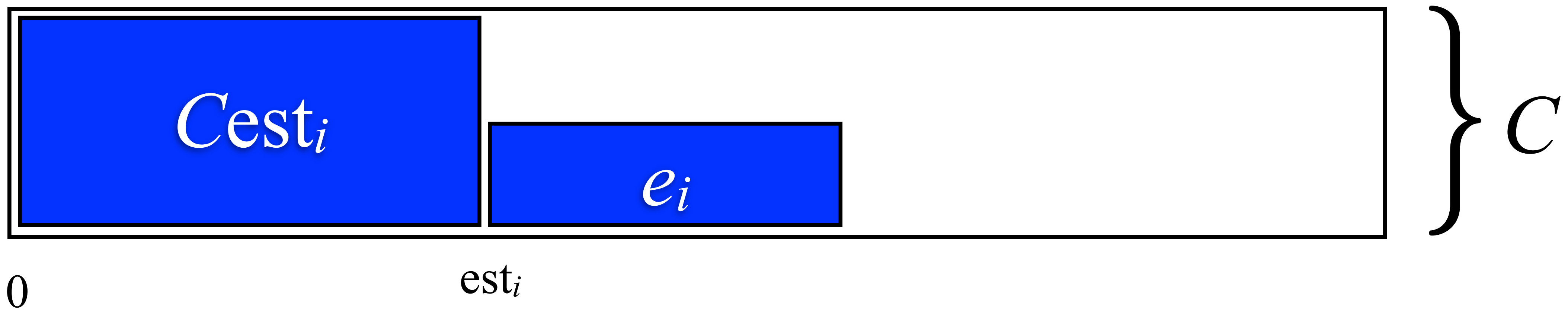
[Vilím CP 2009]

$$Env(i) = C\mathrm{est}_i + e_i$$

Envelop

[Vilím CP 2009]

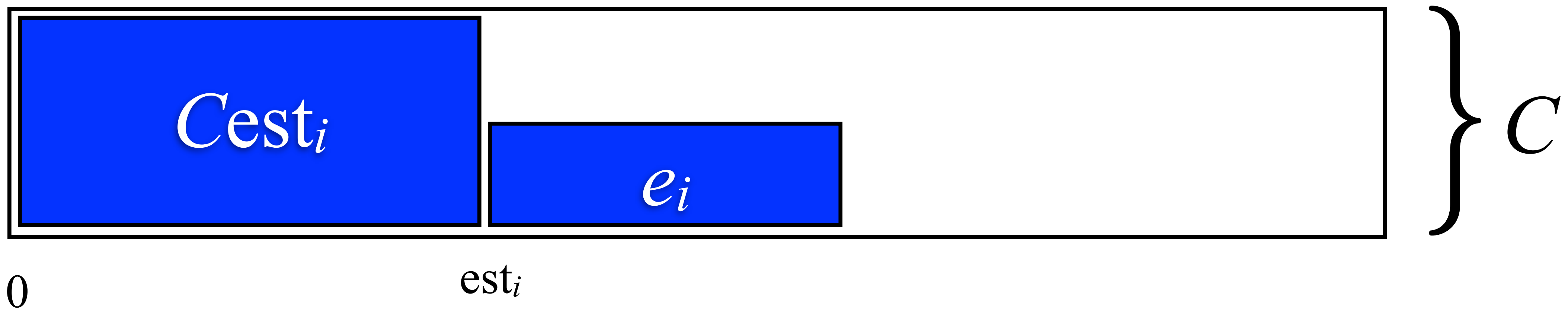
$$Env(i) = C\text{est}_i + e_i$$



Envelop

[Vilím CP 2009]

$$Env(i) = C\text{est}_i + e_i$$

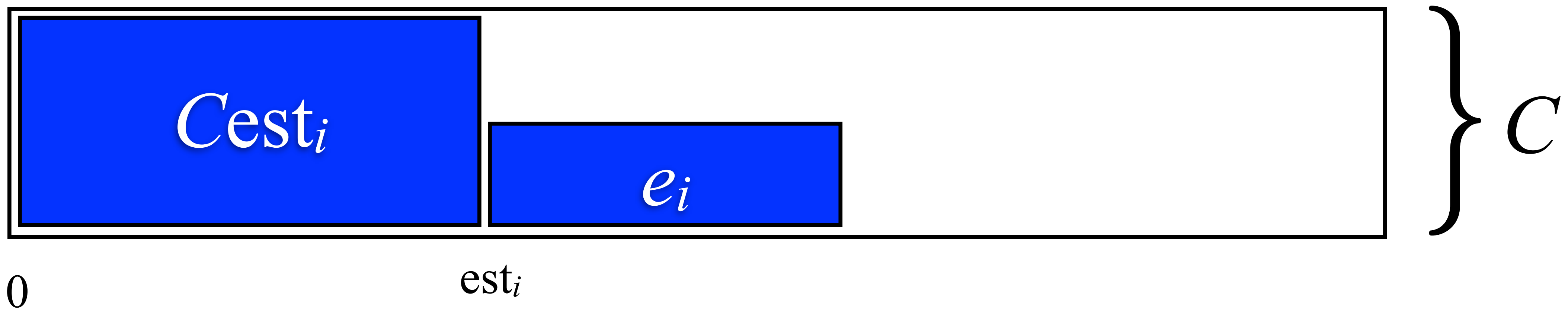


$$Env(\Omega) = \max_{\Theta \subseteq \Omega} C\text{est}_\Theta + e_\Theta$$

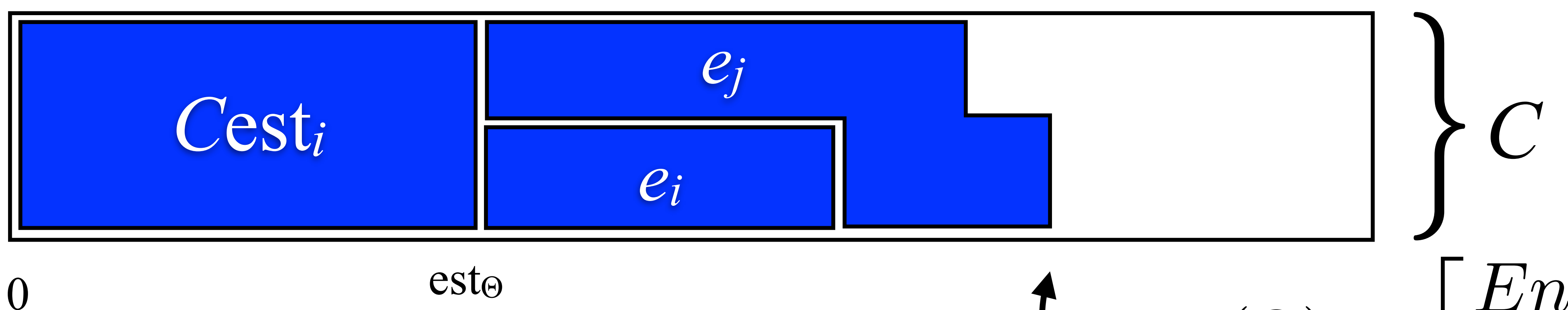
Envelop

[Vilím CP 2009]

$$Env(i) = C\text{est}_i + e_i$$



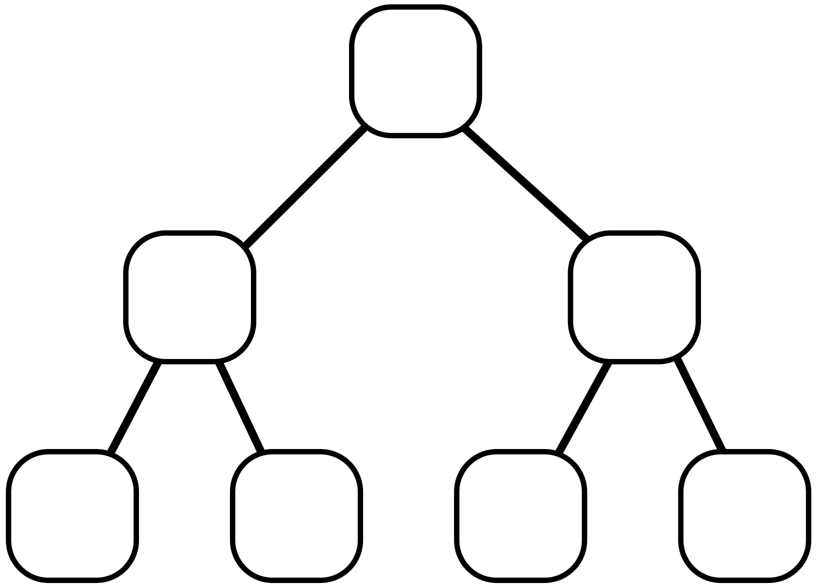
$$Env(\Omega) = \max_{\Theta \subseteq \Omega} C\text{est}_\Theta + e_\Theta$$



$$\text{ect}(\Omega) = \left\lceil \frac{Env(\Omega)}{C} \right\rceil$$

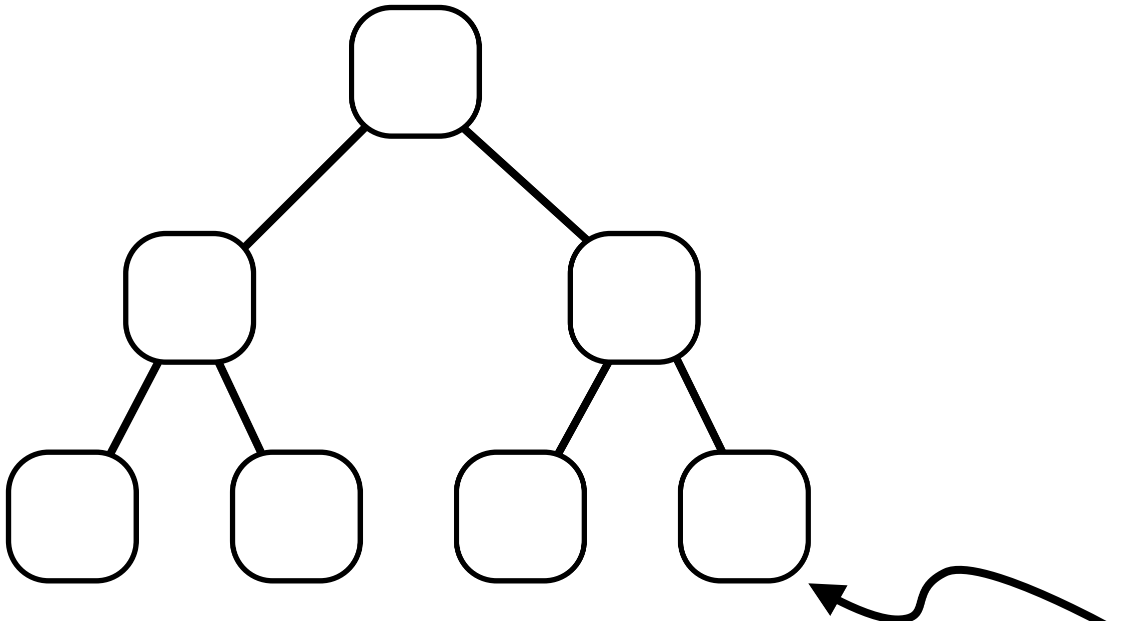
Cumulative Tree

[Vilím CP 2009]



Cumulative Tree

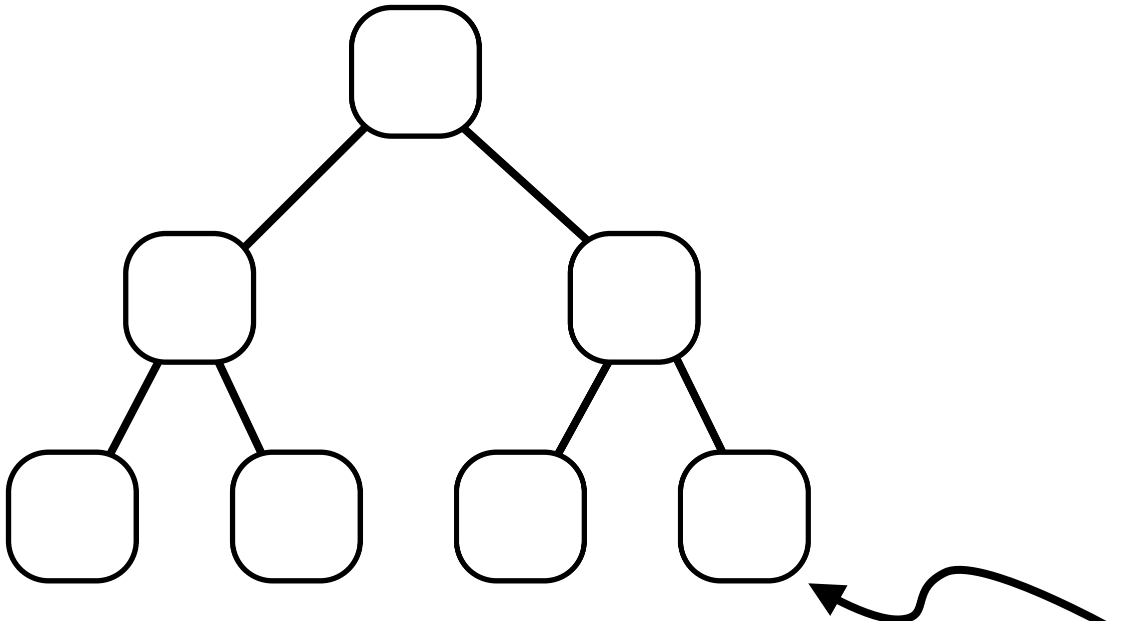
[Vilím CP 2009]



Claude-Guy Quimper

Cumulative Tree

[Vilím CP 2009]

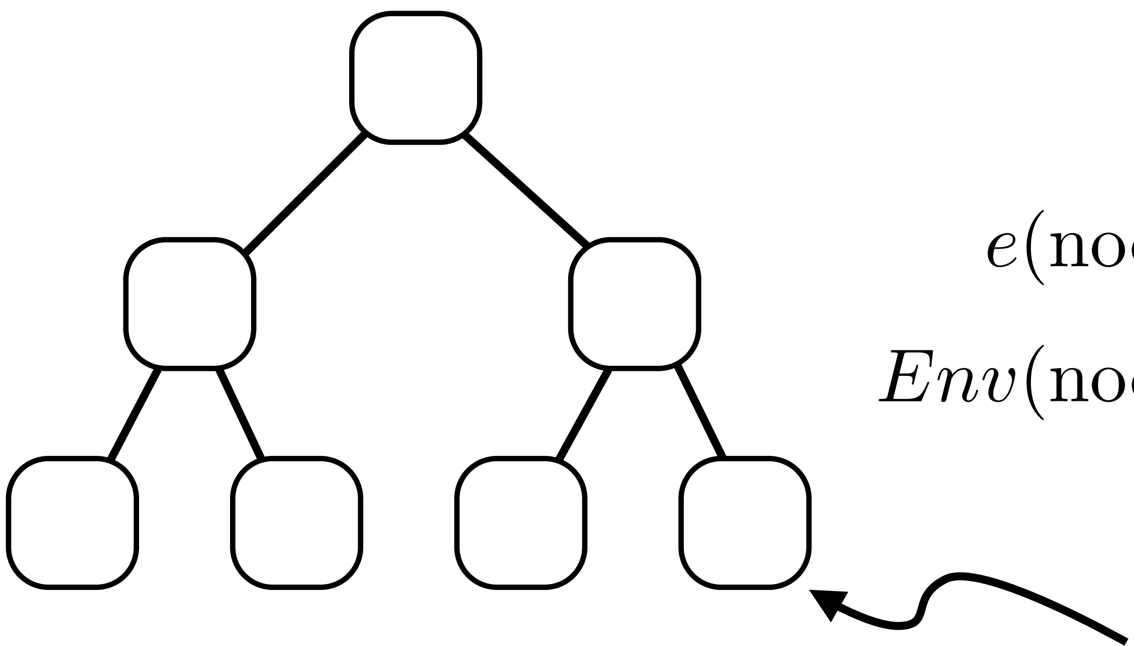


$$e(\text{leaf}) = \quad e_i \quad = p_i h_i$$
$$Env(\text{leaf}) = Env(i) = \; C\text{est}_i + e_i$$

One leaf per task

Cumulative Tree

[Vilím CP 2009]



$$e(\text{node}) = e_{\text{left}} + e_{\text{right}}$$

$$Env(\text{node}) = \max\left(Env(\text{left}) + e_{\text{right}}, Env(\text{right})\right)$$

$$e(\text{leaf}) = e_i = p_i h_i$$

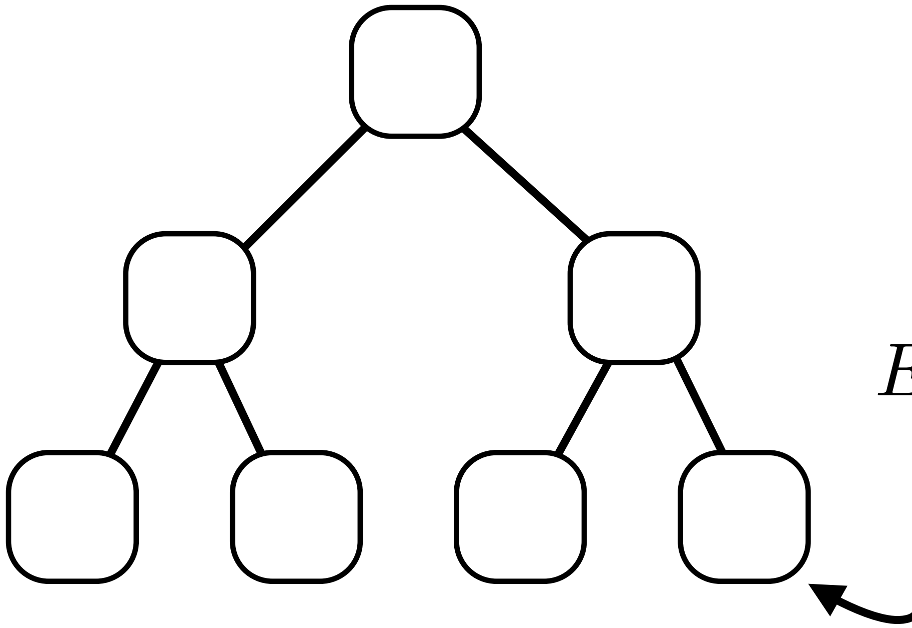
$$Env(\text{leaf}) = Env(i) = C\text{est}_i + e_i$$

One leaf per task

Cumulative Tree

[Vilím CP 2009]

$$Env(\text{root}) = Env(\Omega)$$



$$e(\text{node}) = e_{\text{left}} + e_{\text{right}}$$

$$Env(\text{node}) = \max\left(Env(\text{left}) + e_{\text{right}}, Env(\text{right})\right)$$

$$e(\text{leaf}) = e_i = p_i h_i$$

$$Env(\text{leaf}) = Env(i) = C\text{est}_i + e_i$$

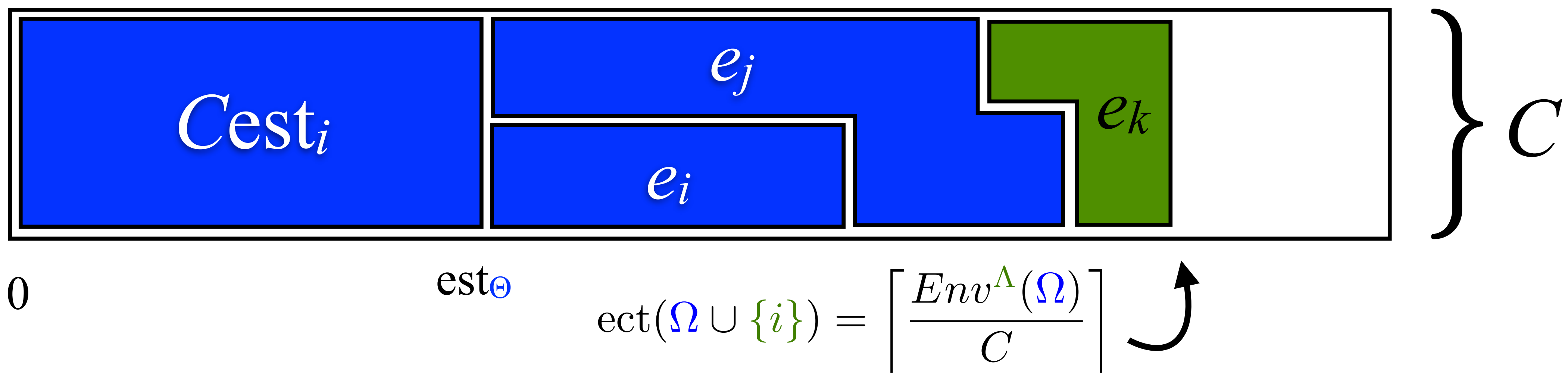
One leaf per task

Lambda Envelope

[Vilím CP 2009]

- Ω is the set of tasks whose lct is before t.
- Λ is the set of tasks whose lct is after t.
- This envelope computes the earliest completion time of all tasks in Ω with one task in Λ.

$$Env^{\Lambda}(\Omega) = \max_{\Theta \subseteq \Omega} \max_{\substack{i \in \Lambda \\ \text{est}_{\Theta} \leq \text{est}_i}} C\text{est}_{\Theta} + e_{\Theta} + e_i$$

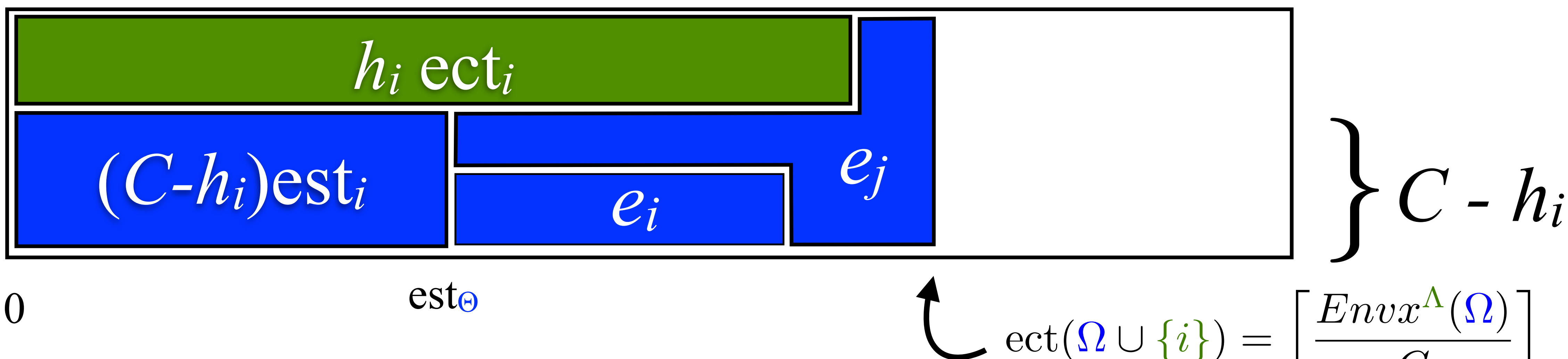


- The cumulative tree can also compute that envelope.

(half) Extended-Edge-Finder

- Ω is the set of tasks whose lct is before t.
- Λ is the set of tasks whose lct is after t **and ect is before t**.

$$Envx^{\Lambda}(\Omega) = \max_{\Theta \subseteq \Omega} \max_{\substack{i \in \Lambda \\ \mathrm{est}_i < \mathrm{est}_\Theta}} (C - h_i)\,\mathrm{est}_\Theta + e_\Theta + h_i\,\mathrm{ect}_i$$



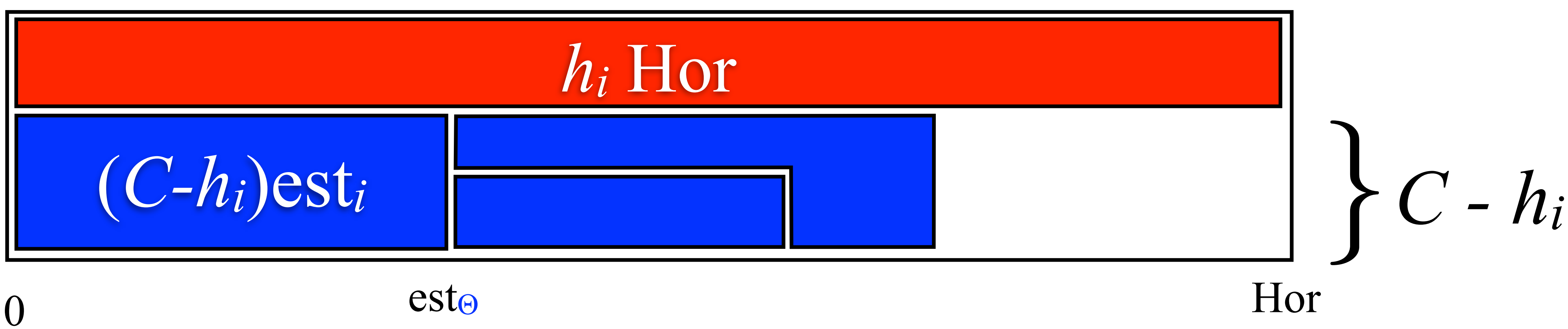
$$\mathrm{ect}(\Omega \cup \{i\}) = \left\lceil \frac{Envx^{\Lambda}(\Omega)}{C} \right\rceil$$

- If ect($\Omega \cup \{i\}$) > t then Ω precedes i.
- This new envelope can be computed with a cumulative tree.

(other half) Extended-Edge-Finder

- Ω is the set of tasks whose lct is before t.
- Ψ is the set of tasks whose lct is after t **and ect is after t**.

$$Envx^{\Psi}(\Omega) = \max_{\Theta \subseteq \Omega} \max_{\substack{i \in \Psi \\ \mathrm{est}_i < \mathrm{est}_\Theta}} (C - h_i)\,\mathrm{est}_\Theta + e_\Theta + h_i Hor$$



- If $\mathrm{Envx}^{\Psi} > Ct + h(Hor - t)$ then Ω precedes i.
- This new envelope can be computed with a cumulative tree.

Extended-Edge-Finder

- For every distinct task height *h*
  - Initialize the cumulative tree with all tasks in Ω and empty sets Λ and Ψ.
  - For latest completion times *t in* decreasing order
    - Move from Ω to Λ the tasks with height *h* whose latest completion time is later then *t*.
    - Move from Λ to Ψ the tasks whose earliest completion time is later than *t*.
    - Update the cumulative tree.
    - If an envelope detects a precedence, proceed to the adjustment and remove from Λ or Ψ the filtered task.

Extended-Edge-Finder

- For every distinct task height *h* } We suppose *k* distinct heights.
  - Initialize the cumulative tree with all tasks in Ω and empty sets Λ and Ψ.
  - For latest completion times *t in* decreasing order
    - Move from Ω to Λ the tasks with height *h* whose latest completion time is later then *t*.
    - Move from Λ to Ψ the tasks whose earliest completion time is later than *t*.
    - Update the cumulative tree.
    - If an envelope detects a precedence, proceed to the adjustment and remove from Λ or Ψ the filtered task.

Extended-Edge-Finder

- For every distinct task height *h*
  - Initialize the cumulative tree with all tasks in Ω and empty sets Λ and Ψ.
  - For latest completion times *t in* decreasing order
    - Move from Ω to Λ the tasks with height *h* whose latest completion time is later then *t*.
    - Move from Λ to Ψ the tasks whose earliest completion time is later than *t*.
    - Update the cumulative tree.
    - If an envelope detects a precedence, proceed to the adjustment and remove from Λ or Ψ the filtered task.

} We suppose *k* distinct heights.

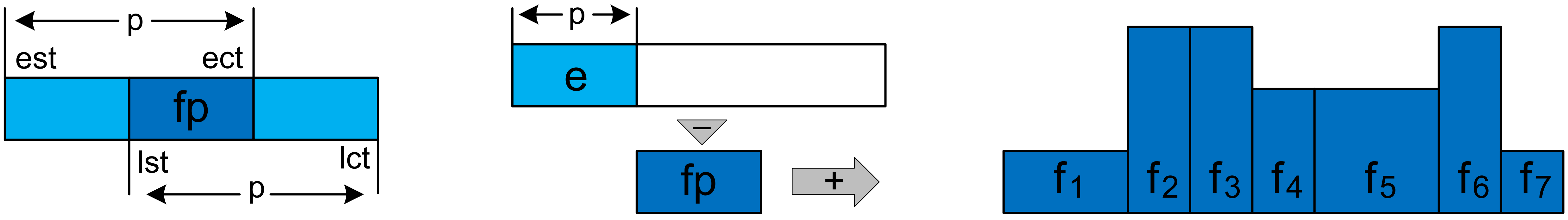
} Each of these 2*n* moves require a O(log *n*) update of the tree.

Extended-Edge-Finder

- For every distinct task height *h*
  - Initialize the cumulative tree with all tasks in Ω and empty sets Λ and Ψ.
  - For latest completion times *t in* decreasing order
    - Move from Ω to Λ the tasks with height *h* whose latest completion time is later then *t*.
    - Move from Λ to Ψ the tasks whose earliest completion time is later than *t*.
    - Update the cumulative tree.
    - If an envelope detects a precedence, proceed to the adjustment and remove from Λ or Ψ the filtered task.

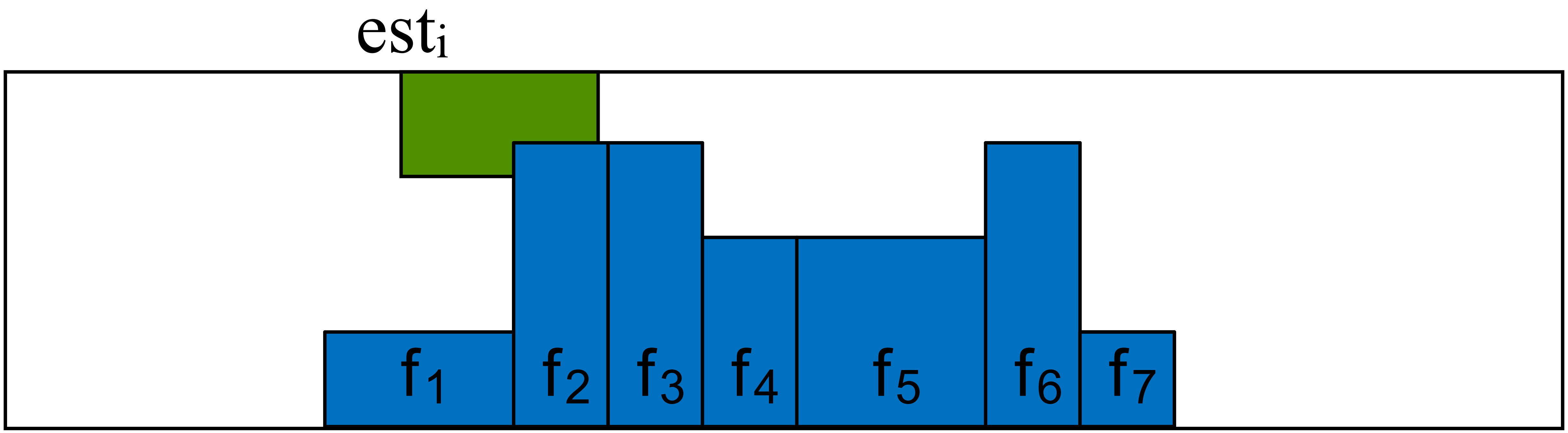
$O(k\ n \log n)$

Time-Table



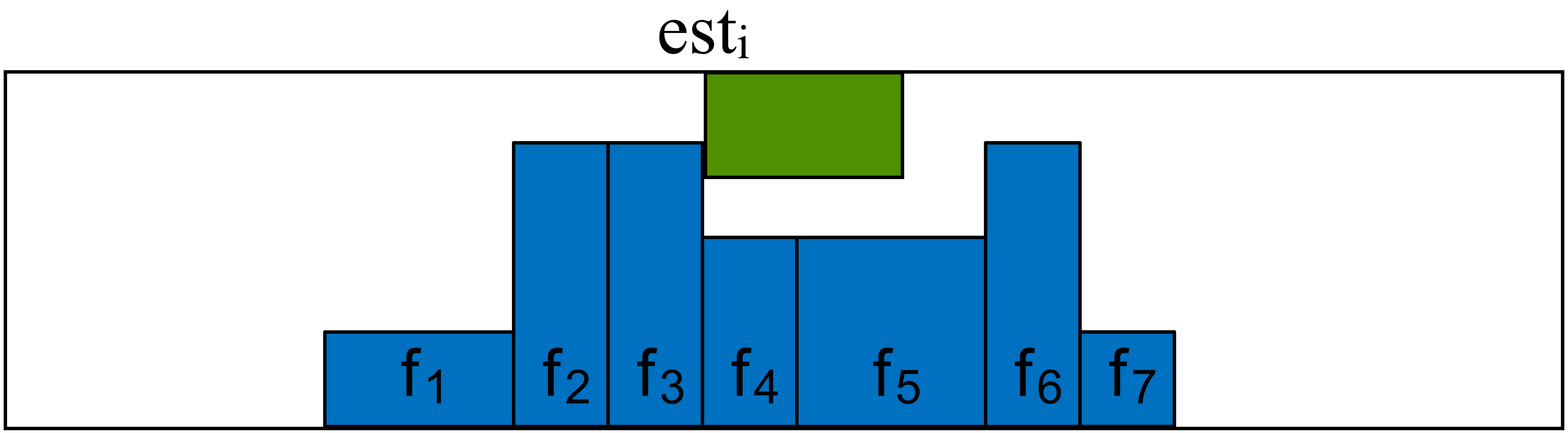
- We present an algorithm that runs in $O(n \log n)$.
  - It decomposes the tasks into fixed and depleted parts.
  - It aggregates the fixed parts into at most n fixed tasks whose domains are disjoint.

Time-Table



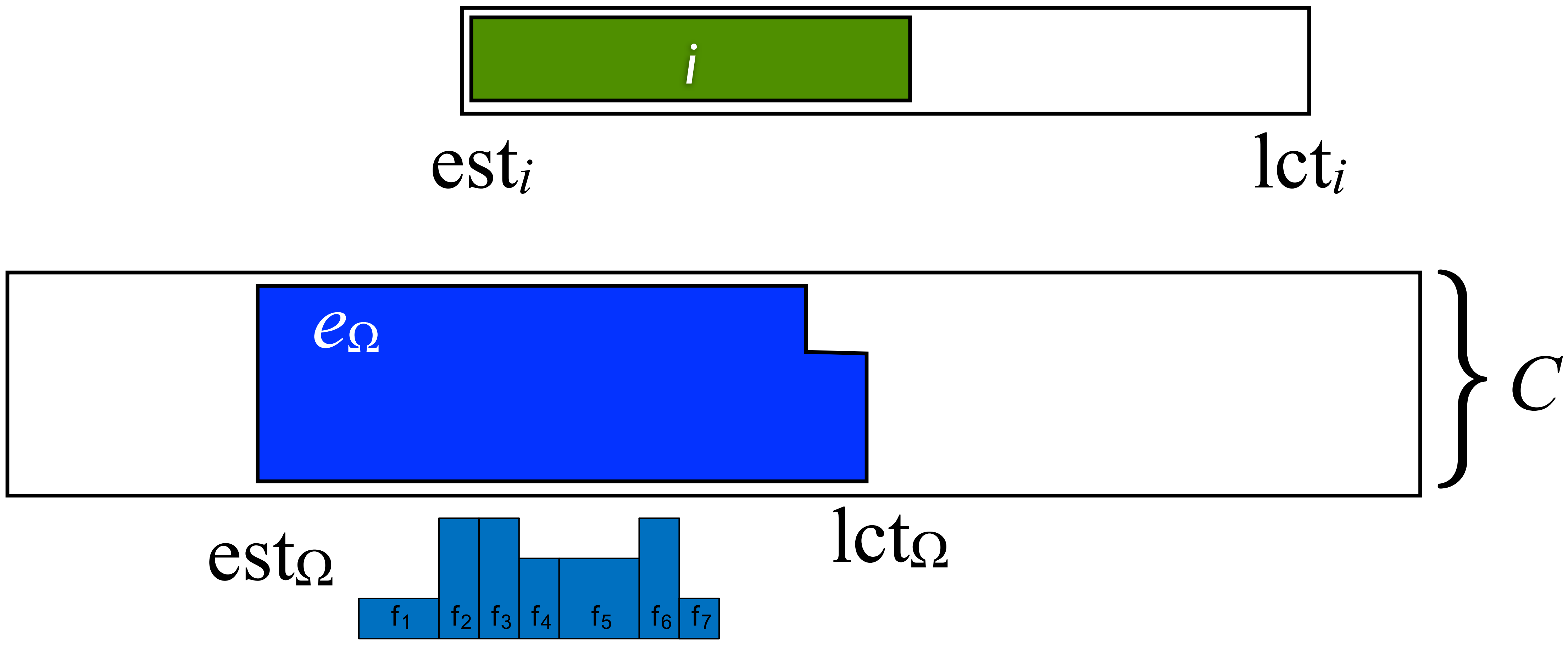
- The algorithm also prunes the earliest starting times in O(n log n).

Time-Table

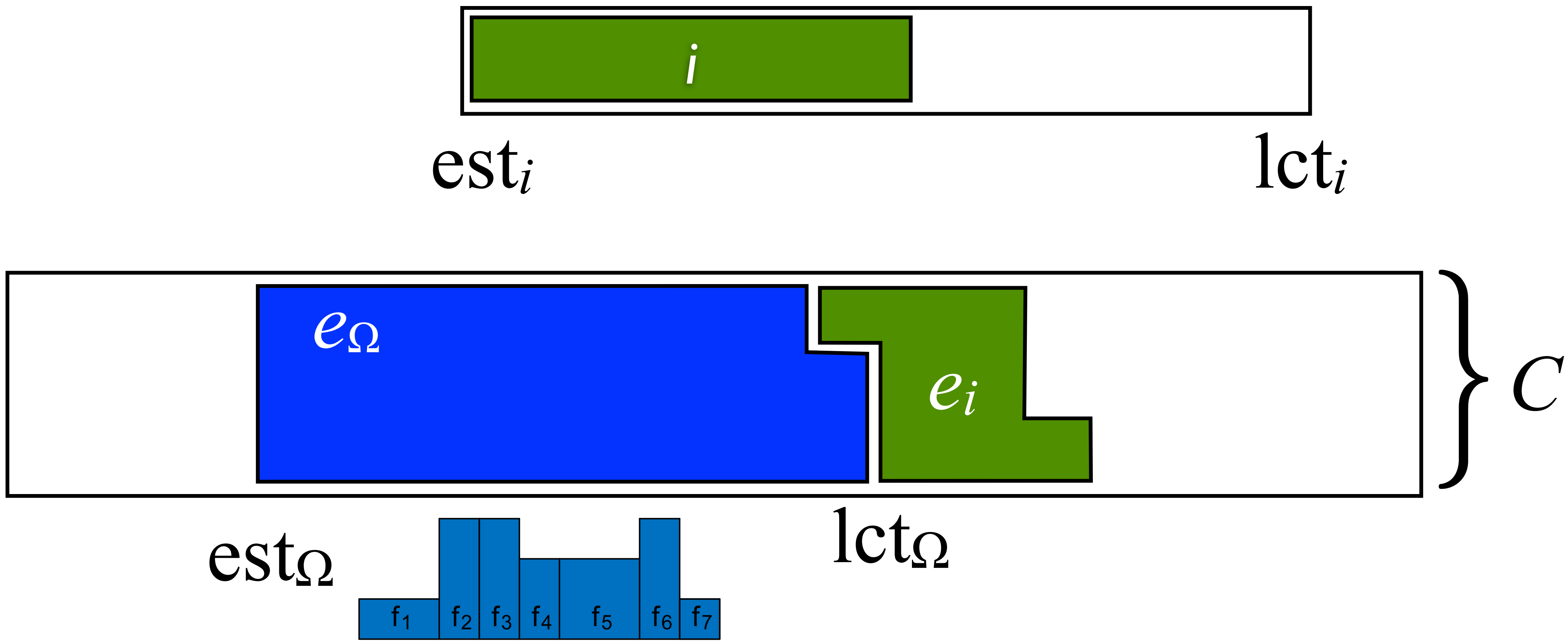


- The algorithm also prunes the earliest starting times in O(n log n).

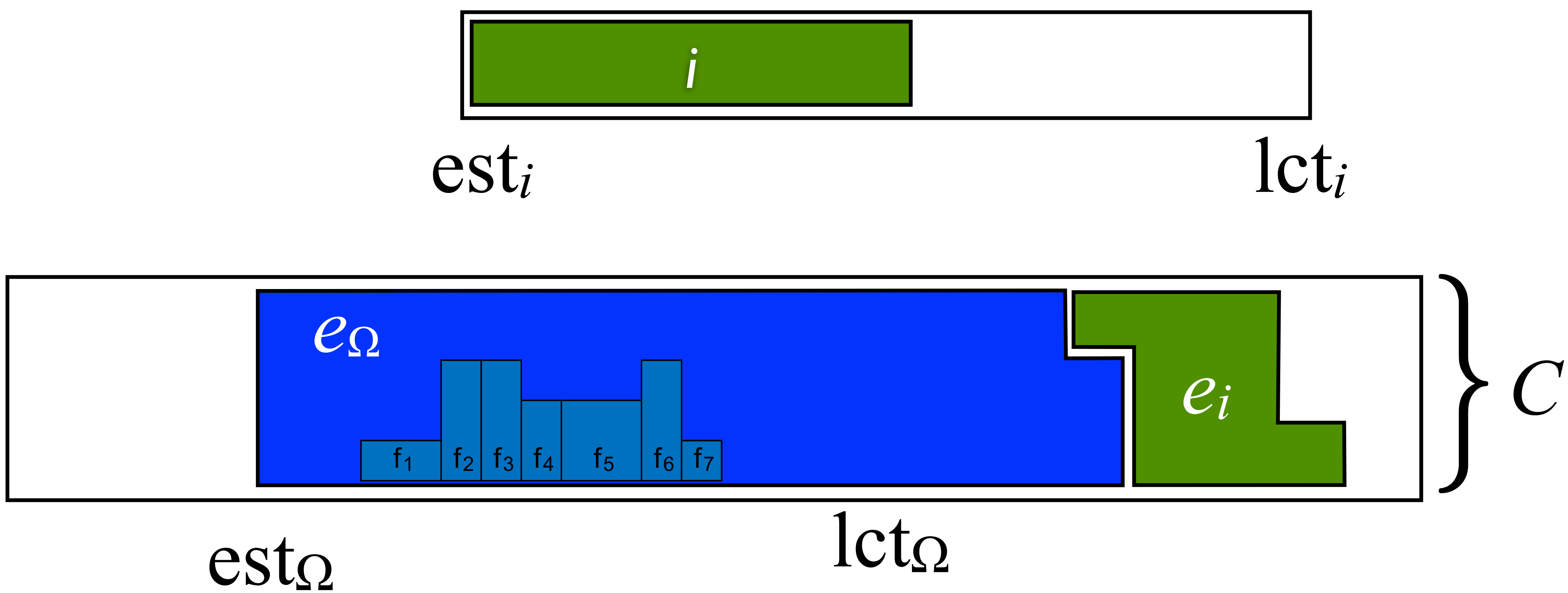
Time-Table Extended-Edge-Finding



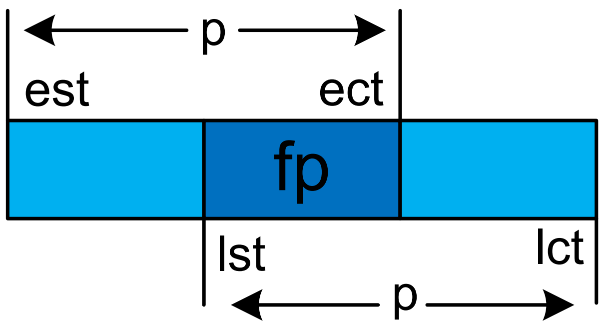
Time-Table Extended-Edge-Finding

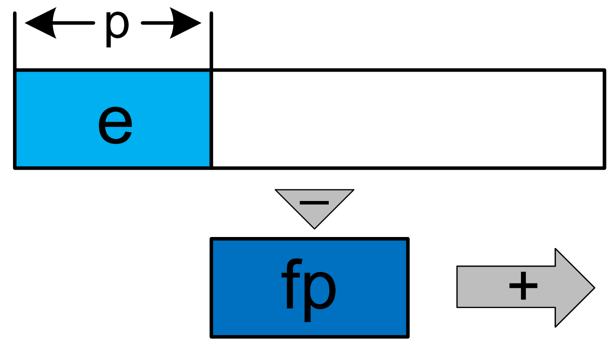


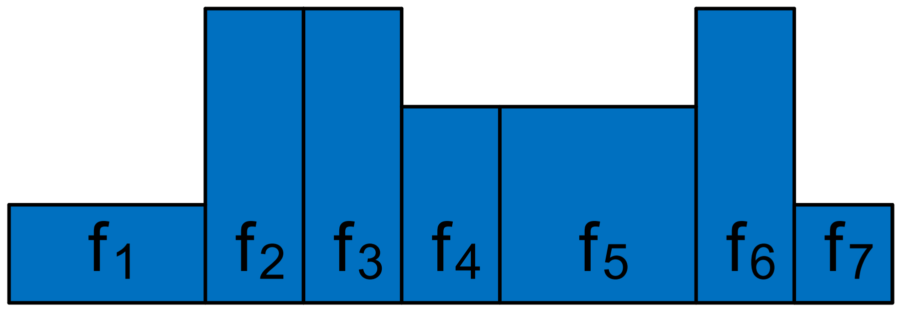
Time-Table Extended-Edge-Finding



Algorithm







- Decompose the problem into fixed and depleted tasks.
- Run the Extended-Edge Finder on the decomposition.
- Analyze the filtering and apply the filtering to the original tasks.
- Complexity: O(k n log n)

Experiments

- We used Choco 2.1.5 on the PspLib benchmark.

Benchmark			Choco			EEF+TT			TTEEF		
n	#instances	time out	solved	bt	time	solved	bt	time	solved	bt	time
30	480	10	364	8757	223	377	8757	50	377	8379	54
60	480	20	332	3074	1527	340	3074	269	341	2861	291
90	480	50	321	5024	5522	327	5024	857	329	4635	913

- Using Extended-Edge-Finding and Time-Tabling produce the same number of backtracks for the 3 x 480 instances.
- Computation times are cut in 6.
- TTEEF did not perform significantly better than EEF+TT.

Conclusion

- We proposed:
  - an Extended-Edge-Finder that runs in O(k n log n).
  - a Time-Tabling algorithm that runs in O(n log n).
  - A Time-Table-Extended-Edge-Finding that runs in O(k n log n).