Optimal Algorithm for Online Multiple Knapsack



Marcin Bieńkowski

Maciej Pacut
(speaker)

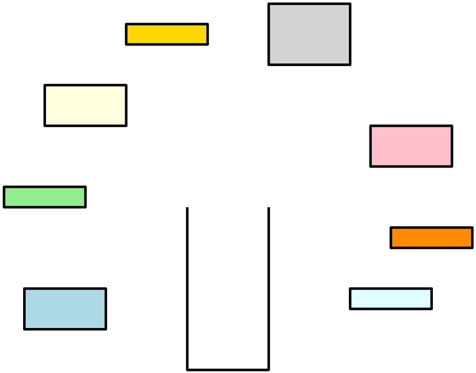
Krzysztof Piecuch

Uniwersytet Wrocławski

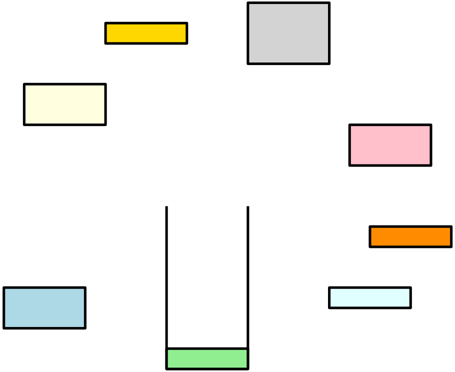
universität wien

Uniwersytet Wrocławski

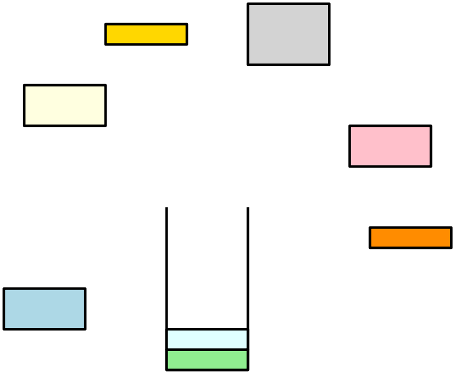
Knapsack



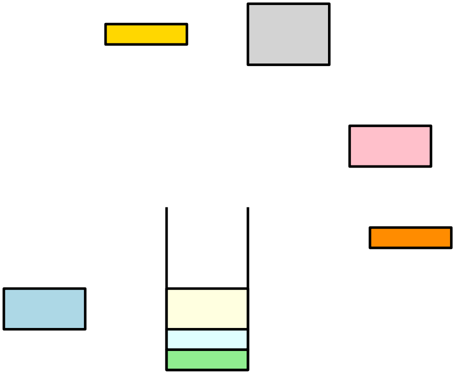
Knapsack



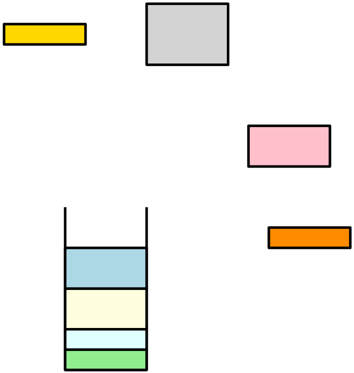
Knapsack



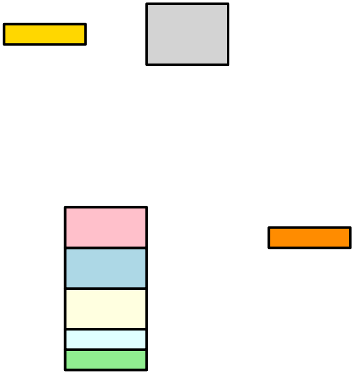
Knapsack



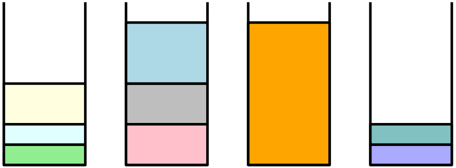
Knapsack



Knapsack



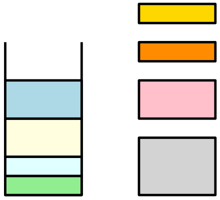
Multiple Knapsack



Textbook Knapsack (offline)

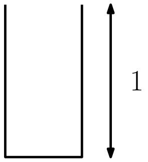
Given

- one knapsack of capacity 1
- multiset of items (size and weight)

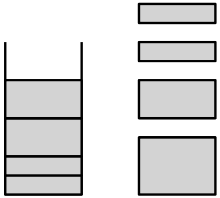


Choose a subset of items

- sum of sizes ≤ 1
- maximize total weight



Proportional Knapsack (offline)

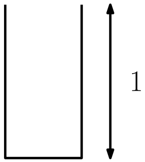


Given

- one knapsack of capacity 1
- multiset of items (size ~~and weight~~)

Choose a subset of items

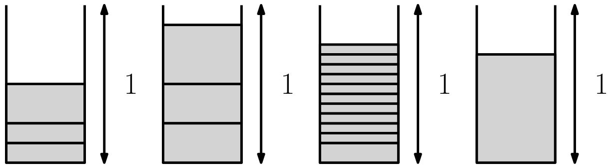
- sum of sizes ≤ 1
- maximize total ~~weight~~ size



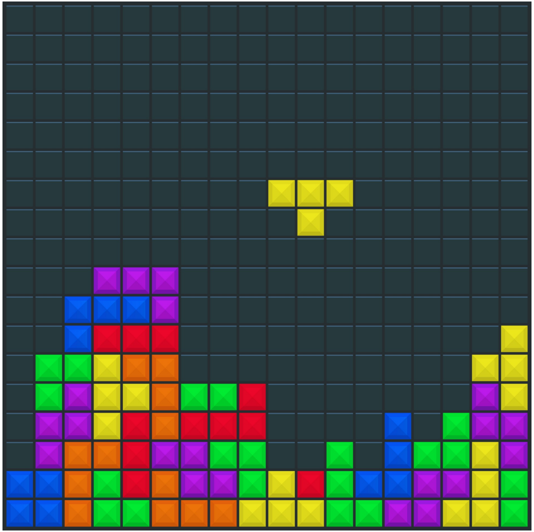
Multiple Knapsack (offline)

Choose a subset of items

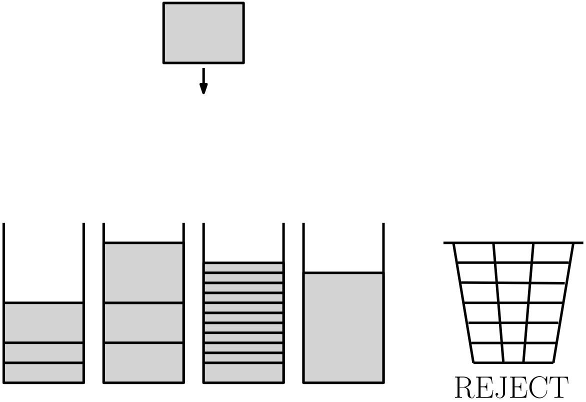
- assign accepted items to a knapsacks
- in each knapsack: total size of items ≤ 1
- maximize total size



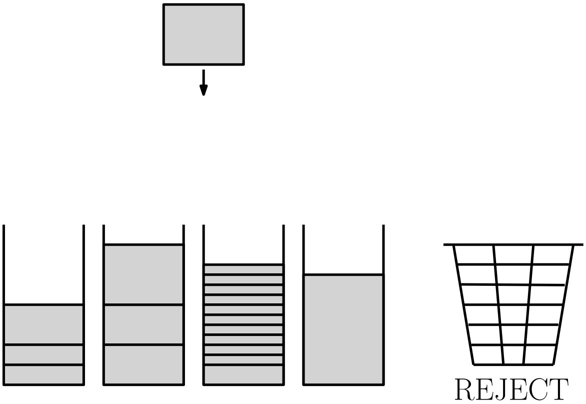
Online



Online Multiple Knapsack



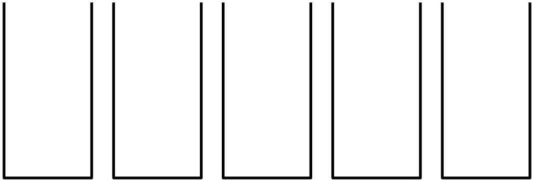
Online Multiple Knapsack



maximize $\frac{ALG_{online}}{OPT_{offline}}$

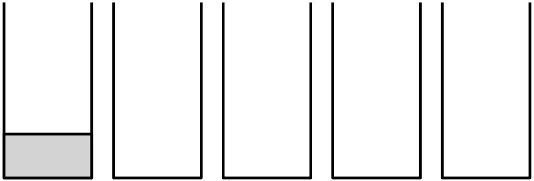
Known results

FirstFit is 0.5-competitive (Cygan et al. [TOCS 2016])



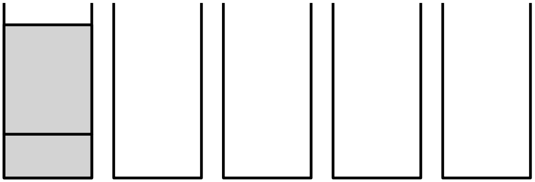
Known results

FirstFit is 0.5-competitive (Cygan et al. [TOCS 2016])



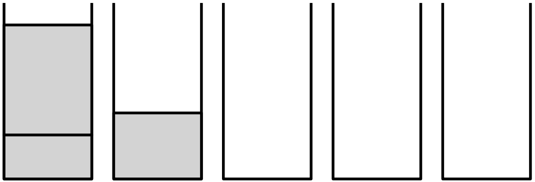
Known results

FirstFit is 0.5-competitive (Cygan et al. [TOCS 2016])



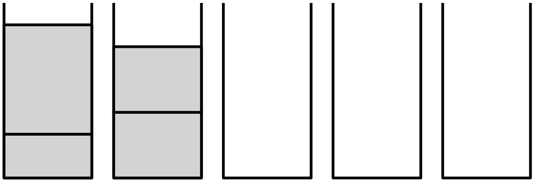
Known results

FirstFit is 0.5-competitive (Cygan et al. [TOCS 2016])



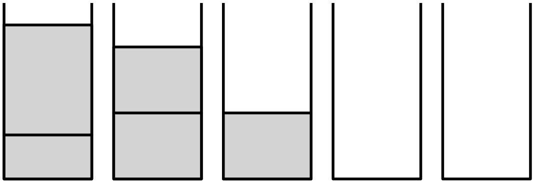
Known results

FirstFit is 0.5-competitive (Cygan et al. [TOCS 2016])



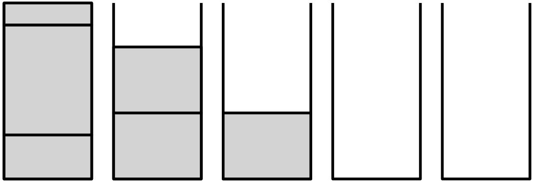
Known results

FirstFit is 0.5-competitive (Cygan et al. [TOCS 2016])



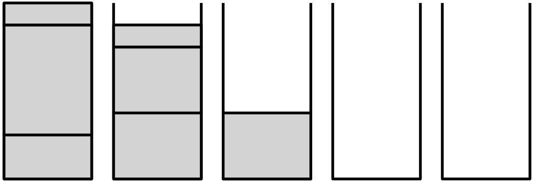
Known results

FirstFit is 0.5-competitive (Cygan et al. [TOCS 2016])



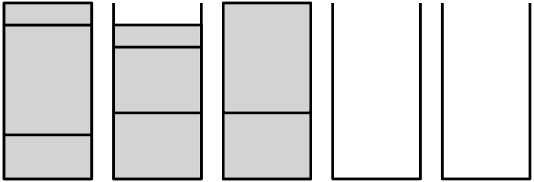
Known results

FirstFit is 0.5-competitive (Cygan et al. [TOCS 2016])



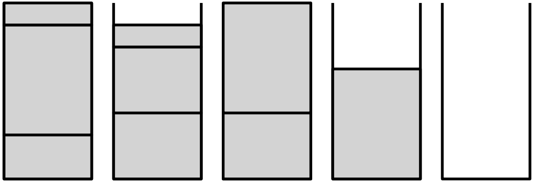
Known results

FirstFit is 0.5-competitive (Cygan et al. [TOCS 2016])



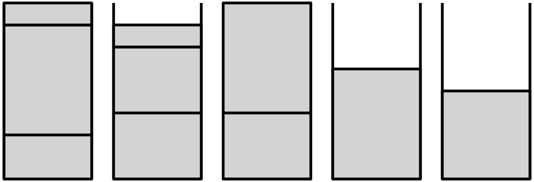
Known results

FirstFit is 0.5-competitive (Cygan et al. [TOCS 2016])



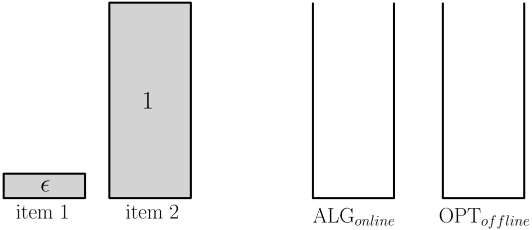
Known results

FirstFit is 0.5-competitive (Cygan et al. [TOCS 2016])



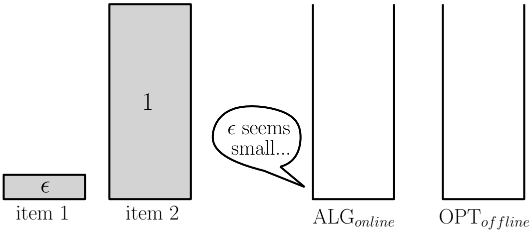
Known results

Bad news for One Online Knaspack



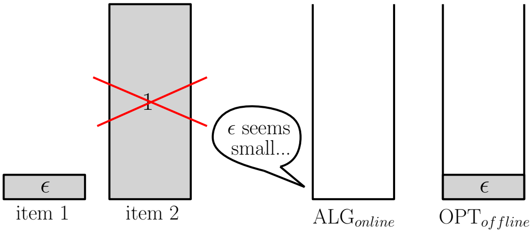
Known results

Bad news for One Online Knaspack



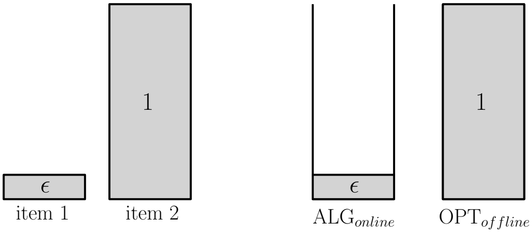
Known results

Bad news for One Online Knaspack



Known results

Bad news for One Online Knaspack



Known results

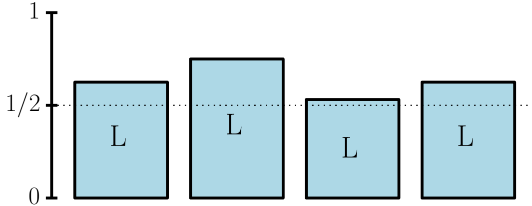
(max objective: higher is better)

Our contributions

(max objective: higher is better)

Rising Threshold Algorithm

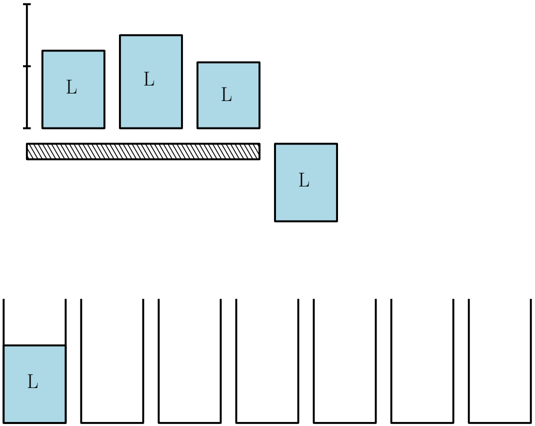
We say that items $(1/2, 1]$ are large



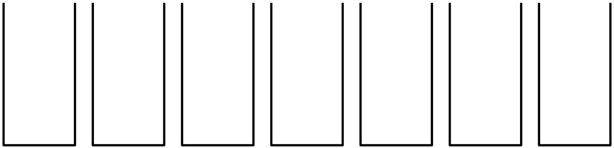
(max 1 large per knapsack)

Rising Threshold Algorithm

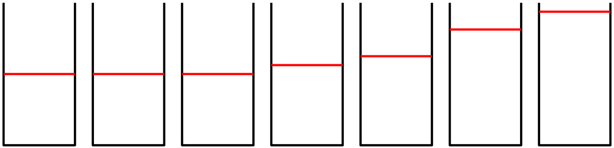
Step 1. Algorithm for large items



Rising Threshold Algorithm (for large items)

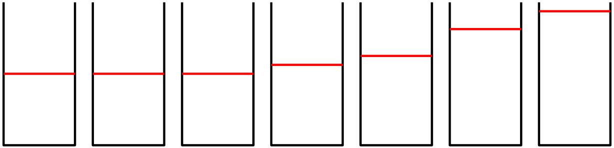


Rising Threshold Algorithm
(for large items)



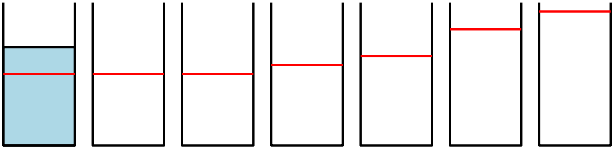
assign each knapsack a threshold (tbd)

Rising Threshold Algorithm (for large items)



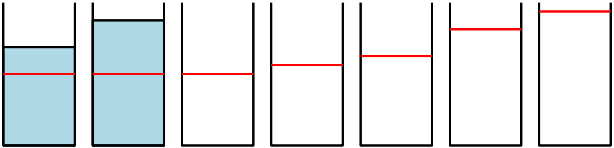
- fill from the left
- reject if under threshold

Rising Threshold Algorithm (for large items)



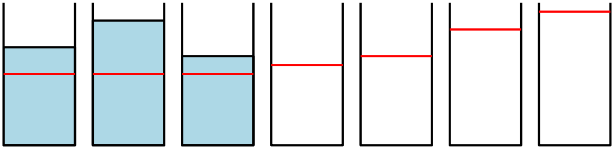
- fill from the left
- reject if under threshold

Rising Threshold Algorithm (for large items)



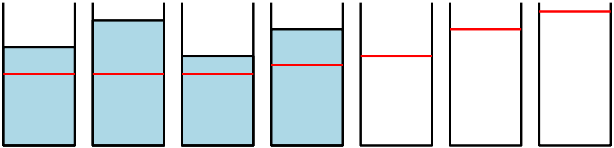
- fill from the left
- reject if under threshold

Rising Threshold Algorithm (for large items)



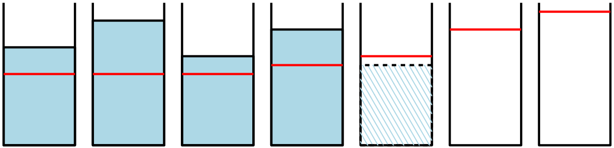
- fill from the left
- reject if under threshold

Rising Threshold Algorithm (for large items)



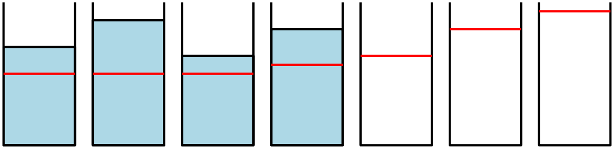
- fill from the left
- reject if under threshold

Rising Threshold Algorithm (for large items)



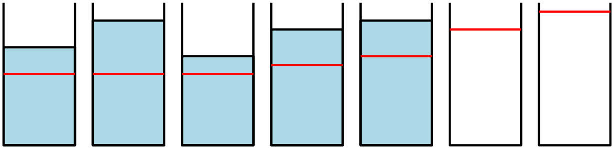
- fill from the left
- reject if under threshold

Rising Threshold Algorithm (for large items)



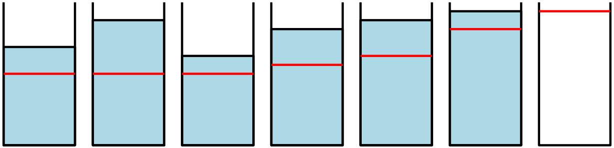
- fill from the left
- reject if under threshold

Rising Threshold Algorithm (for large items)



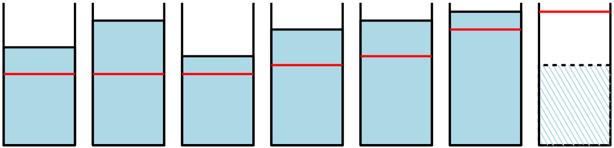
- fill from the left
- reject if under threshold

Rising Threshold Algorithm (for large items)



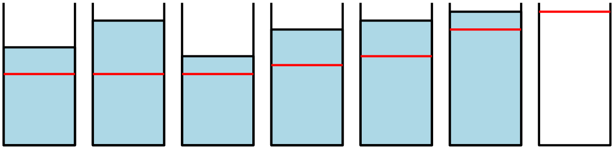
- fill from the left
- reject if under threshold

Rising Threshold Algorithm (for large items)



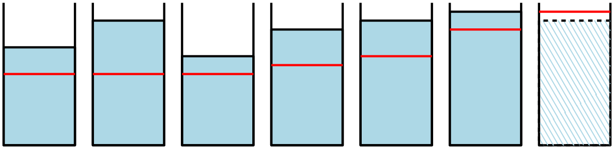
- fill from the left
- reject if under threshold

Rising Threshold Algorithm (for large items)



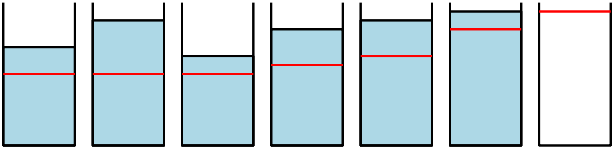
- fill from the left
- reject if under threshold

Rising Threshold Algorithm (for large items)



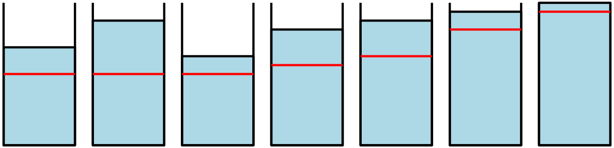
- fill from the left
- reject if under threshold

Rising Threshold Algorithm
(for large items)



- fill from the left
- reject if under threshold

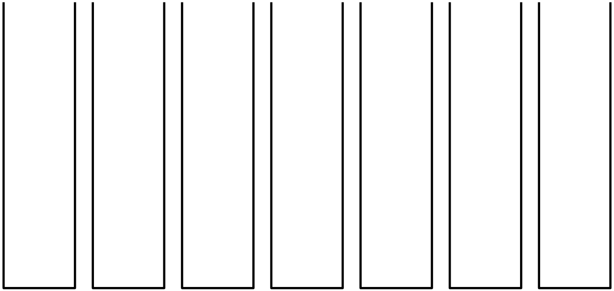
Rising Threshold Algorithm (for large items)



- fill from the left
- reject if under threshold

Rising Threshold Algorithm
Threshold function

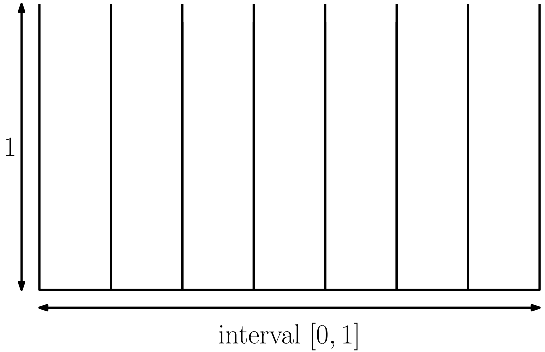
$$f(x) = \max\{1/2, (2e)^{x-1}\}$$



n knapsacks

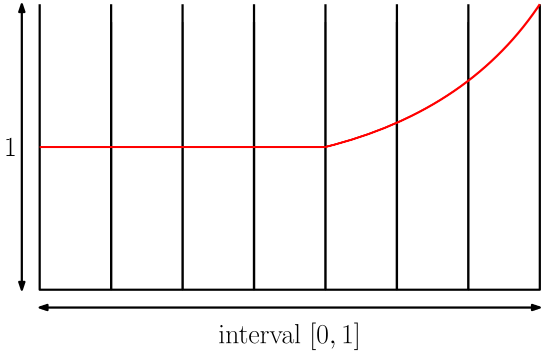
Rising Threshold Algorithm
Threshold function

$$f(x) = \max\{1/2, (2e)^{x-1}\}$$



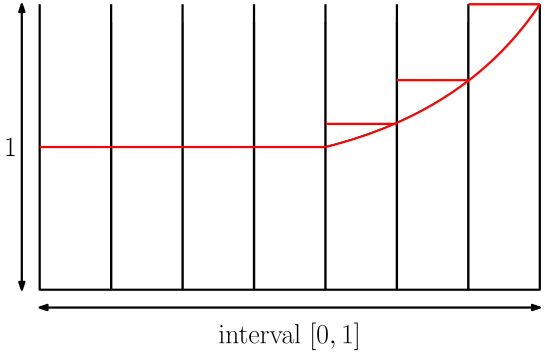
Rising Threshold Algorithm
Threshold function

$$f(x) = \max\{1/2, (2e)^{x-1}\}$$



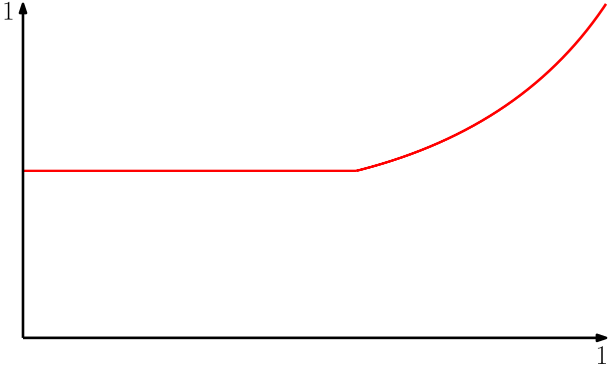
Rising Threshold Algorithm
Threshold function

$$f(x) = \max\{1/2, (2e)^{x-1}\}$$



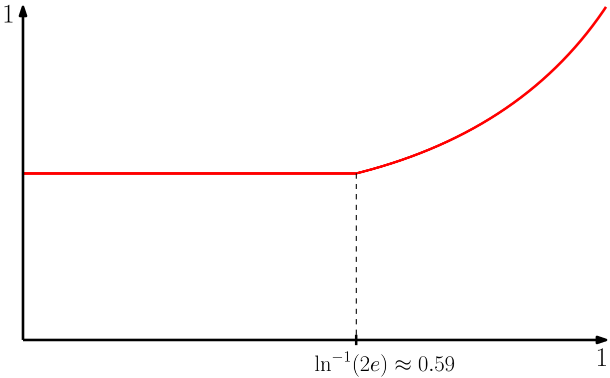
Rising Threshold Algorithm
Basic properties of f

$$f(x) = \max\{1/2, (2e)^{x-1}\}$$



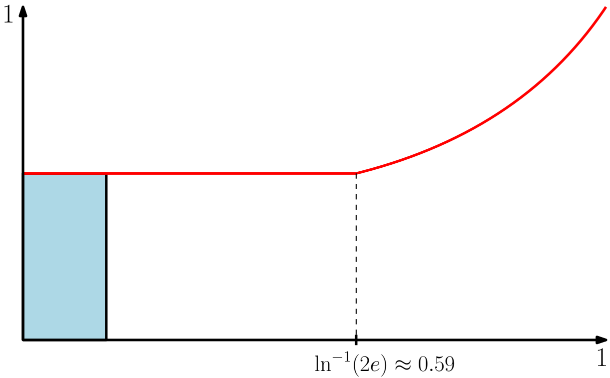
Rising Threshold Algorithm
Basic properties of f

$$f(x) = \max\{1/2, (2e)^{x-1}\}$$



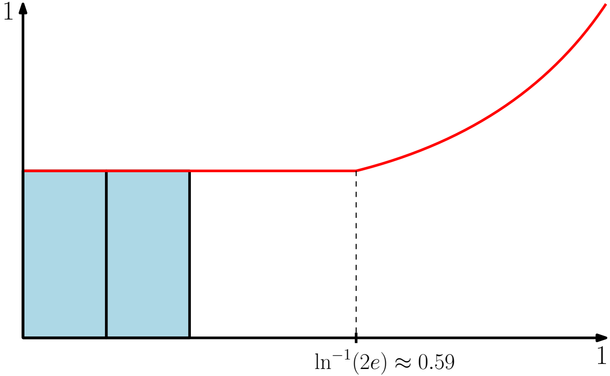
Rising Threshold Algorithm
Basic properties of f

$$f(x) = \max\{1/2, (2e)^{x-1}\}$$



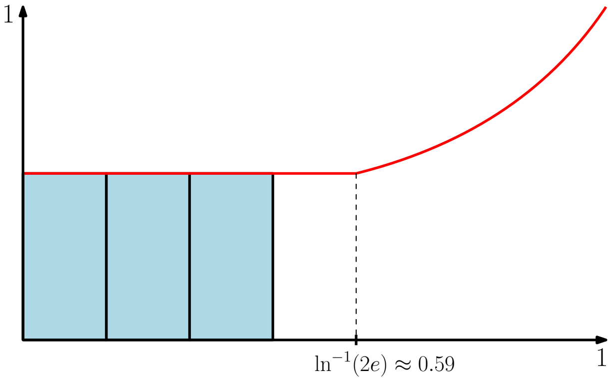
Rising Threshold Algorithm
Basic properties of f

$$f(x) = \max\{1/2, (2e)^{x-1}\}$$



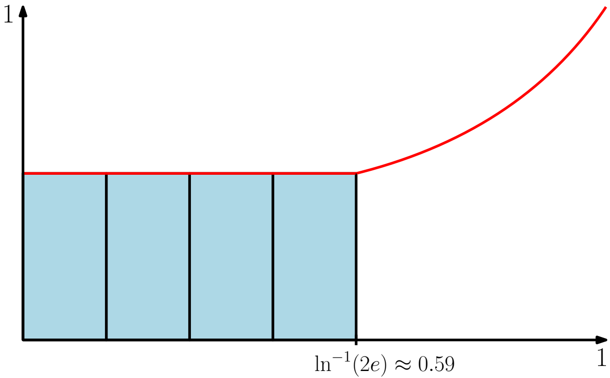
Rising Threshold Algorithm
Basic properties of f

$$f(x) = \max\{1/2, (2e)^{x-1}\}$$



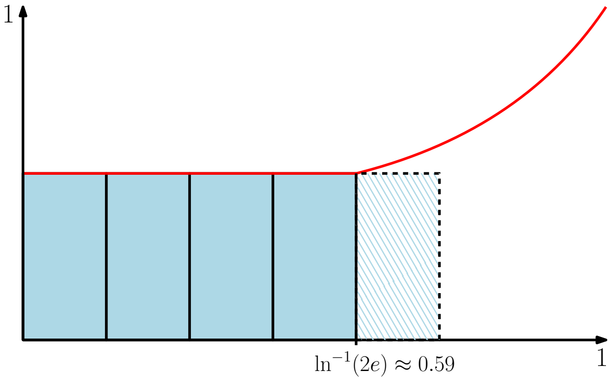
Rising Threshold Algorithm
Basic properties of f

$$f(x) = \max\{1/2, (2e)^{x-1}\}$$



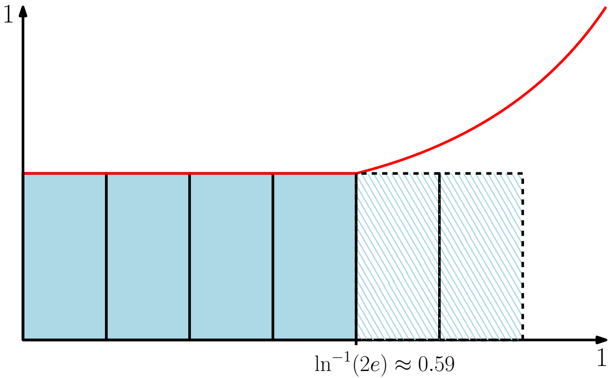
Rising Threshold Algorithm
Basic properties of f

$$f(x) = \max\{1/2, (2e)^{x-1}\}$$



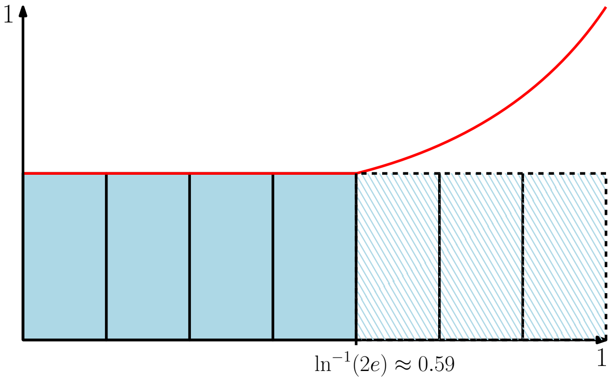
Rising Threshold Algorithm
Basic properties of f

$$f(x) = \max\{1/2, (2e)^{x-1}\}$$



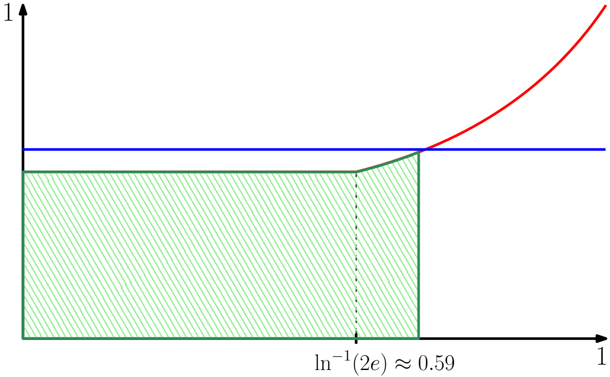
Rising Threshold Algorithm
Basic properties of f

$$f(x) = \max\{1/2, (2e)^{x-1}\}$$



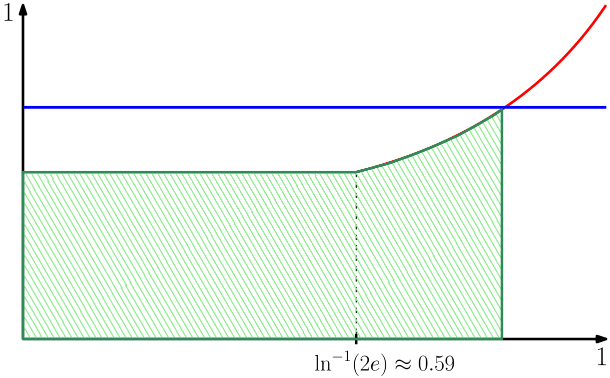
Rising Threshold Algorithm
Most important property of f

$$\frac{\int^{x} f(t)dt}{f(x)\cdot 1} = \frac{\text{green area}}{\text{area below blue}} = \ln^{-1}(2e) \approx 0.59$$



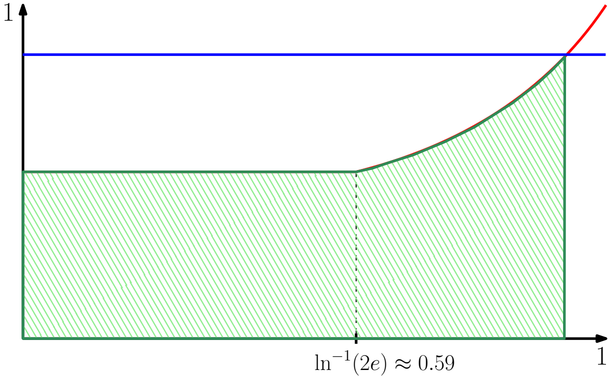
Rising Threshold Algorithm
Most important property of f

$$\frac{\int^{x} f(t)dt}{f(x)\cdot 1} = \frac{\text{green area}}{\text{area below blue}} = \ln^{-1}(2e) \approx 0.59$$



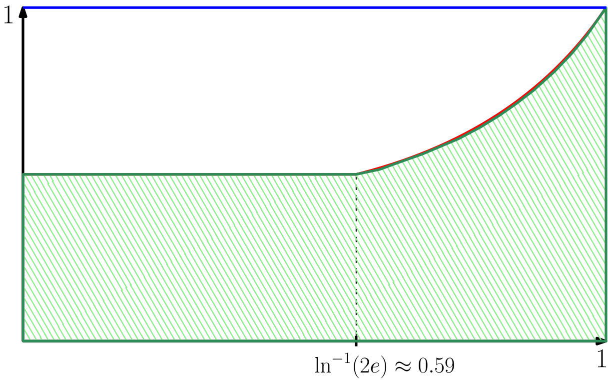
Rising Threshold Algorithm
Most important property of f

$$\frac{\int^x f(t)dt}{f(x)\cdot 1} = \frac{\text{green area}}{\text{area below blue}} = \ln^{-1}(2e) \approx 0.59$$



Rising Threshold Algorithm
Most important property of f

$$\frac{\int^x f(t)dt}{f(x)\cdot 1} = \frac{\text{green area}}{\text{area below blue}} = \ln^{-1}(2e) \approx 0.59$$



Rising Threshold Algorithm

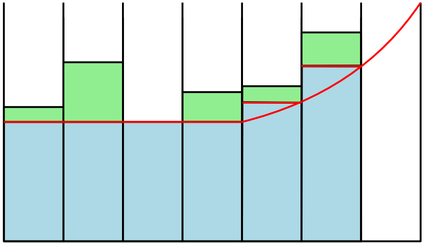
Analysis for large

$$=$$

$$\frac{\int^{x} f(t)dt}{f(x) \cdot 1} \approx 0.59$$

$$+$$

items exceeding threshold benefit both ALG and OPT



Rising Threshold Algorithm
Next steps

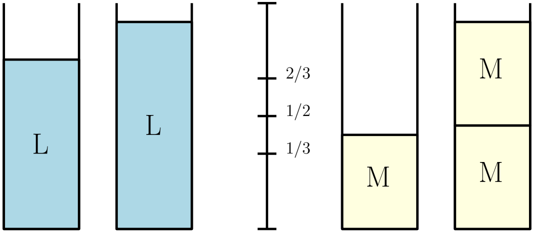
- Step 1. Algorithm for large items $(1/2, 1]$

Rising Threshold Algorithm
Next steps

- Step 1. Algorithm for large items $(1/2, 1]$ $\checkmark$

Rising Threshold Algorithm
Next steps

- Step 1. Algorithm for large items $(1/2, 1]$ ✓
- Step 2. Algorithm for large and medium items $(1/3, 1/2]$:



(two medium items fit in 1 knapsack)

Adding medium items $(1/3, 1/2]$

Algorithm properties

- take large items according to threshold
- never reject medium items

Adding medium items $(1/3, 1/2]$

Algorithm properties

- take large items according to threshold
- never reject medium items

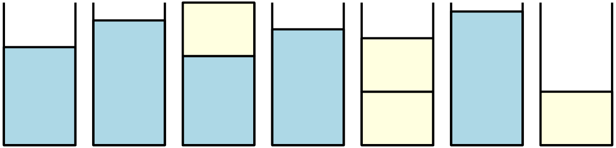
$$\Downarrow$$

Observation. If finished with some empty knapsacks $\Rightarrow$ optimal!

$$\frac{ALG_L + M}{OPT_L + M} \geq \frac{ALG_L}{OPT_L}$$

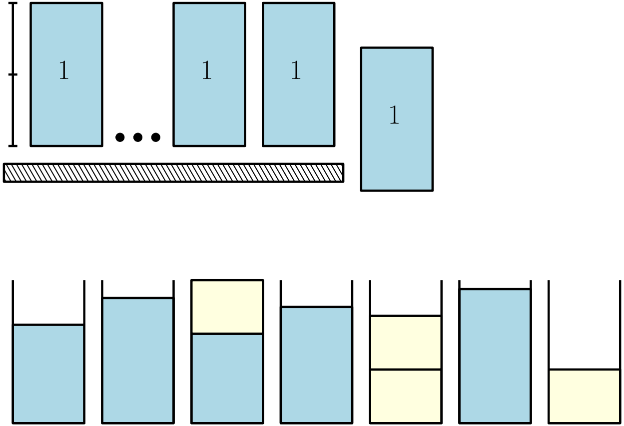
Adding medium items $(1/3, 1/2]$

- Case 1. Finished with some empty knapsacks ✓
- Case 2. Finished with no empty knapsacks:



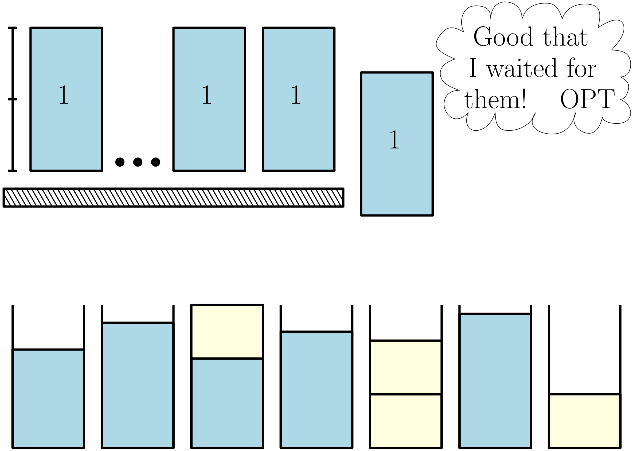
Adding medium items $(1/3, 1/2]$

- Case 1. Finished with some empty knapsacks ✓
- Case 2. Finished with no empty knapsacks:

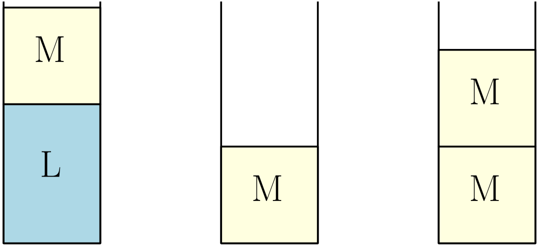


Adding medium items $(1/3, 1/2]$

- Case 1. Finished with some empty knapsacks ✓
- Case 2. Finished with no empty knapsacks:

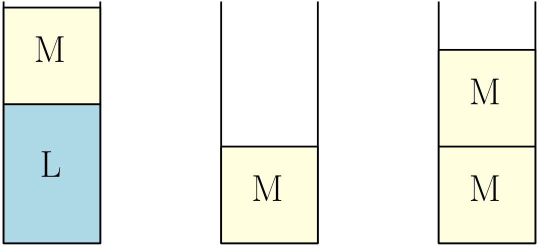


Three options to arrange mediums



combine (with large) | wait (for large) | stack (with medium)

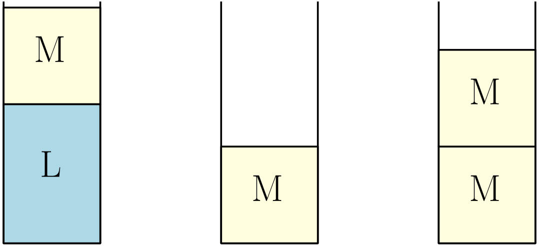
How we arrange mediums



combine (with large) | wait (for large) | stack (with medium)

ALG:

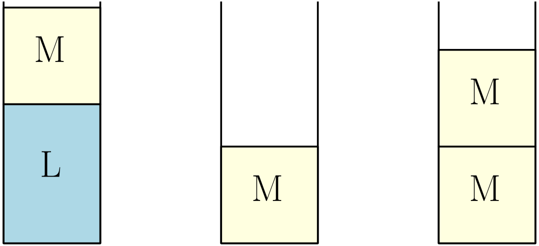
How we arrange mediums



combine (with large) | wait (for large) | stack (with medium)

ALG: $\boxed{\text{always attempt}}$

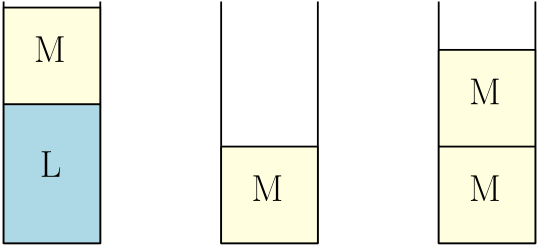
How we arrange mediums



combine	wait	stack
(with large)	(for large)	(with medium)

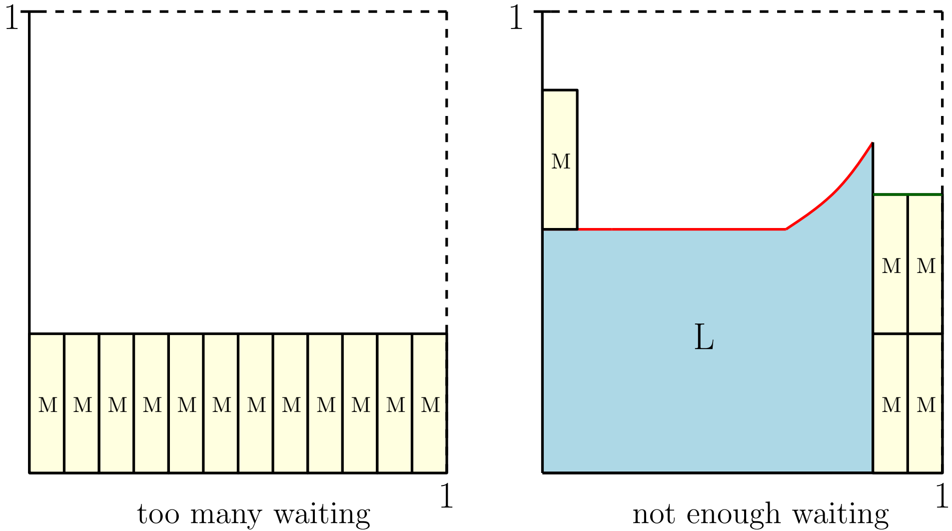
ALG: always attempt | keep some

How we arrange mediums

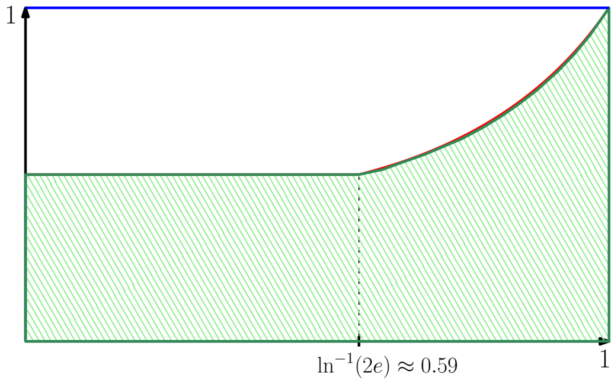


	combine	wait	stack
	(with large)	(for large)	(with medium)
ALG:	always attempt	keep some	if too many waits

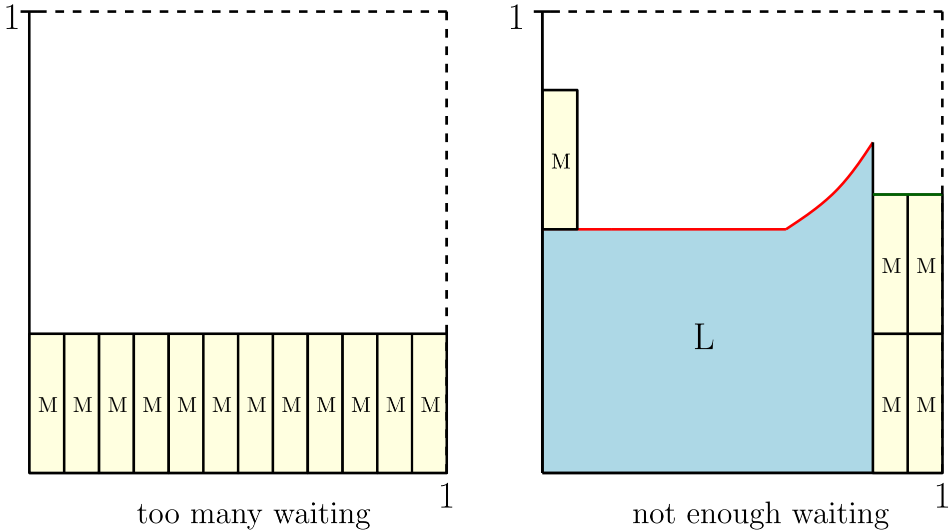
How many medium items should wait?



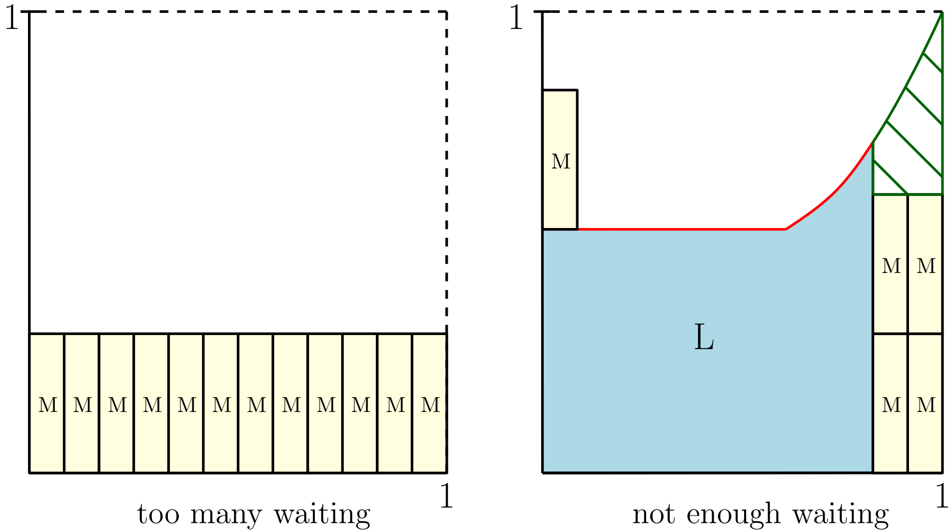
How many medium items should wait?



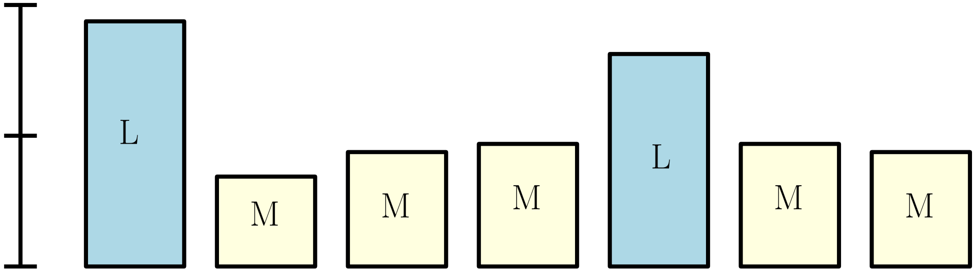
How many medium items should wait?



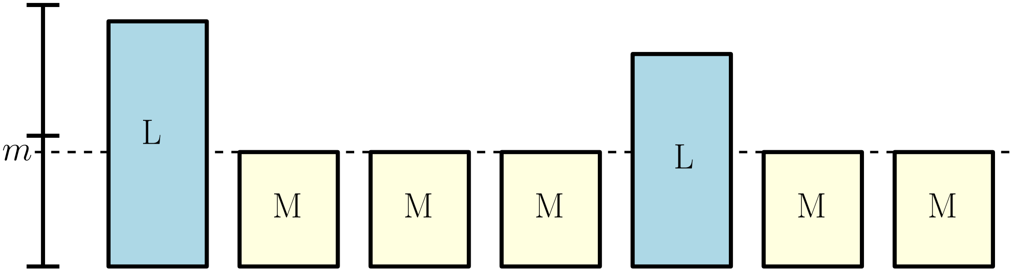
How many medium items should wait?



How many medium items should wait?



How many medium items should wait?



Simplify: all mediums of size $m \in (1/3, 1/2]$

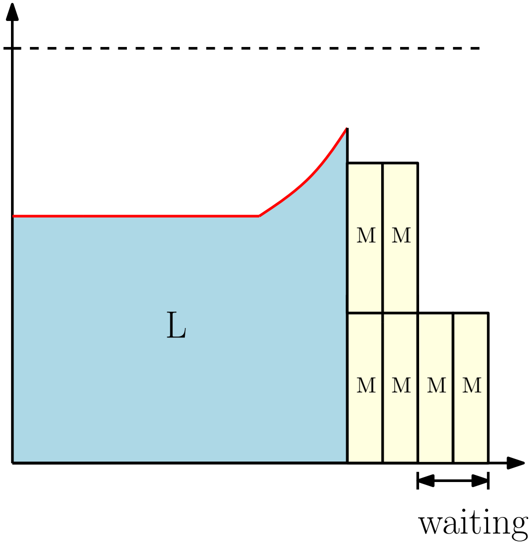
How many medium items should wait?

Answer: to have gain ≥ 0.59

How many medium items should wait?

Each medium item is size m

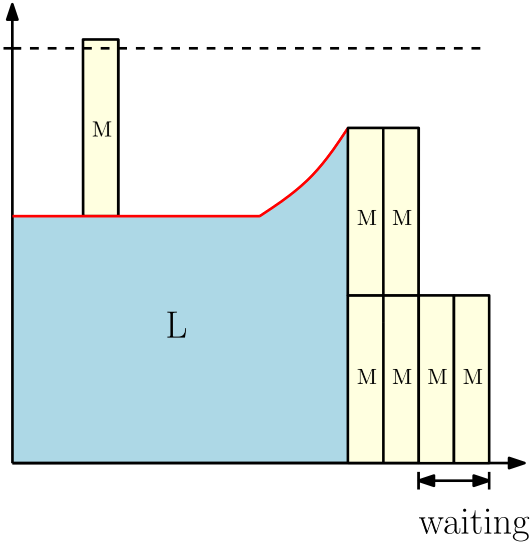
- gain on waiting $= m$
- gain on stacked $= 2m$
- gain on large $\geq 1 - m$



How many medium items should wait?

Each medium item is size m

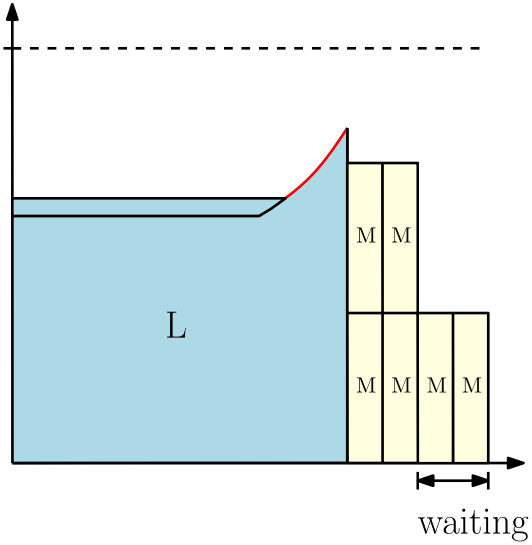
- gain on waiting $= m$
- gain on stacked $= 2m$
- gain on large $\geq 1 - m$



How many medium items should wait?

Each medium item is size m

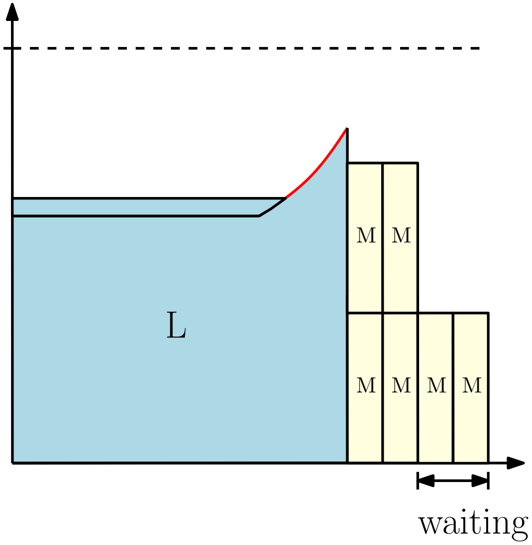
- gain on waiting $= m$
- gain on stacked $= 2m$
- gain on large $\geq 1 - m$



How many medium items should wait?

Each medium item is size m

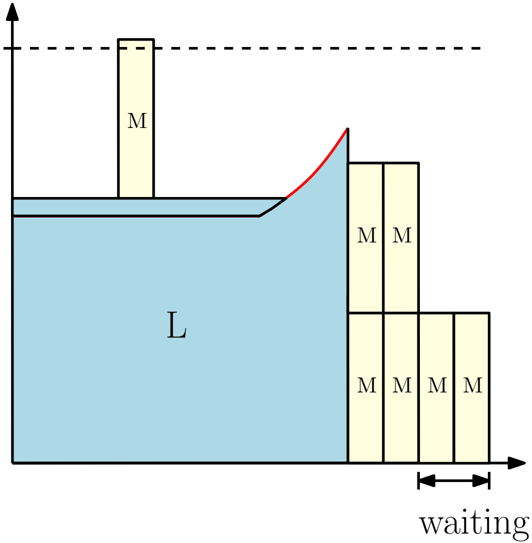
- gain on waiting $= m$
- gain on stacked $= 2m$
- gain on large $\geq 1 - m$



How many medium items should wait?

Each medium item is size m

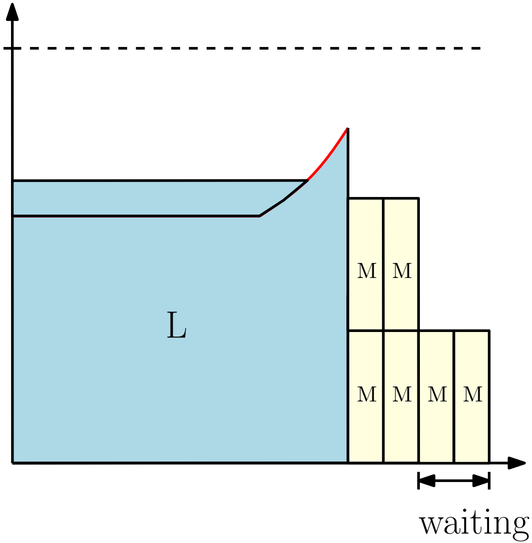
- gain on waiting $= m$
- gain on stacked $= 2m$
- gain on large $\geq 1 - m$



How many medium items should wait?

Each medium item is size m

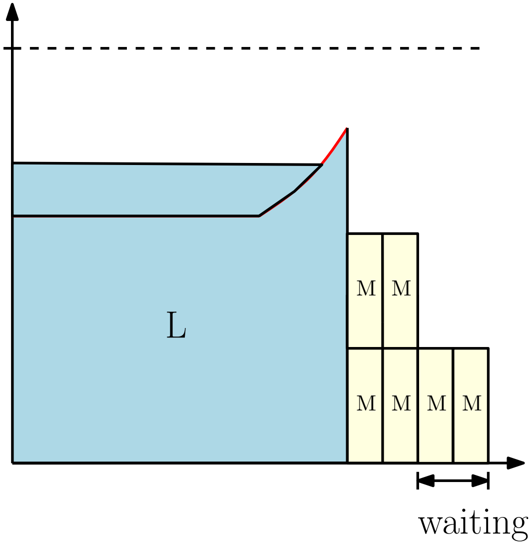
- gain on waiting $= m$
- gain on stacked $= 2m$
- gain on large $\geq 1 - m$



How many medium items should wait?

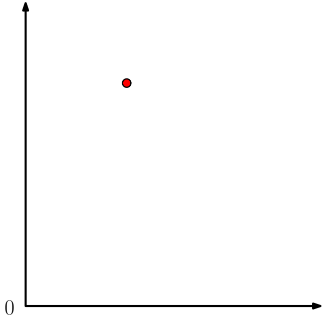
Each medium item is size m

- gain on waiting $= m$
- gain on stacked $= 2m$
- gain on large $\geq 1 - m$



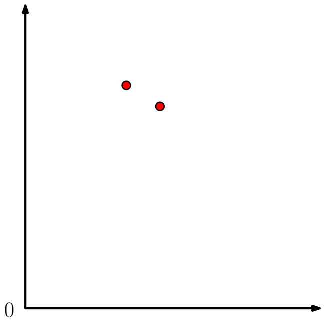
How many medium items should wait?

Answer: fix m, solve for gain(#waiting) ≥ 0.59



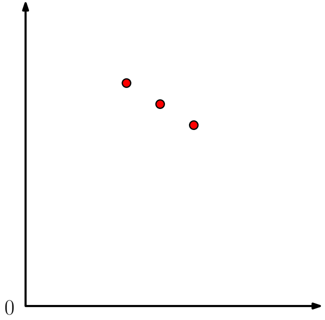
How many medium items should wait?

Answer: fix m, solve for gain(#waiting) ≥ 0.59



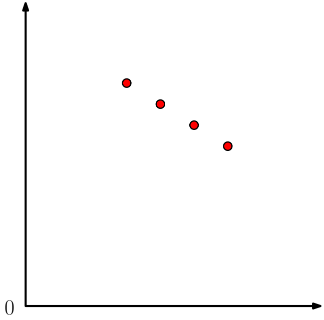
How many medium items should wait?

Answer: fix m, solve for gain(#waiting) ≥ 0.59



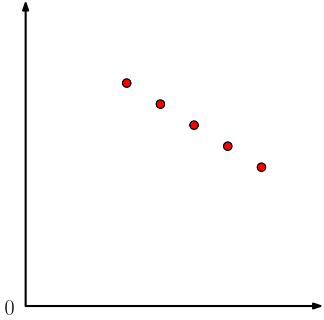
How many medium items should wait?

Answer: fix m, solve for gain(#waiting) ≥ 0.59



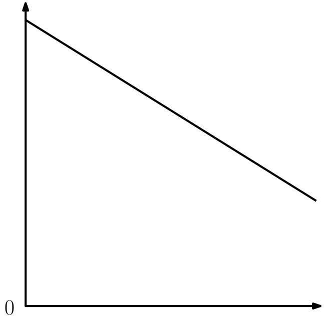
How many medium items should wait?

Answer: fix m, solve for gain(#waiting) ≥ 0.59

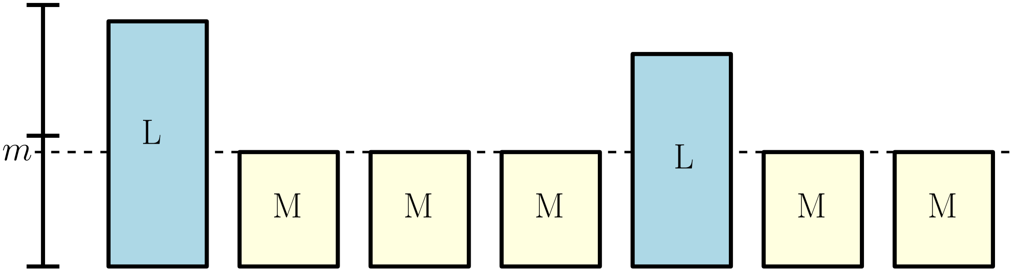


How many medium items should wait?

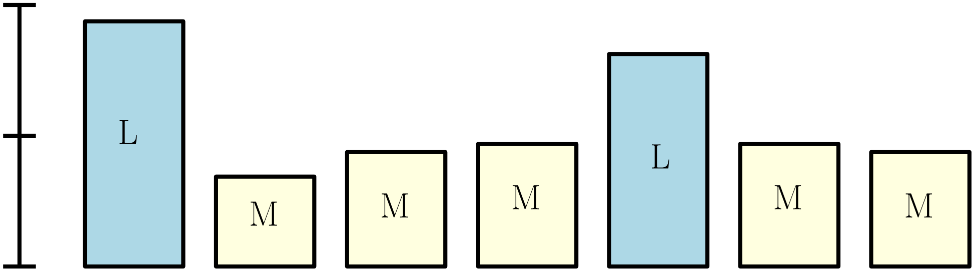
Answer: fix m, solve for gain(#waiting) ≥ 0.59



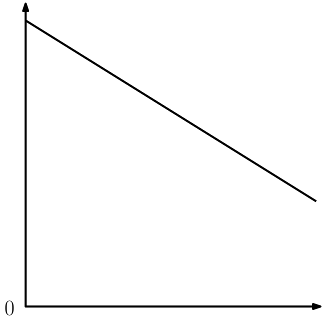
Beyond single medium item



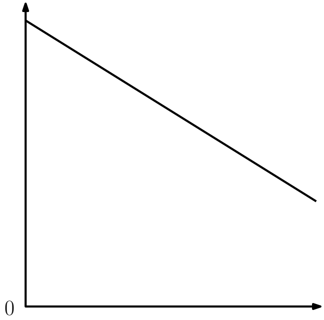
Beyond single medium item



Beyond single medium item size



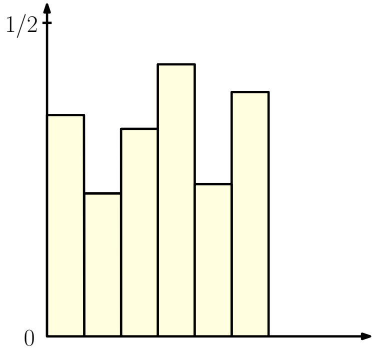
Beyond single medium item size



How many should wait? For different m, different answer!

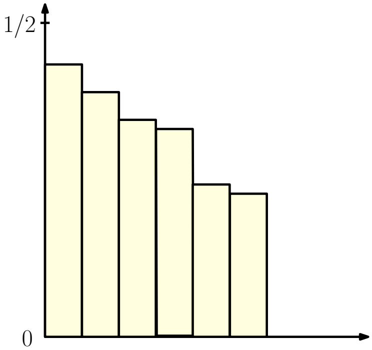
Beyond single medium item size

- Sort waiting medium items
- Incoming medium item waits if fits below the curve



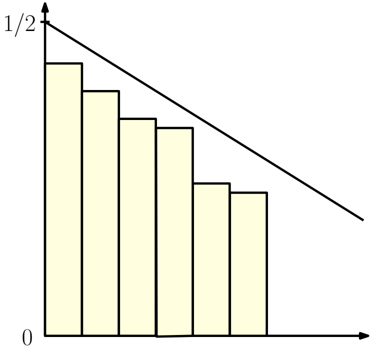
Beyond single medium item size

- Sort waiting medium items
- Incoming medium item waits if fits below the curve



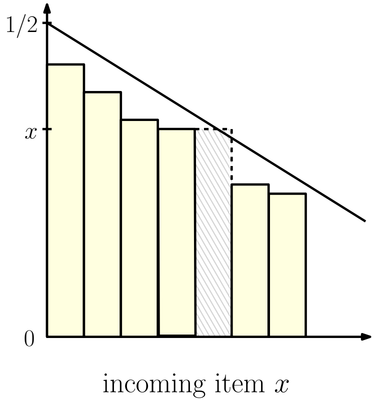
Beyond single medium item size

- Sort waiting medium items
- Incoming medium item waits if fits below the curve

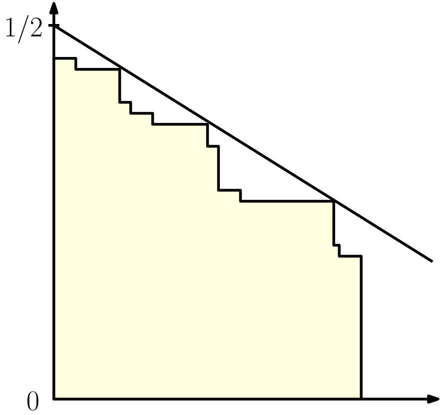


Beyond single medium item size

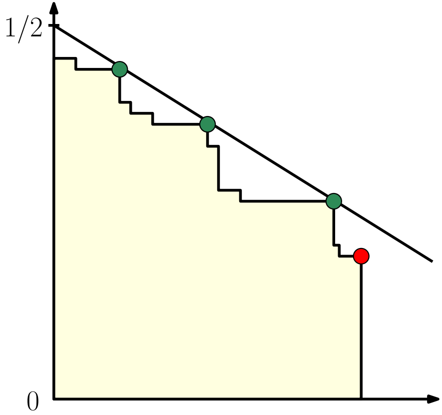
- Sort waiting medium items
- Incoming medium item waits if fits below the curve



Analysis

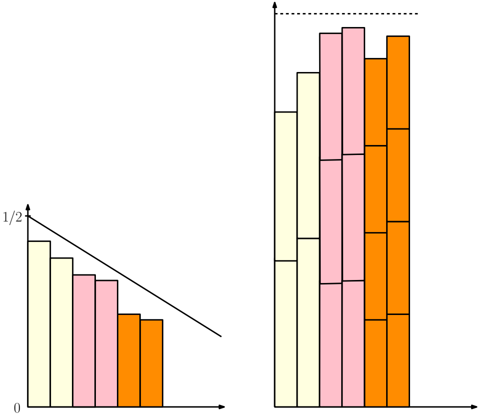


Analysis



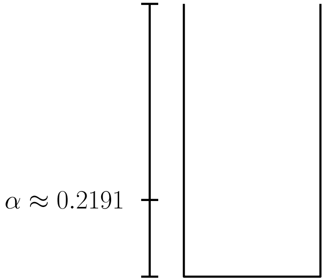
Possible to extend for $(\alpha, 1]$

$(\alpha \approx 0.2192)$



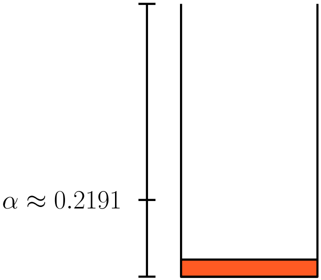
Next steps: algorithm for small items $(0, \alpha]$

just the idea – simply stacking small items is not enough



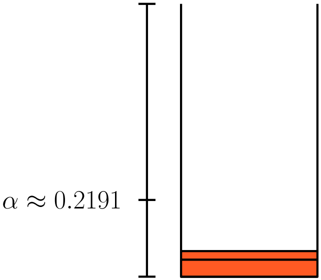
Next steps: algorithm for small items $(0, \alpha]$

just the idea – simply stacking small items is not enough



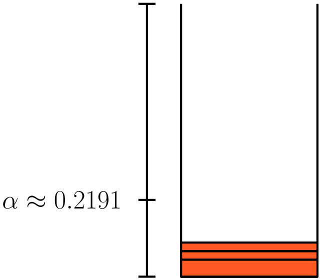
Next steps: algorithm for small items $(0, \alpha]$

just the idea – simply stacking small items is not enough



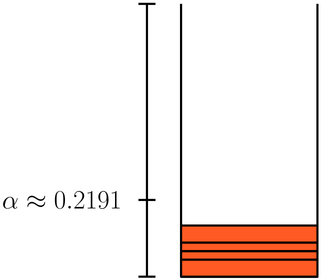
Next steps: algorithm for small items $(0, \alpha]$

just the idea – simply stacking small items is not enough



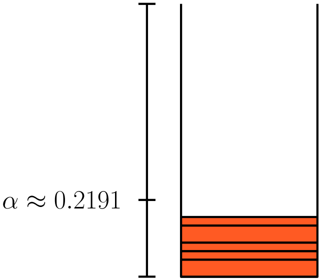
Next steps: algorithm for small items $(0, \alpha]$

just the idea – simply stacking small items is not enough



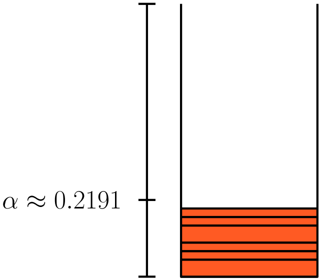
Next steps: algorithm for small items $(0, \alpha]$

just the idea – simply stacking small items is not enough



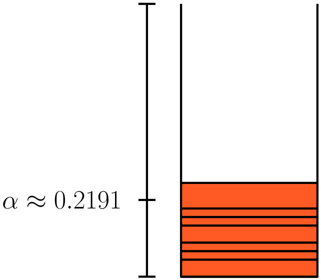
Next steps: algorithm for small items $(0, \alpha]$

just the idea – simply stacking small items is not enough



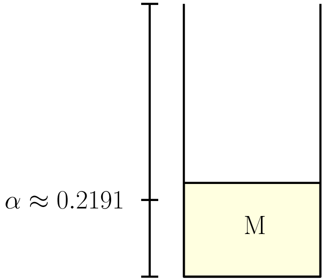
Next steps: algorithm for small items $(0, \alpha]$

just the idea – simply stacking small items is not enough



Next steps: algorithm for small items $(0, \alpha]$

just the idea – simply stacking small items is not enough

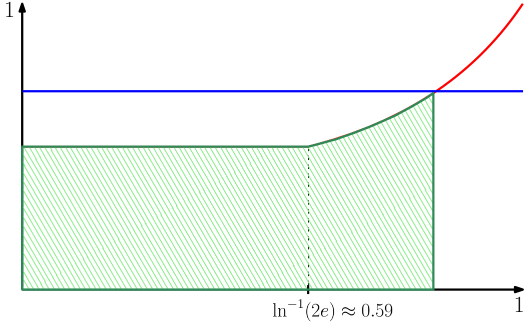


Rising Threshold Algorithm is optimal for Online Knapsack

and the function

$$f(x) = \max\{1/2, (2e)^{x-1}\}$$

is natural for this problem

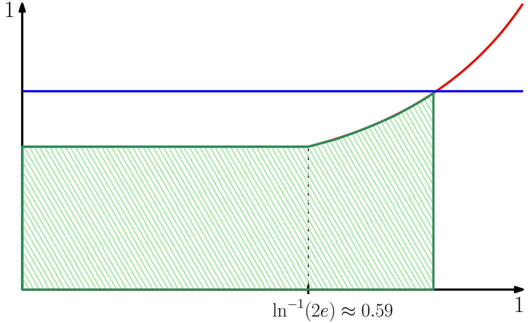


Rising Threshold Algorithm is optimal for Online Knapsack

and the function

$$f(x) = \max\{1/2, (2e)^{x-1}\}$$

is natural for this problem



maciej.pacut@univie.ac.at