Algorithms for Decision Support

Beyond the Worst Case

Machine-Learned Advice

Outline

Deal with Uncertainty

- Online algorithms deal with optimization under uncertainty (about future input)

Deal with Uncertainty

- Online algorithms deal with optimization under uncertainty (about future input)
  - How if the future information can be predicted or learned by machine-learning?

Deal with Uncertainty

- Online algorithms deal with optimization under uncertainty (about future input)
  - How if the future information can be predicted or learned by machine-learning?
    - Example: weather forecast

Deal with Uncertainty

- Online algorithms deal with optimization under uncertainty (about future input)
  - How if the future information can be predicted or learned by machine-learning?
    - Example: weather forecast
    - These predictions or learned information may not be 100% correct

Deal with Uncertainty

- Online algorithms deal with optimization under uncertainty (about future input)
  - How if the future information can be predicted or learned by machine-learning?
    - Example: weather forecast
    - These predictions or learned information may not be 100% correct
      - Completely trust the predictions may be a disaster

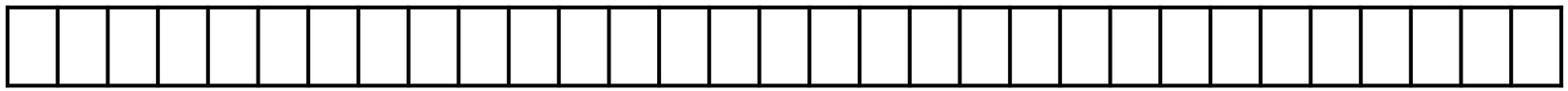
Deal with Uncertainty

- Online algorithms deal with optimization under uncertainty (about future input)
  - How if the future information can be predicted or learned by machine-learning?
    - Example: weather forecast
    - These predictions or learned information may not be 100% correct
      - Completely trust the predictions may be a disaster
      - How can we design online algorithms with this kind of (maybe) untrusted advice?

Searching with Machine-Learned Advice

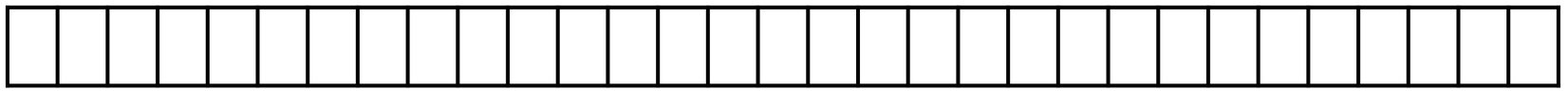
- Goal: Given an ordered sequence and a requested value, find the requested value in minimum number of steps

Searching with Machine-Learned Advice

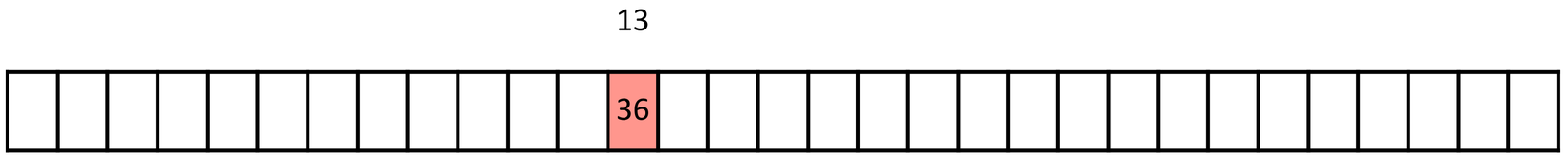


Searching with Machine-Learned Advice

13



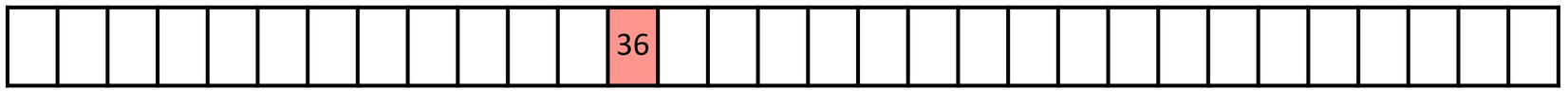
Searching with Machine-Learned Advice



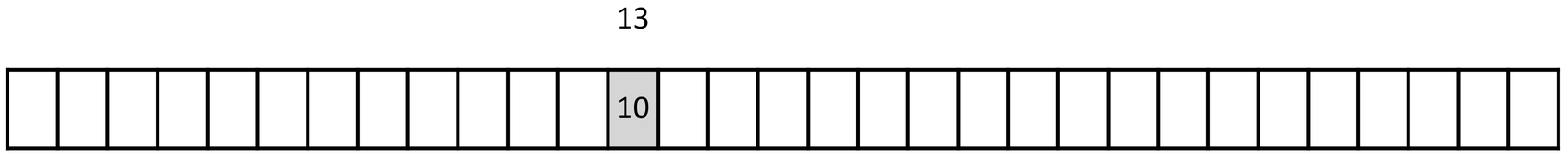
Searching with Machine-Learned Advice

Cost = 1

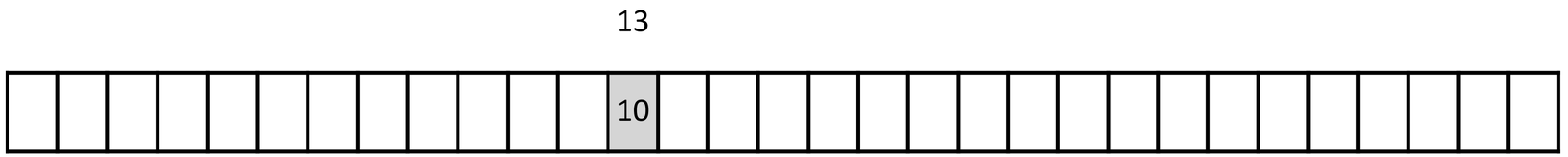
13



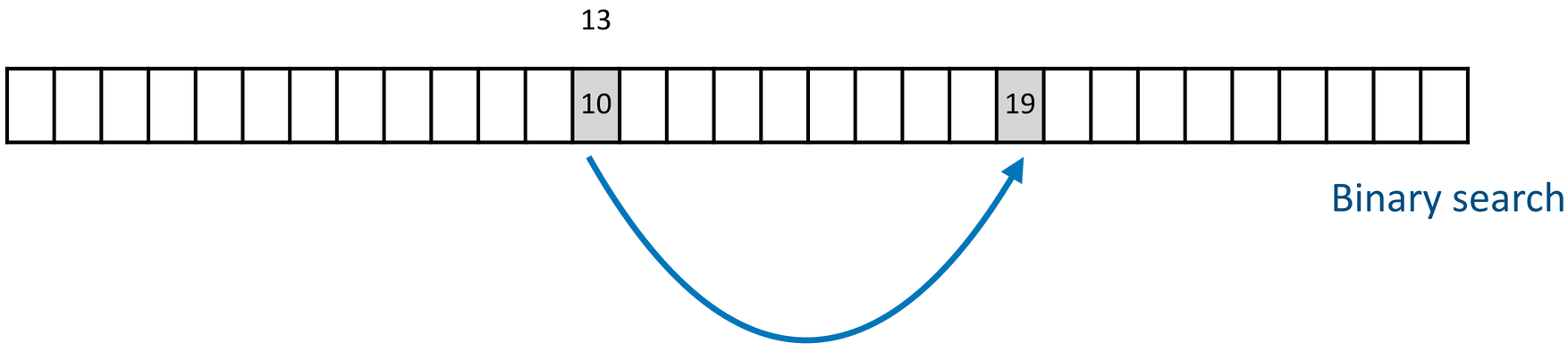
Searching with Machine-Learned Advice



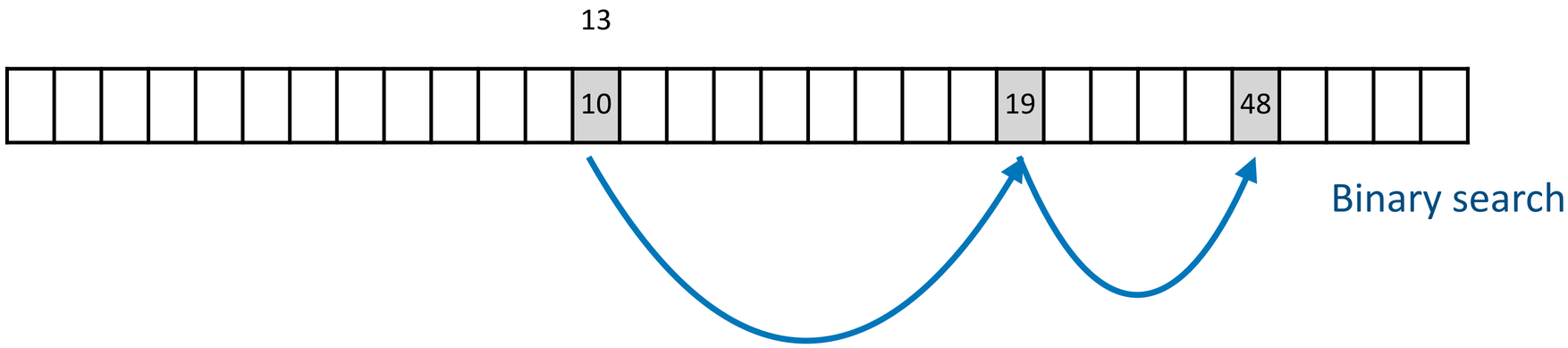
Searching with Machine-Learned Advice



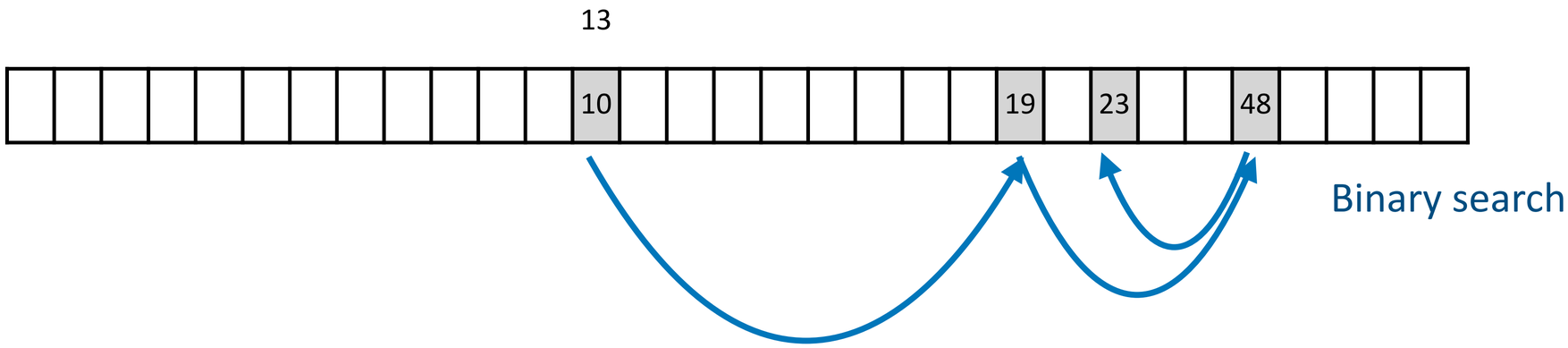
Searching with Machine-Learned Advice



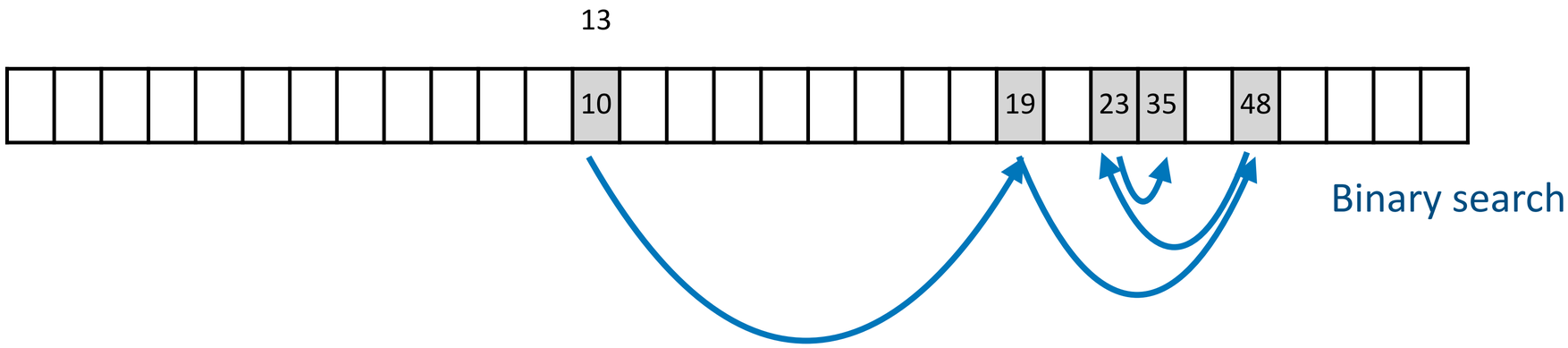
Searching with Machine-Learned Advice



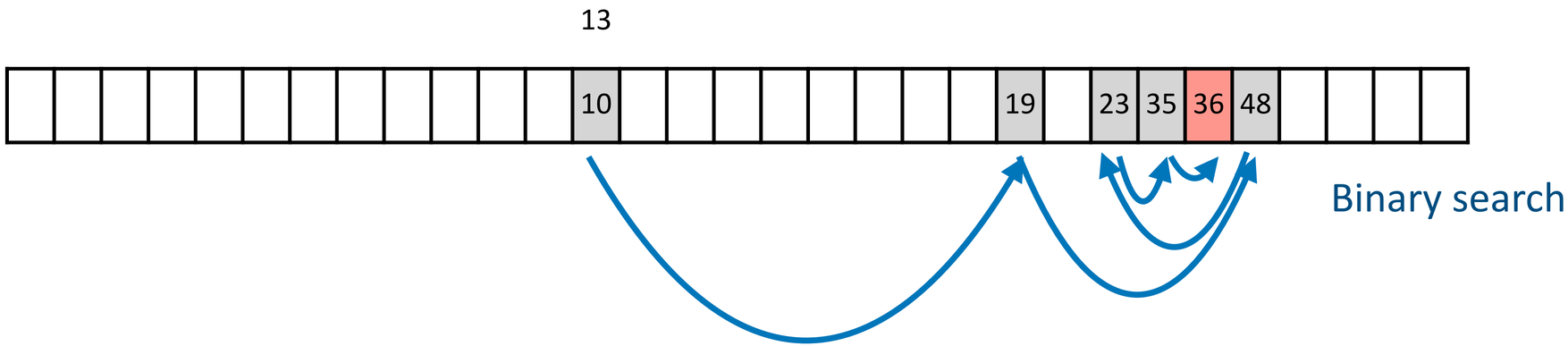
Searching with Machine-Learned Advice



Searching with Machine-Learned Advice

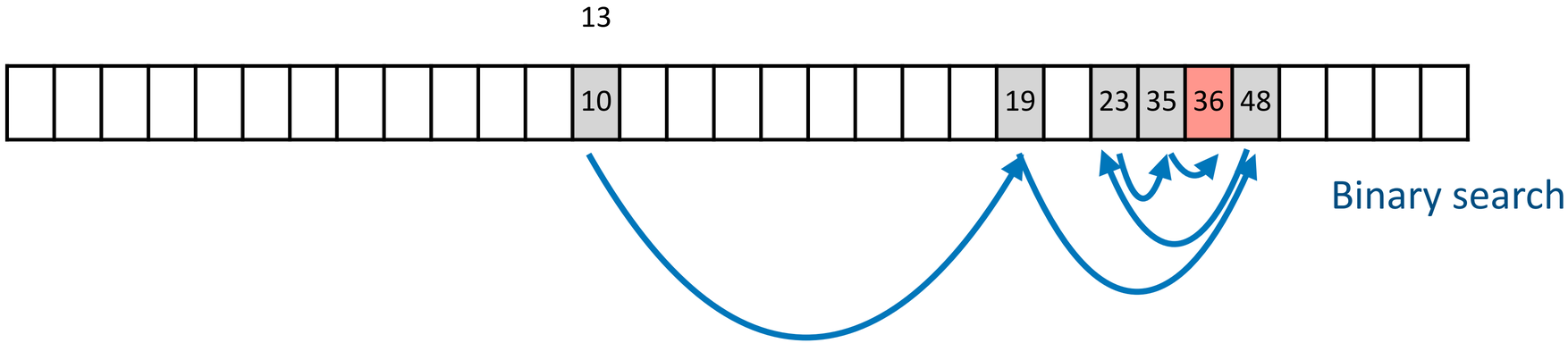


Searching with Machine-Learned Advice



Searching with Machine-Learned Advice

Cost = $\log n$

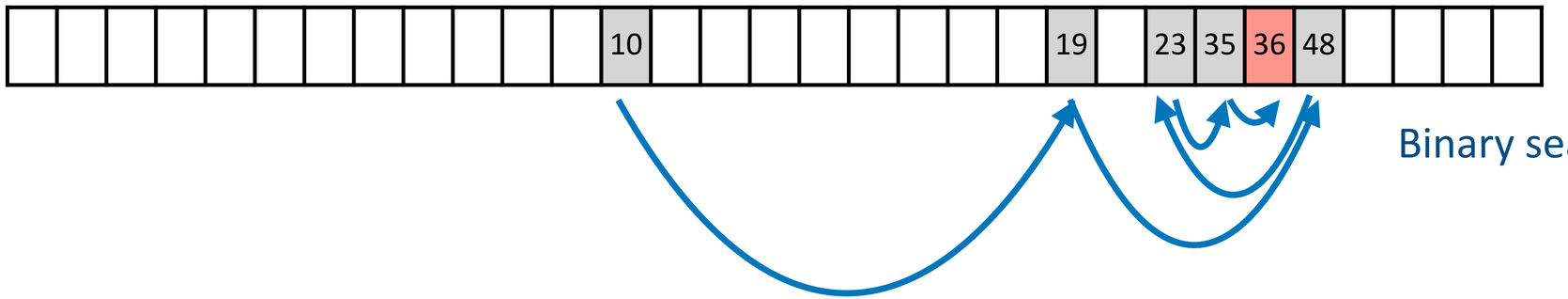


Consistency and Robustness

- We look for algorithms that
  - Are **consistent**: given more accurate predictions, the online algorithm should perform close to the optimal offline algorithm

Cost = 1

  - Are **robust**: if the prediction is wrong, the online algorithm performance should be close to the online algorithm without predictions



Cost = $\log n$

Ski Rental with Machine Learned Advice

- Imagine that you are having a ski holiday
  - The price of renting a ski is 1 per day
  - The buying price for ski is B.
  - Cost: the total money you pay
  - p: machine-learned prediction on the number of skiing days
- Suppose you want to spend money as little as possible. Should you buy the ski or rent it?

Ski Rental Algorithm 1

SKI-Rental with prediction (p)

If $p \geq B$
    Buy the ski on the first day
else ($p < B$)
    Keep **renting** for all skiing days

Ski Rental Algorithm 1

SKI-Rental with prediction (p)

```
If p ≥ B
    Buy the ski on the first day
else ( p < B )
    Keep renting for all skiing days
```

S: Actual number of skiing days

Truth: $S \geq B$ (OPT buy)	Truth: $S < B$ (OPT rent)

Ski Rental Algorithm 1

SKI-Rental with prediction (p)

```
If p ≥ B
    Buy the ski on the first day
else ( p < B )
    Keep renting for all skiing days
```

S: Actual number of skiing days

	Truth: $S \geq B$ (OPT buy)	Truth: $S < B$ (OPT rent)
 Good prediction		
 Bad prediction		

Ski Rental Algorithm 1

SKI-Rental with prediction (p)

```
If p ≥ B
    Buy the ski on the first day
else ( p < B )
    Keep renting for all skiing days
```

S: Actual number of skiing days

	Truth: $S \geq B$ (OPT buy)	Truth: $S < B$ (OPT rent)
 Good prediction	Advice: $p \geq B$ $\frac{B}{B}$	
 Bad prediction		

Ski Rental Algorithm 1

SKI-Rental with prediction (p)

```
If p ≥ B
    Buy the ski on the first day
else ( p < B )
    Keep renting for all skiing days
```

S: Actual number of skiing days

	Truth: $S \geq B$ (OPT buy)	Truth: $S < B$ (OPT rent)
 Good prediction	Advice: $p \geq B$ $\frac{B}{B}$	Advice: $p < B$ $\frac{S}{S}$
 Bad prediction		

Ski Rental Algorithm 1

SKI-Rental with prediction (p)

```
If p ≥ B
    Buy the ski on the first day
else ( p < B )
    Keep renting for all skiing days
```

S: Actual number of skiing days

	Truth: $S \geq B$ (OPT buy)	Truth: $S < B$ (OPT rent)
 Good prediction	Advice: $p \geq B$ $\frac{B}{B}$	Advice: $p < B$ $\frac{S}{S}$
 Bad prediction	Advice: $p < B$ $\frac{S}{B}$	

Ski Rental Algorithm 1

SKI-Rental with prediction (p)

```
If p ≥ B
    Buy the ski on the first day
else ( p < B )
    Keep renting for all skiing days
```

S: Actual number of skiing days

	Truth: $S \geq B$ (OPT buy)	Truth: $S < B$ (OPT rent)
 Good prediction	Advice: $p \geq B$ $\frac{B}{B}$	Advice: $p < B$ $\frac{S}{S}$
 Bad prediction	Advice: $p < B$ $\frac{S}{B} = \frac{\infty}{B} = \infty$	

Ski Rental Algorithm 1

SKI-Rental with prediction (p)

```
If p ≥ B
    Buy the ski on the first day
else ( p < B )
    Keep renting for all skiing days
```

S: Actual number of skiing days

	Truth: $S \geq B$ (OPT buy)	Truth: $S < B$ (OPT rent)
 Good prediction	Advice: $p \geq B$ $\frac{B}{B}$	Advice: $p < B$ $\frac{S}{S}$
 Bad prediction	Advice: $p < B$ $\frac{S}{B} = \frac{\infty}{B} = \infty$	Advice: $p \geq B$ $\frac{B}{S}$

Ski Rental Algorithm 1

SKI-Rental with prediction (p)

```
If p ≥ B
    Buy the ski on the first day
else ( p < B )
    Keep renting for all skiing days
```

S: Actual number of skiing days

	Truth: $S \geq B$ (OPT buy)	Truth: $S < B$ (OPT rent)
 Good prediction	Advice: $p \geq B$ $\frac{B}{B}$	Advice: $p < B$ $\frac{S}{S}$
 Bad prediction	Advice: $p < B$ $\frac{S}{B} = \frac{\infty}{B} = \infty$	Advice: $p \geq B$ $\frac{B}{S} \leq \frac{B}{1} = B$

Ski Rental Algorithm 1

SKI-Rental with prediction (p)

```
If p ≥ B
    Buy the ski on the first day
else ( p < B )
    Keep renting for all skiing days
```

S: Actual number of skiing days

	Truth: $S \geq B$ (OPT buy)	Truth: $S < B$ (OPT rent)
 Good prediction *Consistency*	Advice: $p \geq B$ $\frac{B}{B} = 1$	Advice: $p < B$ $\frac{S}{S} = 1$
 Bad prediction *Robustness*	Advice: $p < B$ $\frac{S}{B} = \frac{\infty}{B} = \infty$	Advice: $p \geq B$ $\frac{B}{S} \leq \frac{B}{1} = B$

Ski Rental Algorithm 1

SKI-Rental with prediction (p)

```
If p ≥ B
    Buy the ski on the first day
else ( p < B )
    Keep renting for all skiing days
```

S: Actual number of skiing days

	Truth: $S \geq B$ (OPT buy)	Truth: $S < B$ (OPT rent)
Good prediction *Consistency*	Advice: $p \geq B$ $\frac{B}{B} = 1$	Advice: $p < B$ $\frac{S}{S} = 1$
Bad prediction *Robustness*	Advice: $p < B$ $\frac{S}{B} = \frac{\infty}{B}$	Advice: $p \geq B$ $\frac{B}{S} \leq \frac{B}{1} = B$

To be robust, we want an algorithm close to 2-competitive in this case

Ski Rental Algorithm 1.5

SKI-Rental with prediction (p)

If $p \geq B$
    Buy the ski on the first day
else ($p < B$)
    Keep **renting** until the B-th day

S: Actual number of skiing days

Ski Rental Algorithm 1.5

SKI-Rental with prediction (p)

```
If p ≥ B
    Buy the ski on the first day
else ( p < B )
    Keep renting until the B-th day
```

S: Actual number of skiing days

	Truth: $S \geq B$ (OPT buy)	Truth: $S < B$ (OPT rent)
 Good prediction *Consistency*	Advice: $p \geq B$ $\frac{B}{B} = 1$	Advice: $p < B$ $\frac{S}{S} = 1$
 Bad prediction *Robustness*		Advice: $p \geq B$ $\frac{B}{S} \leq \frac{B}{1} = B$

Ski Rental Algorithm 1.5

SKI-Rental with prediction (p)

If $p \geq B$
    Buy the ski on the first day
else ($p < B$)
    Keep **renting** until the B-th day

S: Actual number of skiing days

	Truth: $S \geq B$ (OPT buy)	Truth: $S < B$ (OPT rent)
 Good prediction *Consistency*	Advice: $p \geq B$ $\frac{B}{B} = 1$	Advice: $p < B$ $\frac{S}{S} = 1$
 Bad prediction *Robustness*	Advice: $p < B$ $\frac{2B-1}{B}$	Advice: $p \geq B$ $\frac{B}{S} \leq \frac{B}{1} = B$

Ski Rental Algorithm 1.5

SKI-Rental with prediction (p)

```
If p ≥ B
    Buy the ski on the first day
else ( p < B )
    Keep renting for all skiing days
```

S: Actual number of skiing days

	Truth: $S \geq B$ (OPT buy)	Truth: $S < B$ (OPT rent)
Good prediction *Consistency*	Advice: $p \geq B$ $\frac{B}{B} = 1$	Advice: $p < B$ $\frac{S}{S} = 1$
Bad prediction *Robustness*	Advice: $p < B$ $\frac{2B-1}{B}$	Advice: $p \geq B$ $\frac{B}{S} \leq \frac{B}{1} = B$

To be robust, we want an algorithm close to 2-competitive in this case

What Happened

- Blindly trusting the prediction can cause disasters!

Trustness Parameter

- We introduce a parameter k to indicate how much the algorithm trust the advice
  - $k \in [1, B]$
    - $k = 1$: the algorithm fully trusts the advice
    - $k = B$: the algorithm does not trust the advice at all

Ski Rental Algorithm 2

SKI-Rental with prediction (p, k)

If $p \geq B$

Keep renting until the k-th day // k is our "trust parameter"

else ($p < B$)

Keep renting until the B-th day

Ski Rental Algorithm 2

SKI-Rental with prediction (p, k)

If $p \geq B$

Keep renting until the k-th day // k is our "trust parameter"

else ($p < B$)

Keep renting until the B-th day

Fully trust the prediction

Ski Rental Algorithm 2

SKI-Rental with prediction (p, k)

If $p \geq B$

Keep renting until the k-th day // k is our "trust parameter"

else ($p < B$)

Keep renting until the B-th day

Does not trust the prediction at all

Ski Rental Algorithm 2

SKI-Rental with prediction (p, k)

If $p \geq B$

Keep renting until the k-th day // k is our "trust parameter"

else ($p < B$)

Keep renting until the B-th day S: Actual number of skiing days

Truth: $S \geq B$ (OPT buy)	Truth: $S < B$ (OPT rent)

Ski Rental Algorithm 2

SKI-Rental with prediction (p, k)

If $p \geq B$

Keep renting until the k-th day // k is our "trust parameter"

else ($p < B$)

Keep renting until the B-th day S: Actual number of skiing days

	Truth: $S \geq B$ (OPT buy)	Truth: $S < B$ (OPT rent)
 Good prediction		
 Bad prediction		

Ski Rental Algorithm 2

SKI-Rental with prediction (p, k)

If $p \geq B$

Keep renting until the k-th day // k is our "trust parameter"

else ($p < B$)

Keep renting until the B-th day S: Actual number of skiing days

	Truth: $S \geq B$ (OPT buy)	Truth: $S < B$ (OPT rent)
 Good prediction	Advice: $p \geq B$	Advice: $p < B$
 Bad prediction		

Ski Rental Algorithm 2

SKI-Rental with prediction (p, k)

If $p \geq B$

Keep renting until the k-th day // k is our "trust parameter"

else ($p < B$)

Keep renting until the B-th day S: Actual number of skiing days

	Truth: $S \geq B$ (OPT buy)	Truth: $S < B$ (OPT rent)
 Good prediction	Advice: $p \geq B$	Advice: $p < B$
 Bad prediction	Advice: $p < B$	Advice: $p \geq B$

Ski Rental Algorithm 2

SKI-Rental with prediction (p, k)

If $p \geq B$

Keep renting until the k-th day // k is our "trust parameter"

else ($p < B$)

Keep renting until the B-th day S: Actual number of skiing days

	Truth: $S \geq B$ (OPT buy)	Truth: $S < B$ (OPT rent)
Good prediction	Advice: $p \geq B$ $\frac{(k-1)+B}{B}$	Advice: $p < B$
Bad prediction	Advice: $p < B$	Advice: $p \geq B$

Ski Rental Algorithm 2

SKI-Rental with prediction (p, k)

If $p \geq B$

Keep renting until the k-th day // k is our "trust parameter"

else ($p < B$)

Keep renting until the B-th day S: Actual number of skiing days

	Truth: $S \geq B$ (OPT buy)	Truth: $S < B$ (OPT rent)
 Good prediction	Advice: $p \geq B$ $\frac{(k-1)+B}{B}$	Advice: $p < B$ $\frac{S}{S}$
 Bad prediction	Advice: $p < B$	Advice: $p \geq B$

Ski Rental Algorithm 2

SKI-Rental with prediction (p, k)

If $p \geq B$

Keep renting until the k-th day // k is our "trust parameter"

else ($p < B$)

Keep renting until the B-th day S: Actual number of skiing days

	Truth: $S \geq B$ (OPT buy)	Truth: $S < B$ (OPT rent)
 Good prediction	Advice: $p \geq B$ $\frac{(k-1)+B}{B}$	Advice: $p < B$ $\frac{S}{S}$
 Bad prediction	Advice: $p < B$	Advice: $p \geq B$

Ski Rental Algorithm 2

SKI-Rental with prediction (p, k)

If $p \geq B$

Keep renting until the k-th day // k is our "trust parameter"

else ($p < B$)

Keep renting until the B-th day S: Actual number of skiing days

	Truth: $S \geq B$ (OPT buy)	Truth: $S < B$ (OPT rent)
Good prediction	Advice: $p \geq B$ $\frac{(k-1)+B}{B}$	Advice: $p < B$ $\frac{S}{S}$
Bad prediction	Advice: $p < B$ $\frac{(B-1)+B}{B}$	Advice: $p \geq B$

Ski Rental Algorithm 2

SKI-Rental with prediction (p, k)

If $p \geq B$

Keep renting until the k-th day // k is our "trust parameter"

else ($p < B$)

Keep renting until the B-th day S: Actual number of skiing days

	Truth: $S \geq B$ (OPT buy)	Truth: $S < B$ (OPT rent)
 Good prediction	Advice: $p \geq B$ $\frac{(k-1)+B}{B}$	Advice: $p < B$ $\frac{S}{S}$
 Bad prediction	Advice: $p < B$ $\frac{(B-1)+B}{B}$	Advice: $p \geq B$ $\frac{(k-1)+B}{k-1}$

Ski Rental Algorithm 2

SKI-Rental with prediction (p, k)

If $p \geq B$

Keep renting until the k-th day // k is our "trust parameter"

else ($p < B$)

Keep renting until the B-th day S: Actual number of skiing days

	Truth: $S \geq B$ (OPT buy)	Truth: $S < B$ (OPT rent)
Good prediction	Advice: $p \geq B$ $\frac{(k-1)+B}{B}$	Advice: $p < B$ $\frac{S}{S}$
Bad prediction	Advice: $p < B$ $\frac{(B-1)+B}{B}$	Advice: $p \geq B$ $\frac{(k-1)+B}{k-1}$

Ski Rental Algorithm 2

SKI-Rental with prediction (p, k)

If $p \geq B$

Keep renting until the k-th day // k is our "trust parameter"

else ($p < B$)

Keep renting until the B-th day S: Actual number of skiing days

	Truth: $S \geq B$ (OPT buy)	Truth: $S < B$ (OPT rent)
 Good prediction	Advice: $p \geq B$ $\frac{(k-1)+B}{B} = 1 + \frac{k-1}{B}$	Advice: $p < B$ $\frac{S}{S} = 1$
 Bad prediction	Advice: $p < B$ $\frac{(B-1)+B}{B} = 2 + \frac{1}{B}$	Advice: $p \geq B$ $\frac{(k-1)+B}{k-1} = 1 + \frac{B}{k-1}$

Ski Rental Algorithm 2

SKI-Rental with prediction (p, k)

If $p \geq B$

Keep renting until the k-th day // k is our "trust parameter"

else ($p < B$)

Keep renting until the B-th day S: Actual number of skiing days

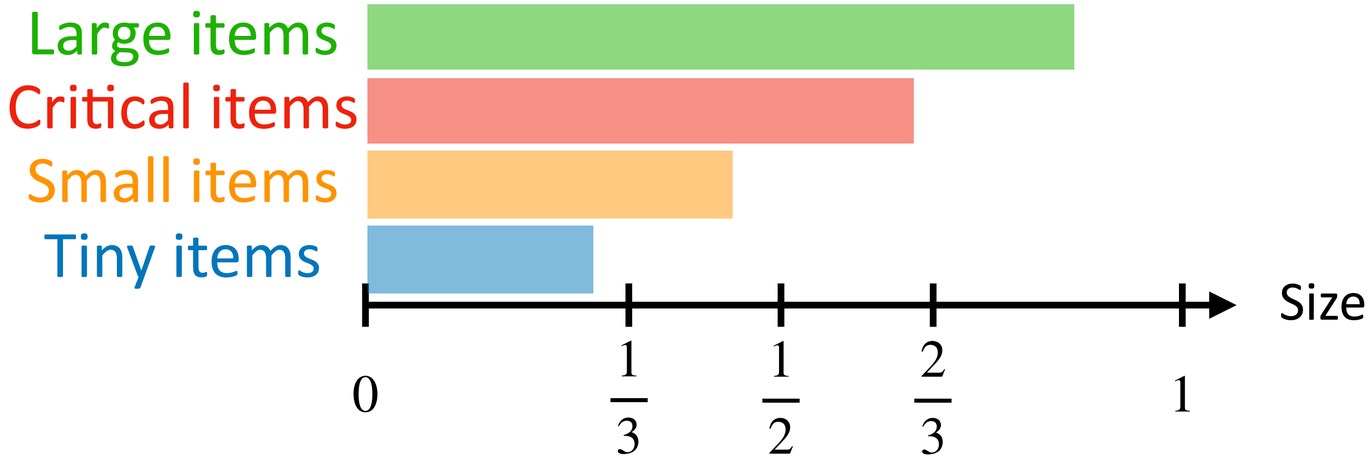
	Truth: $S \geq B$ (OPT buy)	Truth: $S < B$ (OPT rent)
 Good prediction *Consistency*	Advice: $p \geq B$ $\frac{(k-1)+B}{B} = 1 + \frac{k-1}{B}$	Advice: $p < B$ $\frac{S}{S} = 1$
 Bad prediction *Robustness*	Advice: $p < B$ $\frac{(B-1)+B}{B} = 2 + \frac{1}{B}$	Advice: $p \geq B$ $\frac{(k-1)+B}{k-1} = 1 + \frac{B}{k-1}$

What Happened

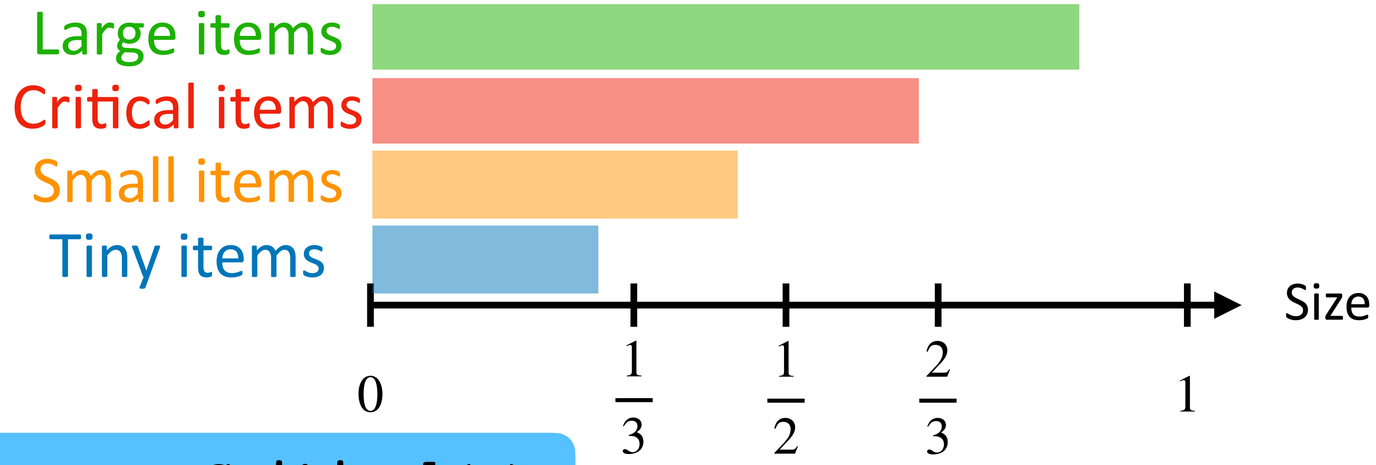
- Use a trust parameter to *partially* trust the prediction

Bin Packing with Advice

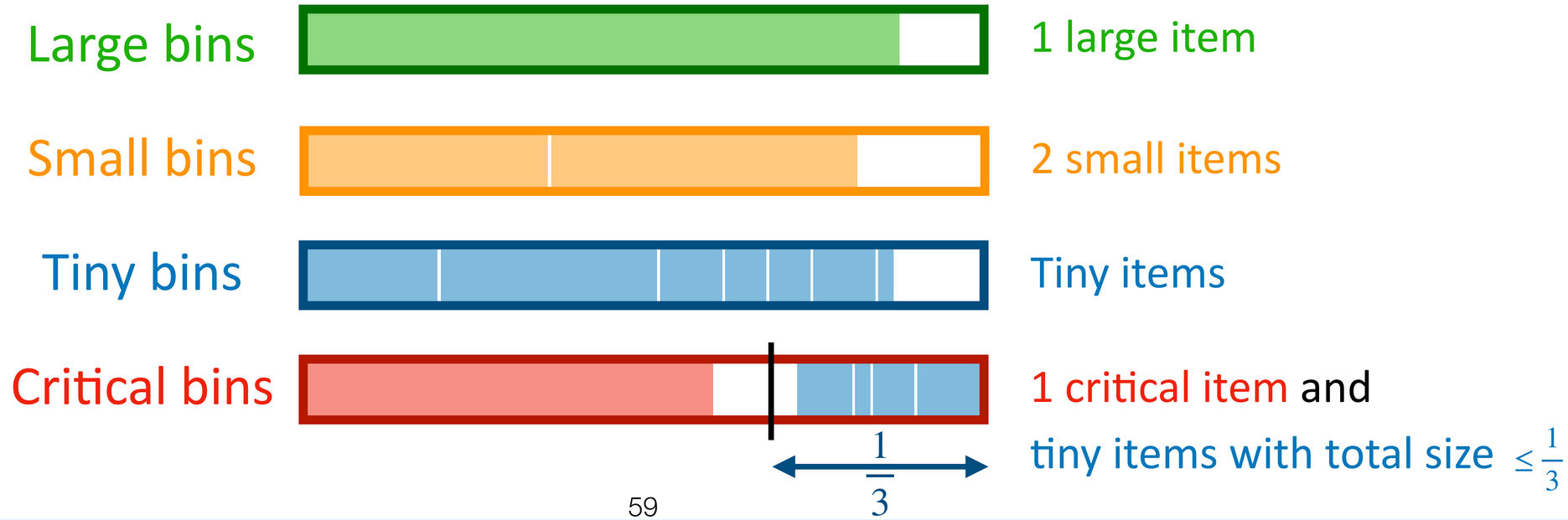
Bin Packing with Advice



Bin Packing with Advice



Algorithm: **`Reserve-Critical(`**c**`)`**



Reserve-Critical Algorithm

Reserve-Critical(c)

- Open c critical bins. In each critical bin, pack one critical item and tiny items with total size $\leq 1/6$
- Pack large items into large bins, small items into small bins
- When a tiny job arrives, pack it into a critical bin if possible (without making the total size of tiny items in that bin exceeds $1/6$); otherwise, pack it into a tiny bin by First-Fit

- Theorem: If the number of critical items is known, **Reserve-Critical** is 1.5 -competitive

Reserve-Critical algorithm guarantees that every non-critical bin is at least $\frac{2}{3}$-full

(may with a constant number of exceptions

Reserve-Critical Algorithm

Reserve-Critical(c)

- Open c critical bins. In each critical bin, pack one critical item and tiny items with total size $\leq 1/6$
- Pack large items into large bins, small items into small bins
- When a tiny job arrives, pack it into a critical bin if possible (without making the total size of tiny items in that bin exceeds $1/6$); otherwise, pack it into a tiny bin by First-Fit

- Theorem: If the number of critical items is known, **Reserve-Critical** is 1.5 -competitive

Reserve-Critical algorithm guarantees that every non-critical bin is at least $\frac{2}{3}$-full

(may with a constant number of exceptions

Reserve-Critical Algorithm

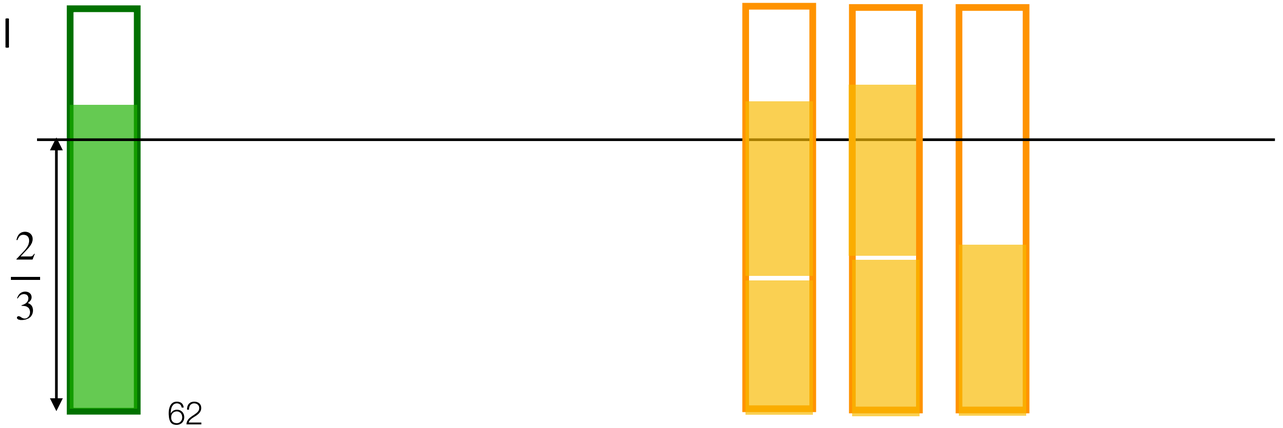
Reserve-Critical(c)

- Open c critical bins. In each critical bin, pack one critical item and tiny items with total size $\leq 1/6$
- Pack large items into large bins, small items into small bins
- When a tiny job arrives, pack it into a critical bin if possible (without making the total size of tiny items in that bin exceeds $1/6$); otherwise, pack it into a tiny bin by First-Fit

- Theorem: If the number of critical items is known, **Reserve-Critical** is 1.5 -competitive

Reserve-Critical algorithm guarantees that every non-critical bin is at least $\frac{2}{3}$-full

(may with a constant number of exceptions



Reserve-Critical Algorithm

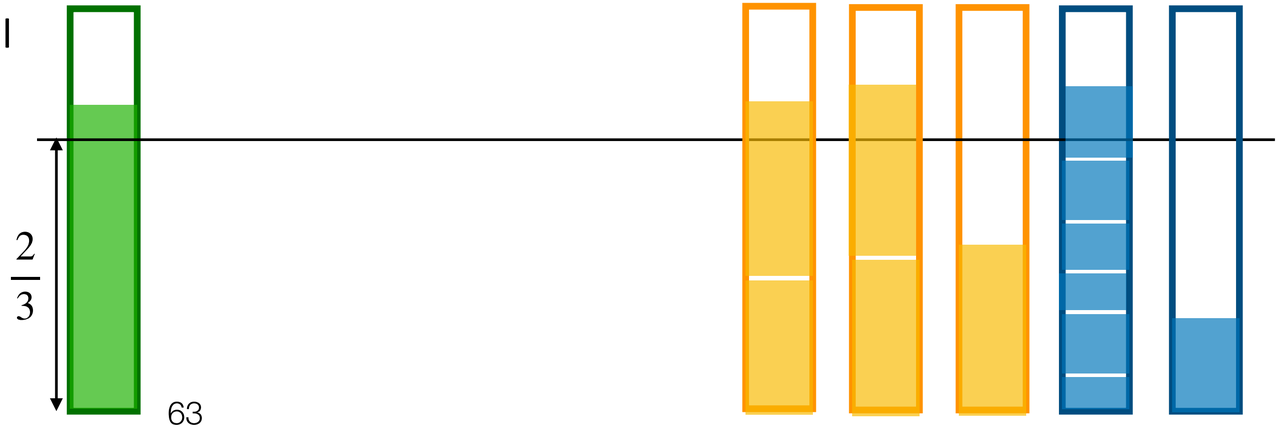
Reserve-Critical(c)

- Open c critical bins. In each critical bin, pack one critical item and tiny items with total size $\leq 1/6$
- Pack large items into large bins, small items into small bins
- When a tiny job arrives, pack it into a critical bin if possible (without making the total size of tiny items in that bin exceeds $1/6$); otherwise, pack it into a tiny bin by First-Fit

- Theorem: If the number of critical items is known, **Reserve-Critical** is 1.5 -competitive

Reserve-Critical algorithm guarantees that every non-critical bin is at least $\frac{2}{3}$-full

(may with a constant number of exceptions



Reserve-Critical Algorithm

Reserve-Critical(c)

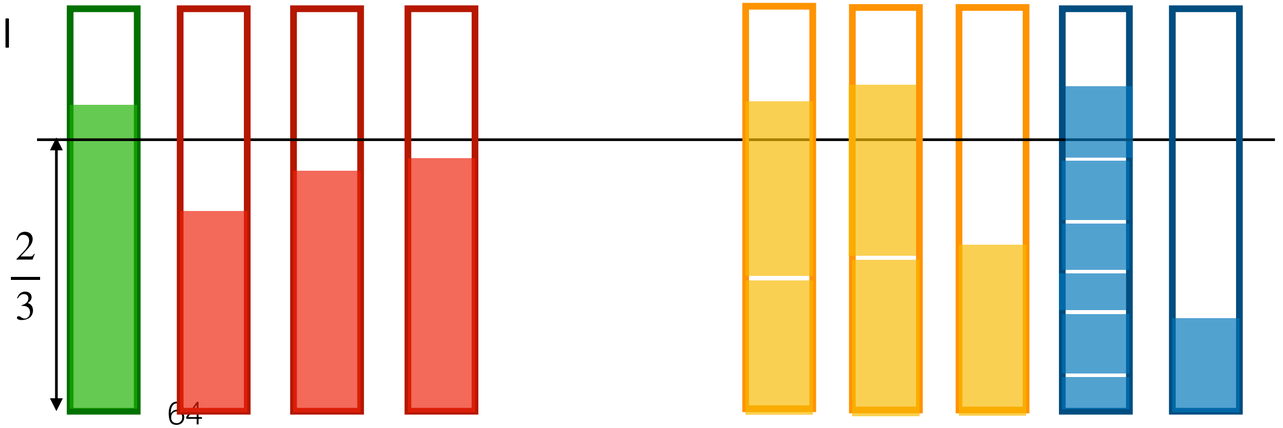
- Open c critical bins. In each critical bin, pack one critical item and tiny items with total size $\leq 1/6$
- Pack large items into large bins, small items into small bins
- When a tiny job arrives, pack it into a critical bin if possible (without making the total size of tiny items in that bin exceeds $1/6$); otherwise, pack it into a tiny bin by First-Fit

- Theorem: If the number of critical items is known, **Reserve-Critical** is 1.5 -competitive

Reserve-Critical algorithm guarantees that every non-critical bin is at least $\frac{2}{3}$-full

(may with a constant number of exceptions

For the critical bins, either they are all at least $\frac{2}{3}$-full, or the optimal also need all these critical bins



Reserve-Critical Algorithm

Reserve-Critical(c)

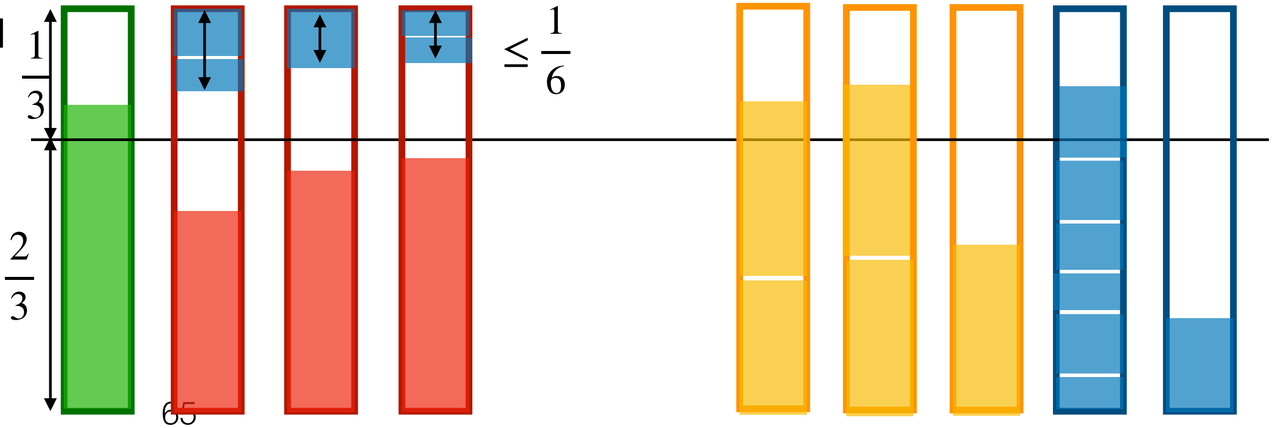
- Open c critical bins. In each critical bin, pack one critical item and tiny items with total size $\leq 1/6$
- Pack large items into large bins, small items into small bins
- When a tiny job arrives, pack it into a critical bin if possible (without making the total size of tiny items in that bin exceeds $1/6$); otherwise, pack it into a tiny bin by First-Fit

- Theorem: If the number of critical items is known, **Reserve-Critical** is 1.5 -competitive

Reserve-Critical algorithm guarantees that every non-critical bin is at least $\frac{2}{3}$-full

(may with a constant number of exceptions

For the critical bins, either they are all at least $\frac{2}{3}$-full, or the optimal also need all these critical bins



Robust-Reserve-Critical Algorithm

Robust-Reserve-Critical($r = \frac{c}{c+t}, \alpha$ **)** // c : number of critical bins, t : number of tiny bins, $\alpha \in [0,1]$

- Let $\beta = \{r, \alpha\}$
- When a critical item arrives, pack it into a critical bin
- When a tiny item arrives, pack it into an available critical bin
  - If there is no available critical bin, pack it into a tiny bin
    - If there is no tiny bin, open a new bin B
      - B is a critical bin if $\frac{c'}{c'+t'} < \beta$, otherwise, it is a tiny bin
- Large items and small items are packed into large bins and small bins respectively

Robust-Reserve-Critical Algorithm

Robust-Reserve-Critical($r = \frac{c}{c+t}, \alpha$ **)** // c : number of critical bins, t : number of tiny bins, $\alpha \in [0,1]$

- Let $\beta = \{r, \alpha\}$
- When a critical item arrives, pack it into a critical bin
- When a tiny item arrives, pack it into an available critical bin
  - If there is no available critical bin, pack it into a tiny bin
    - If there is no tiny bin, open a new bin B
      - B is a critical bin if $\frac{c'}{c'+t'} < \beta$, otherwise, it is a tiny bin
- Large items and small items are packed into large bins and small bins respectively

Prediction: the number of critical bins and the number of tiny bins

$$\frac{c}{c+t}$$

Robust-Reserve-Critical Algorithm

Robust-Reserve-Critical($r = \frac{c}{c+t}, \alpha$ **)** // c : number of critical bins, t : number of tiny bins, $\alpha \in [0,1]$

- Let $\beta = \{r, \alpha\}$
- When a critical item arrives, pack it into a critical bin
- When a tiny item arrives, pack it into an available critical bin
  - If there is no available critical bin, pack it into a tiny bin
    - If there is no tiny bin, open a new bin B
      - B is a critical bin if $\frac{c'}{c'+t'} < \beta$, otherwise, it is a tiny bin
- Large items and small items are packed into large bins and small bins respectively

Robust-Reserve-Critical Algorithm

Robust-Reserve-Critical($r = \frac{c}{c+t}, \alpha$ **)** // c : number of critical bins, t : number of tiny bins, $\alpha \in [0,1]$

- Let $\beta = \{r, \alpha\}$
- When a critical item arrives, pack it into a critical bin
- When a tiny item arrives, pack it into an available critical bin
  - If there is no available critical bin, pack it into a tiny bin
    - If there is no tiny bin, open a new bin B
      - B is a critical bin if $\frac{c'}{c'+t'} < \beta$, otherwise, it is a tiny bin
- Large items and small items are packed into large bins and small bins respectively

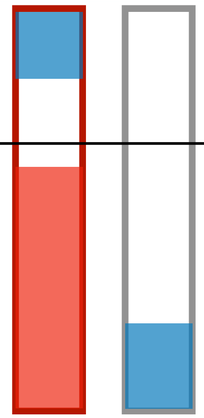
Robust-Reserve-Critical Algorithm

Robust-Reserve-Critical($r = \frac{c}{c+t}, \alpha$ **)** // c : number of critical bins, t : number of tiny bins, $\alpha \in [0,1]$

- Let $\beta = \{r, \alpha\}$
- When a critical item arrives, pack it into a critical bin
- When a tiny item arrives, pack it into an available critical bin
  - If there is no available critical bin, pack it into a tiny bin
    - If there is no tiny bin, open a new bin B
      - B is a critical bin if $\frac{c'}{c'+t'} < \beta$, otherwise, it is a tiny bin
- Large items and small items are packed into large bins and small bins respectively

$\beta = r = \frac{1}{2}$,

$\alpha = \frac{2}{3}$



Currently, $\frac{c'}{c'+t'} = \frac{1}{1+0}$

Robust-Reserve-Critical Algorithm

Robust-Reserve-Critical($r = \frac{c}{c+t}, \alpha$ **)** // c : number of critical bins, t : number of tiny bins, $\alpha \in [0,1]$

- Let $\beta = \{r, \alpha\}$
- When a critical item arrives, pack it into a critical bin
- When a tiny item arrives, pack it into an available critical bin
  - If there is no available critical bin, pack it into a tiny bin
    - If there is no tiny bin, open a new bin B
      - B is a critical bin if $\frac{c'}{c'+t'} < \beta$, otherwise, it is a tiny bin
- Large items and small items are packed into large bins and small bins respectively

$\beta = r = \frac{1}{2},$

$\alpha = \frac{2}{3}$

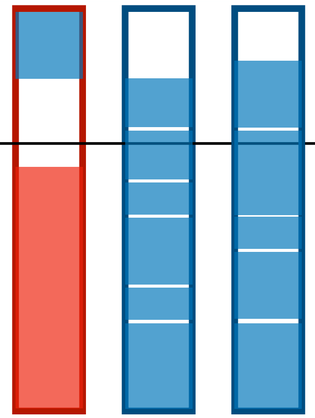
Robust-Reserve-Critical Algorithm

Robust-Reserve-Critical($r = \frac{c}{c+t}, \alpha$ **)** // c : number of critical bins, t : number of tiny bins, $\alpha \in [0,1]$

- Let $\beta = \{r, \alpha\}$
- When a critical item arrives, pack it into a critical bin
- When a tiny item arrives, pack it into an available critical bin
  - If there is no available critical bin, pack it into a tiny bin
    - If there is no tiny bin, open a new bin B
      - B is a critical bin if $\frac{c'}{c'+t'} < \beta$, otherwise, it is a tiny bin
- Large items and small items are packed into large bins and small bins respectively

$\beta = r = \frac{1}{2}$,

$\alpha = \frac{2}{3}$



Currently, $\frac{c'}{c'+t'} = \frac{1}{1+1} = \frac{1}{2}$

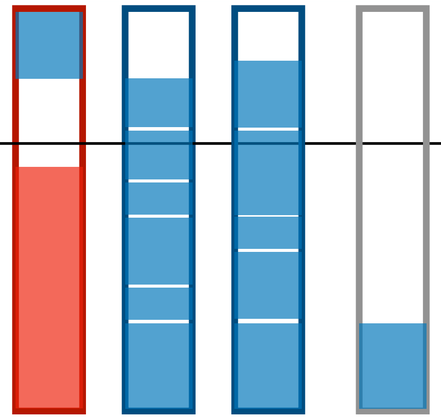
Robust-Reserve-Critical Algorithm

Robust-Reserve-Critical($r = \frac{c}{c+t}, \alpha$ **)** // c : number of critical bins, t : number of tiny bins, $\alpha \in [0,1]$

- Let $\beta = \{r, \alpha\}$
- When a critical item arrives, pack it into a critical bin
- When a tiny item arrives, pack it into an available critical bin
  - If there is no available critical bin, pack it into a tiny bin
    - If there is no tiny bin, open a new bin B
      - B is a critical bin if $\frac{c'}{c'+t'} < \beta$, otherwise, it is a tiny bin
- Large items and small items are packed into large bins and small bins respectively

$\beta = r = \frac{1}{2},$

$\alpha = \frac{2}{3}$



Currently, $\frac{c'}{c'+t'} = \frac{1}{1+2} = \frac{1}{3}$

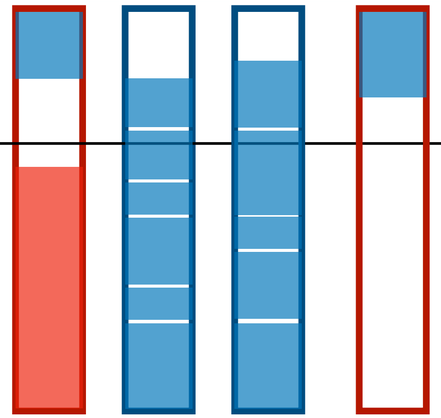
Robust-Reserve-Critical Algorithm

Robust-Reserve-Critical($r = \frac{c}{c+t}, \alpha$ **)** // c : number of critical bins, t : number of tiny bins, $\alpha \in [0,1]$

- Let $\beta = \{r, \alpha\}$
- When a critical item arrives, pack it into a critical bin
- When a tiny item arrives, pack it into an available critical bin
  - If there is no available critical bin, pack it into a tiny bin
    - If there is no tiny bin, open a new bin B
      - B is a critical bin if $\frac{c'}{c'+t'} < \beta$, otherwise, it is a tiny bin
- Large items and small items are packed into large bins and small bins respectively

$\beta = r = \frac{1}{2}$,

$\alpha = \frac{2}{3}$



Currently, $\frac{c'}{c'+t'} = \frac{1}{1+2} = \frac{1}{3}$

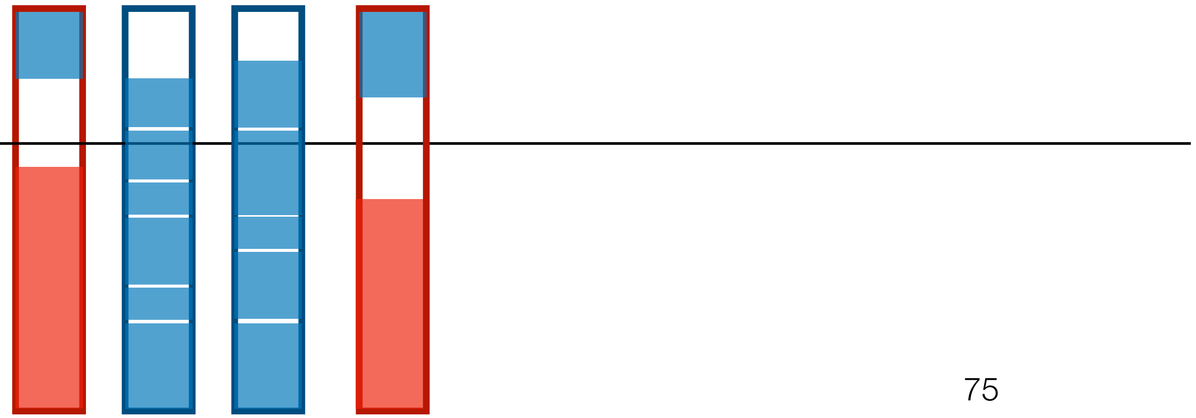
Robust-Reserve-Critical Algorithm

Robust-Reserve-Critical($r = \frac{c}{c+t}, \alpha$ **)** // c : number of critical bins, t : number of tiny bins, $\alpha \in [0,1]$

- Let $\beta = \{r, \alpha\}$
- When a critical item arrives, pack it into a critical bin
- When a tiny item arrives, pack it into an available critical bin
  - If there is no available critical bin, pack it into a tiny bin
    - If there is no tiny bin, open a new bin B
      - B is a critical bin if $\frac{c'}{c'+t'} < \beta$, otherwise, it is a tiny bin
- Large items and small items are packed into large bins and small bins respectively

$\beta = r = \frac{1}{2},$

$\alpha = \frac{2}{3}$



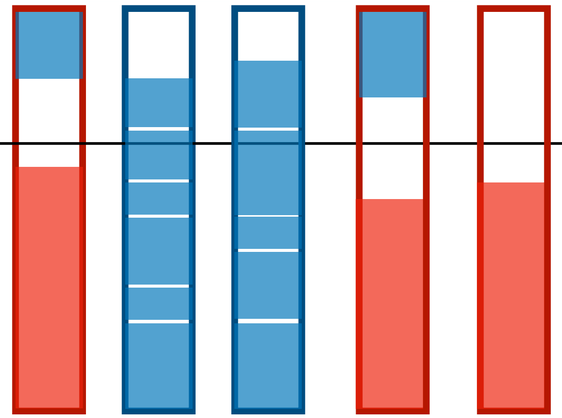
Robust-Reserve-Critical Algorithm

Robust-Reserve-Critical($r = \frac{c}{c+t}, \alpha$ **)** // c : number of critical bins, t : number of tiny bins, $\alpha \in [0,1]$

- Let $\beta = \{r, \alpha\}$
- When a critical item arrives, pack it into a critical bin
- When a tiny item arrives, pack it into an available critical bin
  - If there is no available critical bin, pack it into a tiny bin
    - If there is no tiny bin, open a new bin B
      - B is a critical bin if $\frac{c'}{c'+t'} < \beta$, otherwise, it is a tiny bin
- Large items and small items are packed into large bins and small bins respectively

$\beta = r = \frac{1}{2}$,

$\alpha = \frac{2}{3}$



Robust-Reserve-Critical always keep the ratio $\frac{c'}{c'+t'}$ close to β

Robust-Reserve-Critical Algorithm

Robust-Reserve-Critical($r = \frac{c}{c+t}, \alpha$ **)** // c : number of critical bins, t : number of tiny bins, $\alpha \in [0,1]$

- Let $\beta = \{r, \alpha\}$
- When a critical item arrives, pack it into a critical bin
- When a tiny item arrives, pack it into an available critical bin
  - If there is no available critical bin, pack it into a tiny bin
    - If there is no tiny bin, open a new bin B
      - B is a critical bin if $\frac{c'}{c'+t'} < \beta$, otherwise, it is a tiny bin
- Large items and small items are packed into large bins and small bins respectively

Prediction: the number of critical bins and the number of tiny bins

$$\frac{c}{c+t}$$

Every critical bin has one critical item

We my open too many critical bins

Robust-Reserve-Critical Algorithm

- Lemma: If every critical bin has one critical item, **Robust-Reserve-Critical** is at most $1.5+\frac{1-\beta}{4-3\beta}$-competitive

By the algorithm, we can make sure that each bin is full enough

- Lemma: If there are some critical bins have no critical item, **Robust-Reserve-Critical** is at most $1.5+\frac{9\beta}{8-6\beta}$-competitive

The critical bin without a critical item must be opened because of a tiny item

$\Rightarrow$ at the time the tiny item arrives, the number of critical bins is low

Robust-Reserve-Critical Algorithm

- Theorem: If the prediction on $r = \frac{c}{c+t}$ is correct, **Robust-Reserve-Critical** is at most $1.5+\frac{1-\alpha}{4-3\alpha}$-competitive. Otherwise, **Robust-Reserve-Critical** is at most $1.5+\max\{\frac{1}{4}, \frac{9\alpha}{8-6\alpha}\}$-competitive.

What Happened

- Use extra information to bound the optimal cost from below

Prediction Error η

- Absolute error $\eta_1 = |p - S|$
- Squared error $\eta_2 = |p - S|^2$
- Classification error $\eta_c = 1$ if $p \neq S$

Ski Rental Algorithms Revisit: $\eta_1 = |\, p - S \,|$

- $\eta_1 = |\, p - S \,|$

Algorithm 3

If $p \geq B$ then **Buy** on the $\lceil B \cdot (1-\alpha) + \varepsilon \rceil$-th day

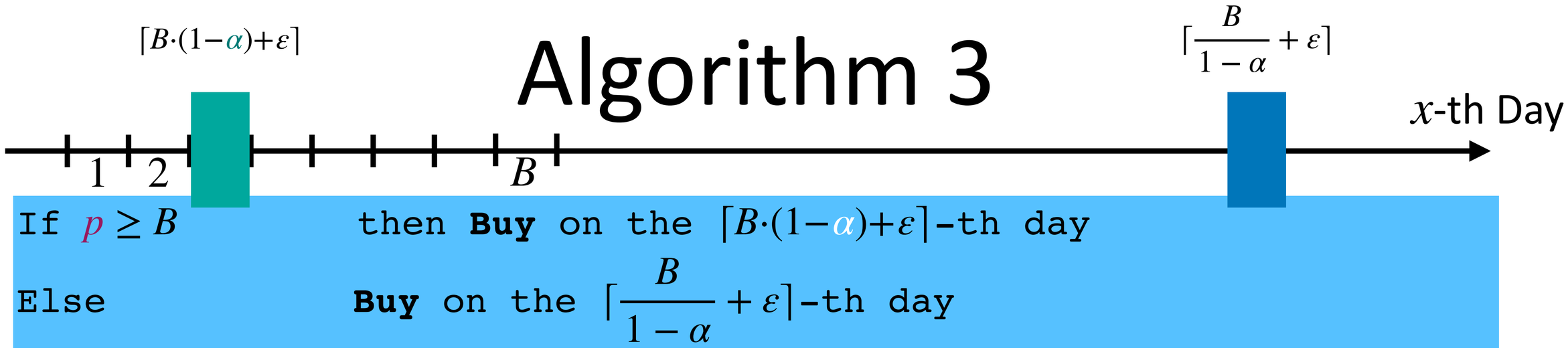
Else **Buy** on the $\lceil \frac{B}{1-\alpha} + \varepsilon \rceil$-th day

Algorithm 3

If $p \geq B$ then **Buy** on the $\lceil B\cdot(1-\alpha)+\varepsilon \rceil$-th day

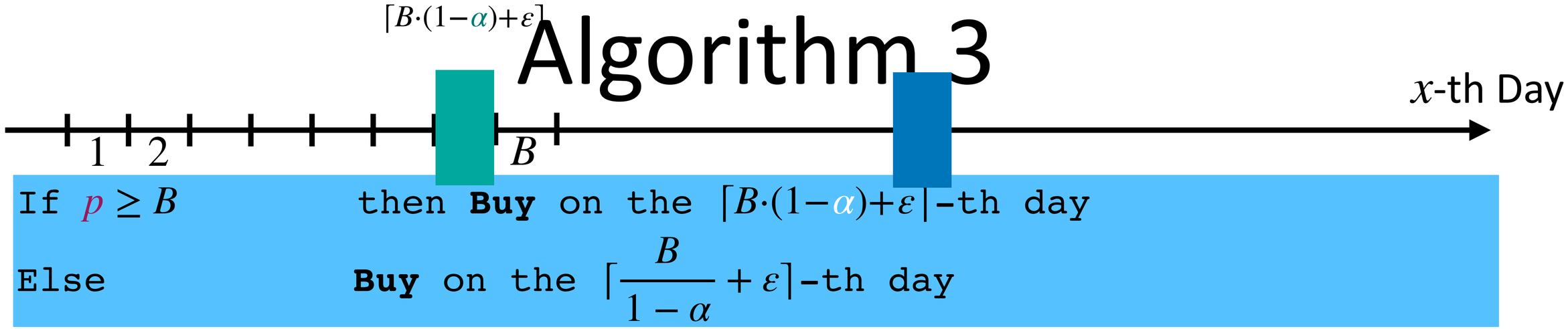
Else **Buy** on the $\lceil \frac{B}{1-\alpha}+\varepsilon \rceil$-th day

Algorithm 3



Big α

Algorithm 3



If $p \geq B$ then **Buy** on the $\lceil B\cdot(1-\alpha)+\varepsilon \rceil$-th day

Else **Buy** on the $\lceil \frac{B}{1-\alpha} + \varepsilon \rceil$-th day

Small α

Algorithm 3

If $p \geq B$ then **Buy** on the $\lceil B\cdot(1-\alpha)+\varepsilon \rceil$-th day

Else **Buy** on the $\lceil \frac{B}{1-\alpha}+\varepsilon \rceil$-th day

	Truth: $S \geq B$ (OPT buy)	Truth: $S < B$ (OPT rent)
Good prediction	Advice: $p \geq B$ $\frac{B(1-\alpha)+B}{B} = 2-\alpha$	
Bad prediction		

η_1

Algorithm 3

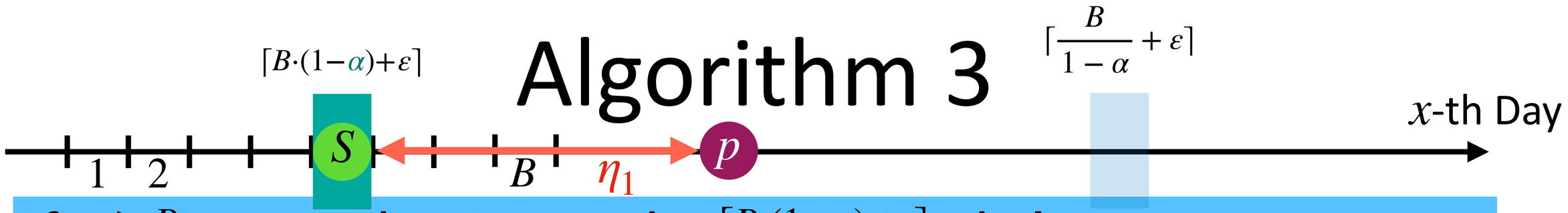
If $p \geq B$ then **Buy** on the $\lceil B\cdot(1-\alpha)+\varepsilon \rceil$-th day

Else **Buy** on the $\lceil \frac{B}{1-\alpha}+\varepsilon \rceil$-th day

	Truth: $S \geq B$ (OPT buy)	Truth: $S < B$ (OPT rent)
Good prediction	Advice: $p \geq B$ $\frac{B(1-\alpha)+B}{B} = 2-\alpha$	Advice: $p < B$ $\frac{S}{S} = 1$
Bad prediction		

η_1

Algorithm 3

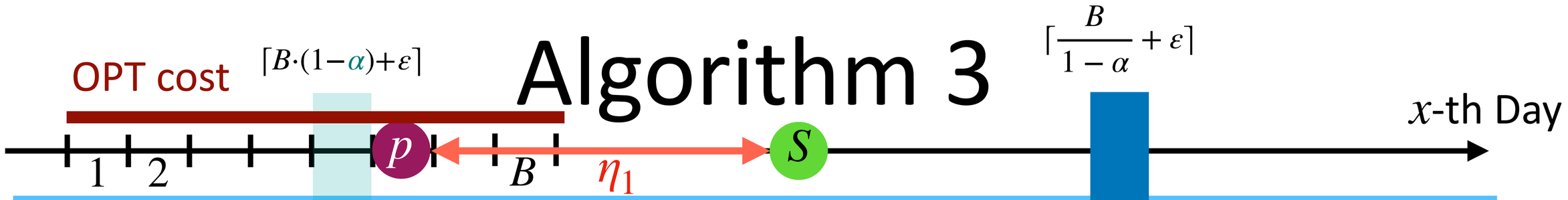


If $p \geq B$ then **Buy** on the $\lceil B\cdot(1-\alpha)+\varepsilon \rceil$-th day

Else **Buy** on the $\lceil \frac{B}{1-\alpha}+\varepsilon \rceil$-th day

	Truth: $S \geq B$ (OPT buy)	Truth: $S < B$ (OPT rent)
 Good prediction	Advice: $p \geq B$ $\frac{B(1-\alpha)+B}{B} = 2-\alpha$	Advice: $p < B$ $\frac{S}{S} = 1$
 Bad prediction		Advice: $p \geq B \Rightarrow \eta_1 = p - S$ $\Rightarrow B \leq \eta_1 + S = \eta_1 + \text{OPT}$ ALG buys $\text{ALG} \leq B(1-\alpha)+B$ $\leq (2-\alpha)(\text{OPT}+\eta_1)$

Algorithm 3



If $p \geq B$ then **Buy** on the $\lceil B\cdot(1-\alpha)+\varepsilon \rceil$-th day

Else **Buy** on the $\lceil \frac{B}{1-\alpha}+\varepsilon \rceil$-th day

	Truth: $S \geq B$ (OPT buy)	Truth: $S < B$ (OPT rent)
Good prediction	Advice: $p \geq B$ $\frac{B(1-\alpha)+B}{B} = 2-\alpha$	Advice: $p < B$ $\frac{S}{S} = 1$
Bad prediction	Advice: $p < B \Rightarrow \eta_1 = S-p \geq S-B$ If $S < \frac{B}{1-\alpha}$, ALG rents $\Rightarrow$ ALG $= S = p+\eta_1 \leq$ OPT $+\eta_1$	Advice: $p \geq B \Rightarrow \eta_1 = p-S$ $\Rightarrow B \leq \eta_1 + S = \eta_1 +$ OPT ALG buys ALG $\leq B(1-\alpha)+B$ $\leq (2-\alpha)(\text{OPT}+\eta_1)$

Algorithm 3

If $p \geq B$ then **Buy** on the $\lceil B\cdot(1-\alpha)+\varepsilon \rceil$-th day

Else **Buy** on the $\lceil \frac{B}{1-\alpha}+\varepsilon \rceil$-th day

	Truth: $S \geq B$ (OPT buy)	Truth: $S < B$ (OPT rent)
Good prediction	Advice: $p \geq B$ $\frac{B(1-\alpha)+B}{B} = 2-\alpha$	Advice: $p < B$ $\frac{S}{S} = 1$
Bad prediction	Advice: $p < B \Rightarrow \eta_1 = S-p \geq S-B$ If S $\geq \frac{B}{1-\alpha}$, ALG buys ($\eta_1 \geq \frac{B}{1-\alpha} - B$) $\Rightarrow$ ALG $\leq B + \frac{B}{1-\alpha} \leq$ OPT $+ \frac{\eta_1}{\alpha}$	Advice: $p \geq B \Rightarrow \eta_1 = p-S$ $\Rightarrow B \leq \eta_1 + S = \eta_1 +$ OPT ALG buys ALG $\leq B(1-\alpha)+B$ $\leq (2-\alpha)($OPT $+ \eta_1)$

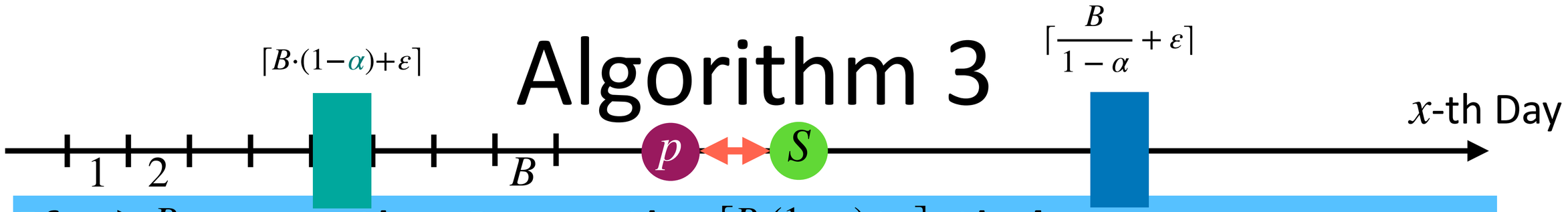
Algorithm 3

If $p \geq B$ then **Buy** on the $\lceil B\cdot(1-\alpha)+\varepsilon \rceil$-th day

Else **Buy** on the $\lceil \frac{B}{1-\alpha}+\varepsilon \rceil$-th day

	Truth: $S \geq B$ (OPT buy)	Truth: $S < B$ (OPT rent)
 Good prediction	Advice: $p \geq B$ $\frac{B(1-\alpha)+B}{B} = 2-\alpha$	Advice: $p < B$ $\frac{S}{S} = 1$
 Bad prediction	Advice: $p < B \Rightarrow \eta_1 = S-p \geq S-B$ If $S < \frac{B}{1-\alpha}$, ALG rents $\Rightarrow$ ALG $= S = p+\eta_1 \leq$ OPT $+\eta_1$ If S $\geq \frac{B}{1-\alpha}$, ALG buys ($\eta_1 \geq \frac{B}{1-\alpha}-B$) $\Rightarrow$ ALG $\leq B+\frac{B}{1-\alpha} \leq$ OPT $+\frac{\eta_1}{\alpha}$	Advice: $p \geq B \Rightarrow \eta_1 = p-S$ $\Rightarrow B \leq \eta_1 + S = \eta_1 +$ OPT ALG buys ALG $\leq B(1-\alpha)+B$ $\leq (2-\alpha)($OPT $+\eta_1)$

Algorithm 3



If $p \geq B$ then **Buy** on the $\lceil B\cdot(1-\alpha)+\varepsilon \rceil$-th day

Else **Buy** on the $\lceil \frac{B}{1-\alpha}+\varepsilon \rceil$-th day

	Truth: $S \geq B$ (OPT buy)	Truth: $S < B$ (OPT rent)
Good prediction	Advice: $p \geq B$ $\frac{B(1-\alpha)+B}{B} = 2-\alpha$	Advice: $p < B$ $\frac{S}{S} = 1$
Bad prediction	Advice: $p < B \Rightarrow \eta_1 = S-p \geq S-B$ If $S < \frac{B}{1-\alpha}$, ALG rents $\Rightarrow$ ALG $= S = p+\eta_1 \leq$ OPT $+\eta_1$ If S $\geq \frac{B}{1-\alpha}$, ALG buys ($\eta_1 \geq \frac{B}{1-\alpha}-B$) $\Rightarrow$ ALG $\leq B+\frac{B}{1-\alpha} \leq$ OPT $+\frac{\eta_1}{\alpha}$	Advice: $p \geq B \Rightarrow \eta_1 = p-S$ $\Rightarrow B \leq \eta_1 + S = \eta_1 +$ OPT ALG buys ALG $\leq B(1-\alpha)+B$ $\leq (2-\alpha)($OPT $+\eta_1)$

Machine-Learned Advice

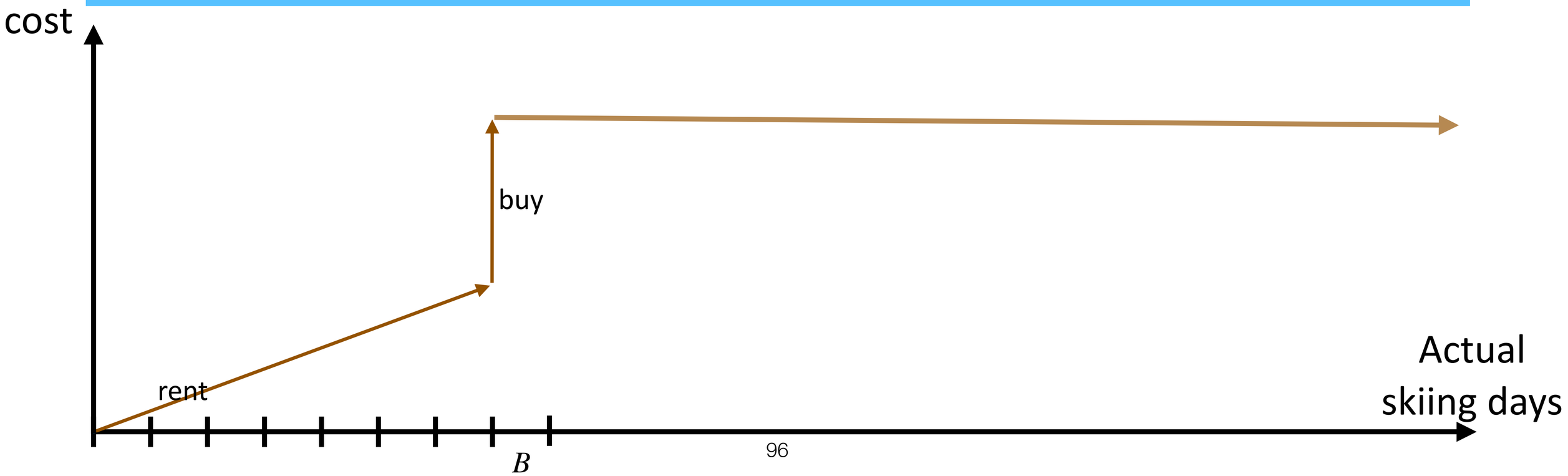
SKI-Rental with prediction (p, k)

If $p \geq B$

Keep renting until the k-th day // k is our "trust parameter"

else ($p < B$)

Keep renting until the B-th day



Machine-Learned Advice

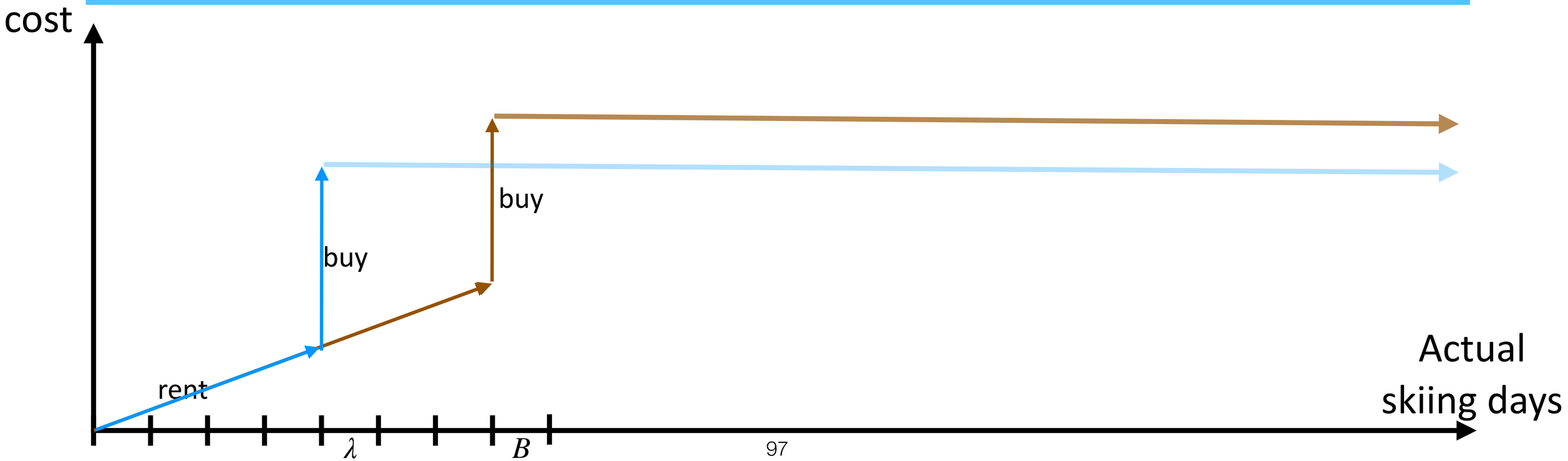
SKI-Rental with prediction (p, k)

If $p \geq B$

Keep renting until the k-th day // k is our "trust parameter"

else ($p < B$)

Keep renting until the B-th day



Machine-Learned Advice

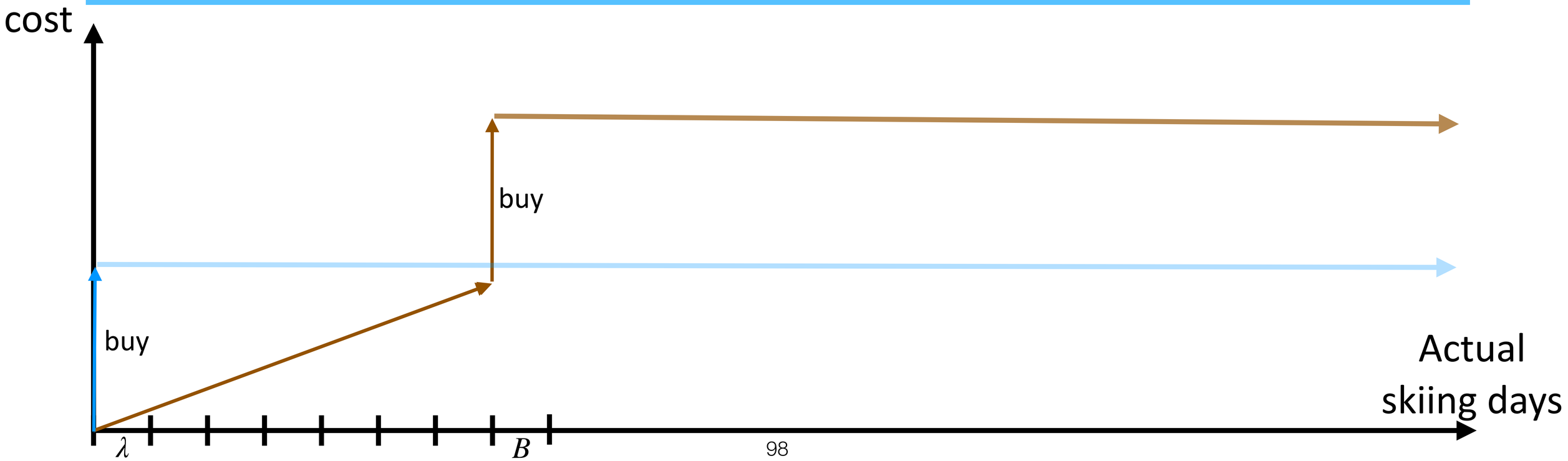
SKI-Rental with prediction (p, k)

If $p \geq B$

Keep renting until the k-th day // k is our "trust parameter"

else ($p < B$)

Keep renting until the B-th day



Machine-Learned Advice

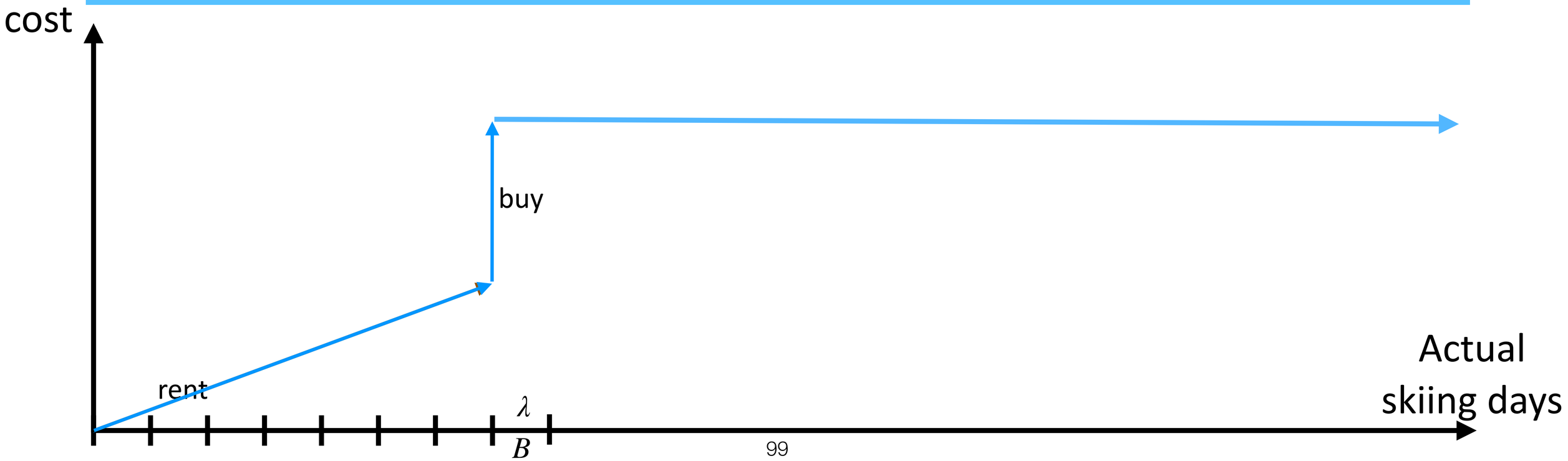
SKI-Rental with prediction (p, k)

If $p \geq B$

Keep renting until the k-th day // k is our "trust parameter"

else ($p < B$)

Keep renting until the B-th day



Machine-Learned Advice

SKI-Rental with prediction (p, k)

If $p \geq B$

Keep renting until the k-th day // k is our "trust parameter"

else ($p < B$)

Keep renting until the B-th day

Machine-Learned Advice

SKI-Rental with prediction (p, k)

If $p \geq B$

Keep renting until the k-th day // k is our "trust parameter"

else ($p < B$)

Keep renting until the B-th day S: Actual number of skiing days

Truth: $S \geq B$ (OPT buy)	Truth: $S < B$ (OPT rent)

Machine-Learned Advice

SKI-Rental with prediction (p, k)

If $p \geq B$

Keep renting until the k-th day // k is our "trust parameter"

else ($p < B$)

Keep renting until the B-th day S: Actual number of skiing days

	Truth: $S \geq B$ (OPT buy)	Truth: $S < B$ (OPT rent)
 Good advice		
 Bad advice		

Machine-Learned Advice

SKI-Rental with prediction (p, k)

If $p \geq B$

Keep renting until the k-th day // k is our "trust parameter"

else ($p < B$)

Keep renting until the B-th day S: Actual number of skiing days

	Truth: $S \geq B$ (OPT buy)	Truth: $S < B$ (OPT rent)
 Good advice	Advice: $p \geq B$	Advice: $p < B$
 Bad advice		

Machine-Learned Advice

SKI-Rental with prediction (p, k)

If $p \geq B$

Keep renting until the k-th day // k is our "trust parameter"

else ($p < B$)

Keep renting until the B-th day S: Actual number of skiing days

	Truth: $S \geq B$ (OPT buy)	Truth: $S < B$ (OPT rent)
 Good advice	Advice: $p \geq B$	Advice: $p < B$
 Bad advice	Advice: $p < B$	Advice: $p \geq B$

Machine-Learned Advice

SKI-Rental with prediction (p, k)

If $p \geq B$

Keep renting until the k-th day // k is our "trust parameter"

else ($p < B$)

Keep renting until the B-th day S: Actual number of skiing days

	Truth: $S \geq B$ (OPT buy)	Truth: $S < B$ (OPT rent)
 Good advice	Advice: $p \geq B$ $\frac{(k-1)+B}{B}$	Advice: $p < B$
 Bad advice	Advice: $p < B$	Advice: $p \geq B$

Machine-Learned Advice

SKI-Rental with prediction (p, k)

If $p \geq B$

Keep renting until the k-th day // k is our "trust parameter"

else ($p < B$)

Keep renting until the B-th day S: Actual number of skiing days

	Truth: $S \geq B$ (OPT buy)	Truth: $S < B$ (OPT rent)
 Good advice	Advice: $p \geq B$ $\frac{(k-1)+B}{B}$	Advice: $p < B$ $\frac{d}{d}$
 Bad advice	Advice: $p < B$	Advice: $p \geq B$

Machine-Learned Advice

SKI-Rental with prediction (p, k)

If $p \geq B$

Keep renting until the k-th day // k is our "trust parameter"

else ($p < B$)

Keep renting until the B-th day S: Actual number of skiing days

	Truth: $S \geq B$ (OPT buy)	Truth: $S < B$ (OPT rent)
 Good advice	Advice: $p \geq B$ $\frac{(k-1)+B}{B}$	Advice: $p < B$ $\frac{d}{d}$
 Bad advice	Advice: $p < B$ $\frac{(B-1)+B}{B}$	Advice: $p \geq B$

Machine-Learned Advice

SKI-Rental with prediction (p, k)

If $p \geq B$

Keep renting until the k-th day // k is our "trust parameter"

else ($p < B$)

Keep renting until the B-th day S: Actual number of skiing days

	Truth: $S \geq B$ (OPT buy)	Truth: $S < B$ (OPT rent)
 Good advice	Advice: $p \geq B$ $\frac{(k-1)+B}{B}$	Advice: $p < B$ $\frac{d}{d}$
 Bad advice	Advice: $p < B$ $\frac{(B-1)+B}{B}$	Advice: $p \geq B$ $\frac{(k-1)+B}{d}$

Machine-Learned Advice

SKI-Rental with prediction (p, k)

If $p \geq B$

Keep renting until the k-th day // k is our "trust parameter"

else ($p < B$)

Keep renting until the B-th day S: Actual number of skiing days

	Truth: $S \geq B$ (OPT buy)	Truth: $S < B$ (OPT rent)
 Good advice	Advice: $p \geq B$ $\frac{(k-1)+B}{B}$	Advice: $p < B$ $\frac{d}{d}$
 Bad advice	Advice: $p < B$ $\frac{(B-1)+B}{B}$	Advice: $p \geq B$ $\frac{(k-1)+B}{d}$

Machine-Learned Advice

SKI-Rental with prediction (p, k)

If $p \geq B$

Keep renting until the k-th day // k is our "trust parameter"

else ($p < B$)

Keep renting until the B-th day S: Actual number of skiing days

	Truth: $S \geq B$ (OPT buy)	Truth: $S < B$ (OPT rent)
 Good advice *Consistency*	Advice: $p \geq B$ $\frac{(k-1)+B}{B}$	Advice: $p < B$ $\frac{d}{d}$
 Bad advice *Robustness*	Advice: $p < B$ $\frac{(B-1)+B}{B}$	Advice: $p \geq B$ $\frac{(k-1)+B}{d}$

Randomization May Help

- If we allow the online algorithm to use randomness, smaller competitive ratios may be attainable
- A *randomized online algorithm* ALG is c-competitive against an oblivious adversary if for every oblivious adversary I_o,

$$\mathbf{E}[\text{ALG}(I_o)] \leq c \cdot \text{OPT}(I_o) + \alpha$$

  - That is, in the algorithm ALG there is some random factor; it can make random choices

RANDOMIZED-SKI-RENTAL

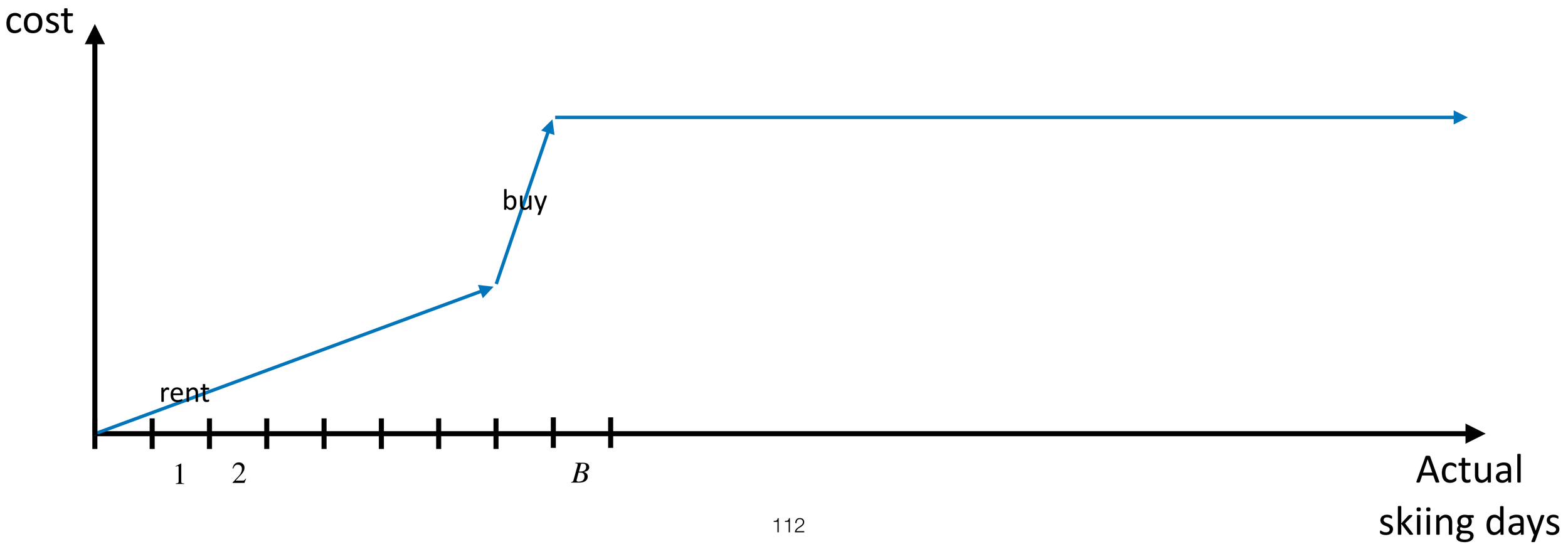
Keep renting until the $(\phi - 1) \cdot B$-th day // ϕ is the golden ratio 1.618…

On the $(\phi - 1) \cdot B$-th day, flip a coin

If it is *head*, buy the ski

Otherwise, keep renting until the B-th day

Randomized Ski-rental Algorithm



Randomized Ski-rental Algorithm

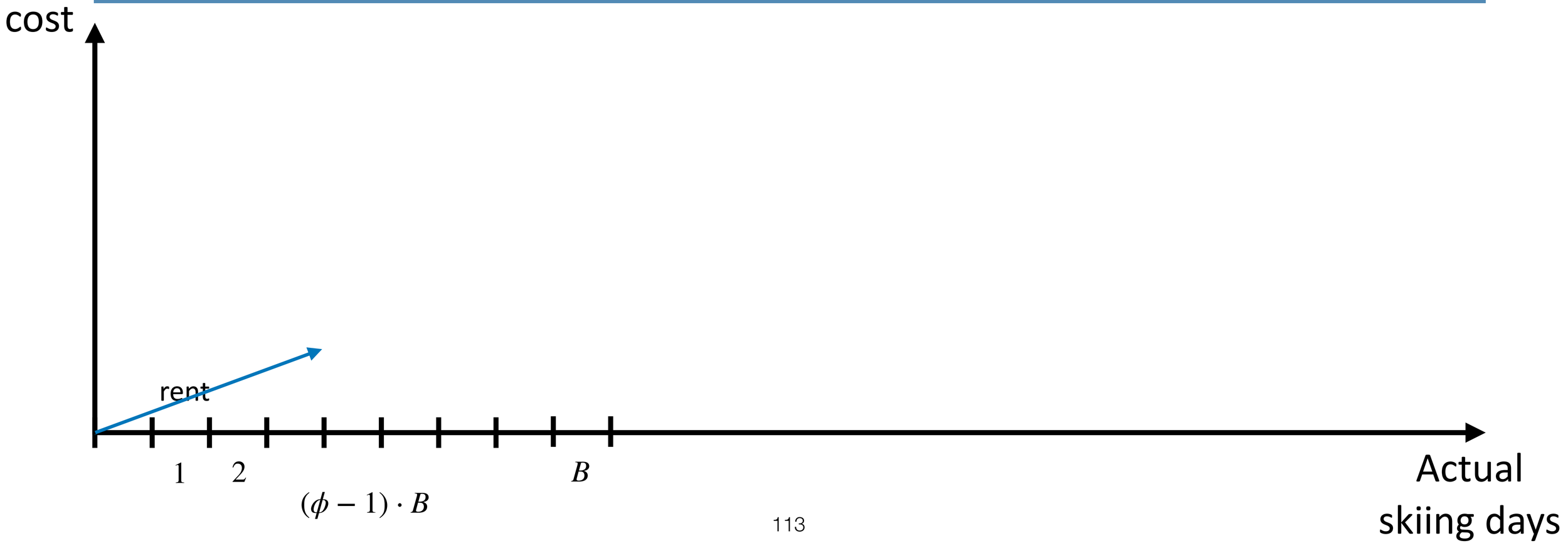
RANDOMIZED-SKI-RENTAL

Keep renting until the $(\phi - 1) \cdot B$-th day // ϕ is the golden ratio 1.618…

On the $(\phi - 1) \cdot B$-th day, flip a coin

If it is *head*, buy the ski

Otherwise, keep renting until the B-th day



Randomized Ski-rental Algorithm

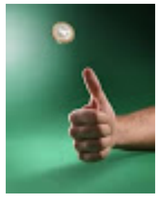
RANDOMIZED-SKI-RENTAL

Keep renting until the $(\phi - 1) \cdot B$-th day // ϕ is the golden ratio 1.618…

On the $(\phi - 1) \cdot B$-th day, flip a coin

If it is *head*, buy the ski

Otherwise, keep renting until the B-th day



Randomized Ski-rental Algorithm

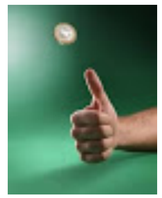
RANDOMIZED-SKI-RENTAL

Keep renting until the $(\phi - 1) \cdot B$-th day // ϕ is the golden ratio 1.618…

On the $(\phi - 1) \cdot B$-th day, flip a coin

If it is *head*, buy the ski

Otherwise, keep renting until the B-th day



Randomized Ski-rental Algorithm

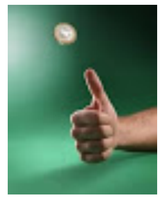
RANDOMIZED-SKI-RENTAL

Keep renting until the $(\phi - 1) \cdot B$-th day // ϕ is the golden ratio 1.618…

On the $(\phi - 1) \cdot B$-th day, flip a coin

If it is *head*, buy the ski

Otherwise, keep renting until the B-th day



Randomized Ski-rental Algorithm

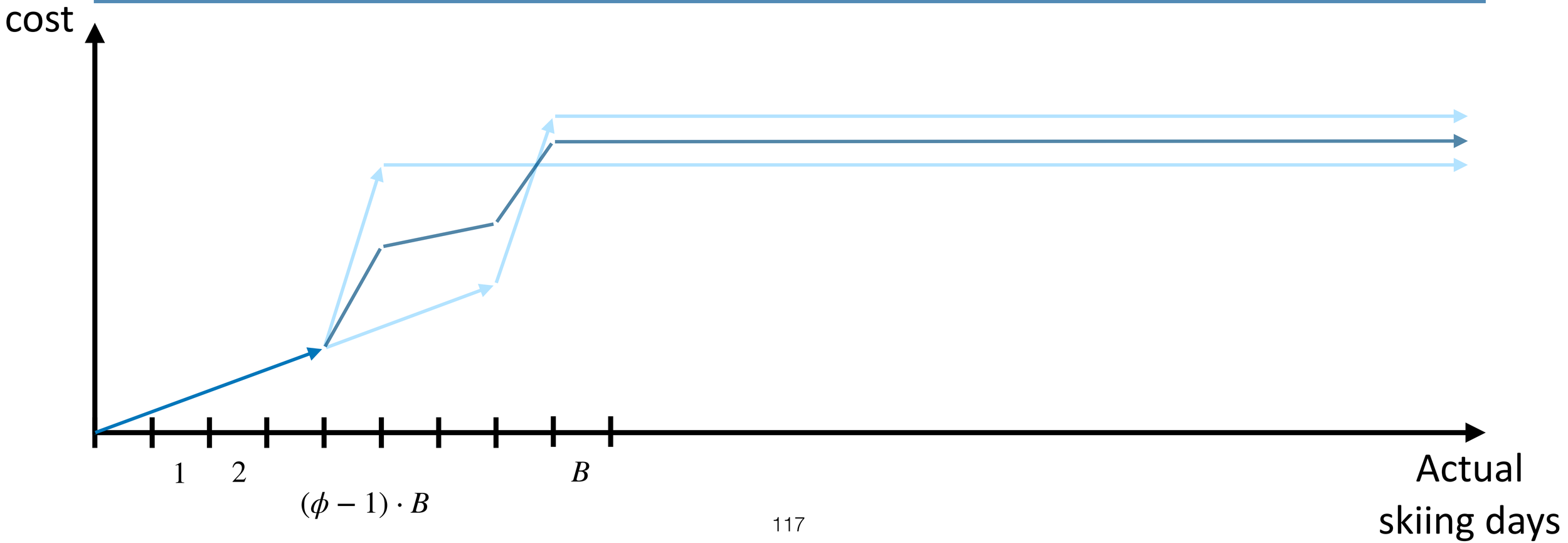
RANDOMIZED-SKI-RENTAL

Keep renting until the $(\phi - 1) \cdot B$-th day // ϕ is the golden ratio 1.618…

On the $(\phi - 1) \cdot B$-th day, flip a coin

If it is *head*, buy the ski

Otherwise, keep renting until the B-th day



Randomized Ski-rental Algorithm

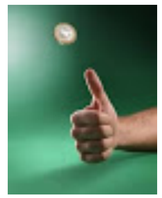
RANDOMIZED-SKI-RENTAL

Keep renting until the $(\phi - 1) \cdot B$-th day // ϕ is the golden ratio 1.618…

On the $(\phi - 1) \cdot B$-th day, flip a coin

If it is *head*, buy the ski

Otherwise, keep renting until the B-th day



cost

1 2 $(\phi - 1) \cdot B$ B

Actual skiing days

Randomized Ski-rental Algorithm

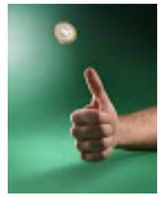
RANDOMIZED-SKI-RENTAL

Keep renting until the $(\phi - 1) \cdot B$-th day // ϕ is the golden ratio 1.618…

On the $(\phi - 1) \cdot B$-th day, flip a coin

If it is *head*, buy the ski

Otherwise, keep renting until the B-th day



Randomized Ski-rental Algorithm

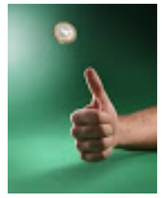
RANDOMIZED-SKI-RENTAL

Keep renting until the $(\phi - 1) \cdot B$-th day // ϕ is the golden ratio 1.618…

On the $(\phi - 1) \cdot B$-th day, flip a coin

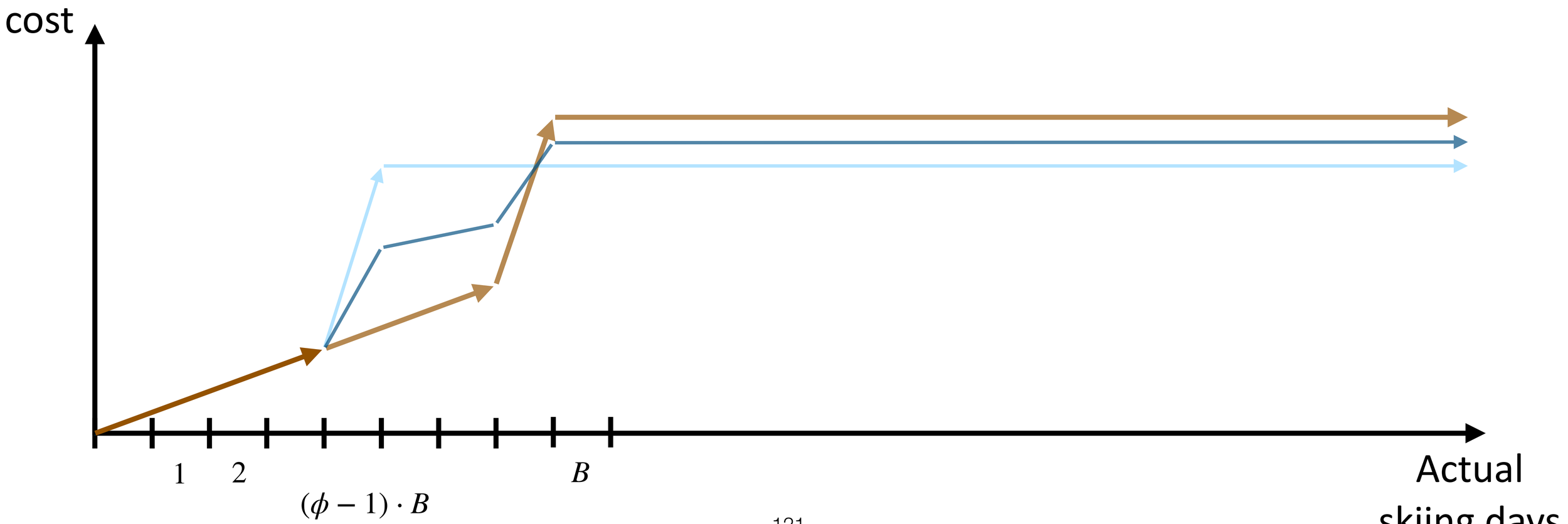
If it is *head*, buy the ski

Otherwise, keep renting until the B-th day



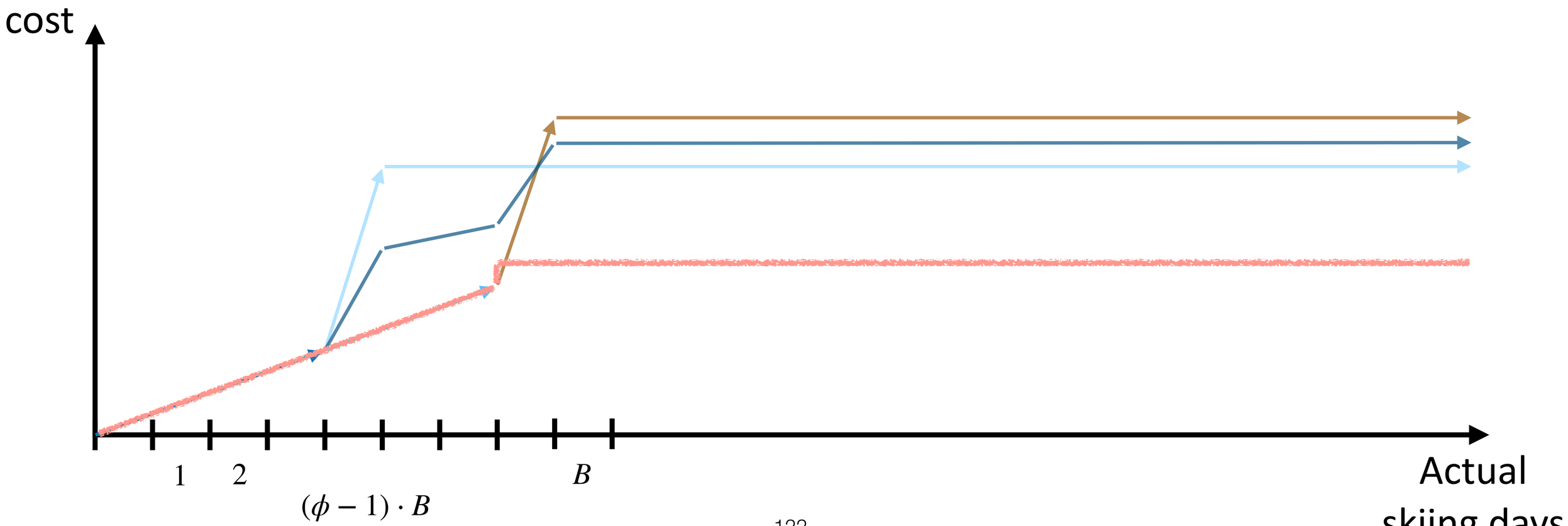
Randomized Ski-rental Algorithm

Deterministic Ski-Rental ALG competitive ratio:



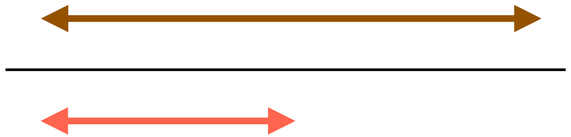
Randomized Ski-rental Algorithm

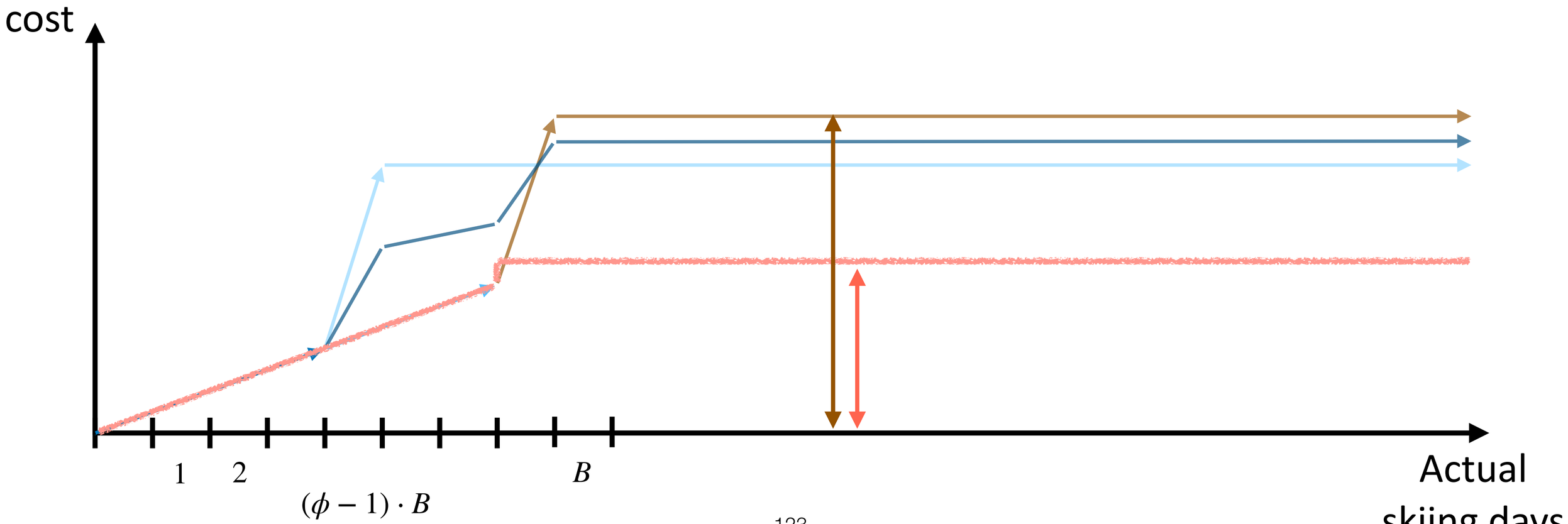
Deterministic Ski-Rental ALG competitive ratio:



Randomized Ski-rental Algorithm

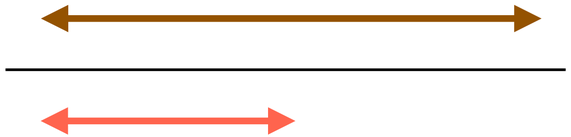
Deterministic Ski-Rental ALG competitive ratio:



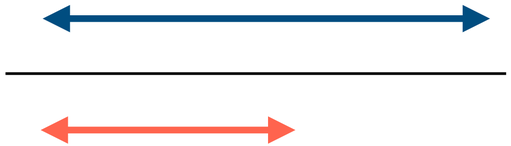


Randomized Ski-rental Algorithm

Deterministic Ski-Rental ALG competitive ratio:



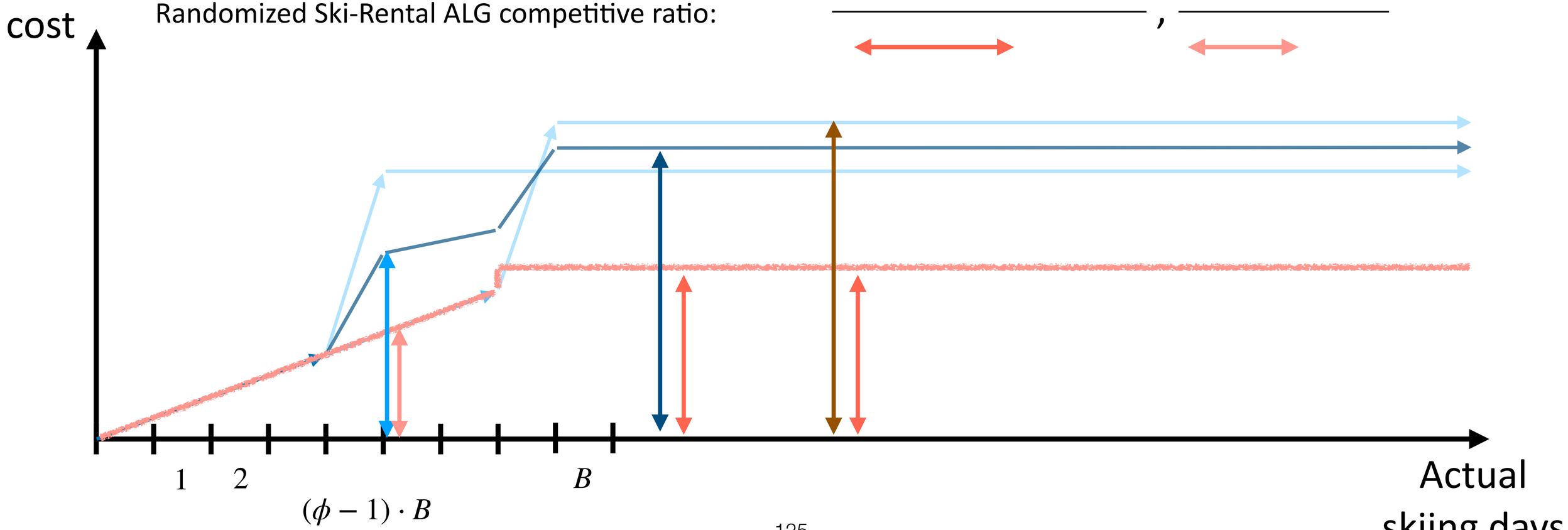
Randomized Ski-Rental ALG competitive ratio:



Randomized Ski-rental Algorithm

Deterministic Ski-Rental ALG competitive ratio: $\frac{2}{1}$

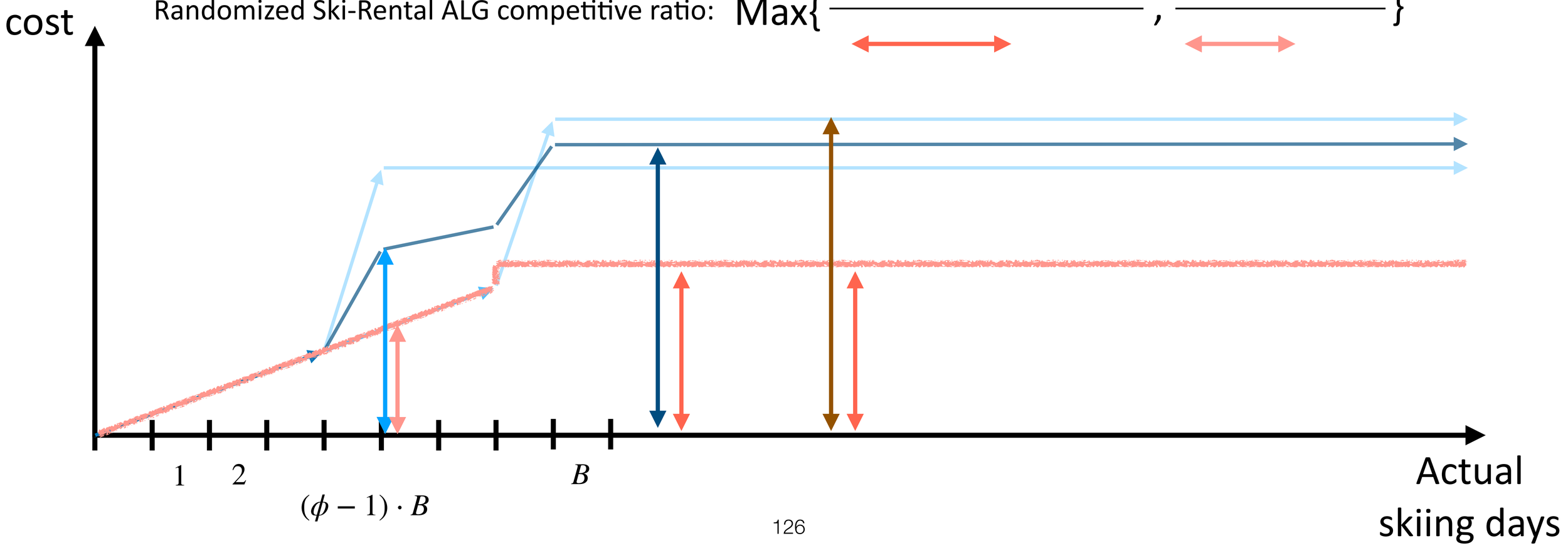
Randomized Ski-Rental ALG competitive ratio: $\frac{2}{1}, \frac{2}{1}$



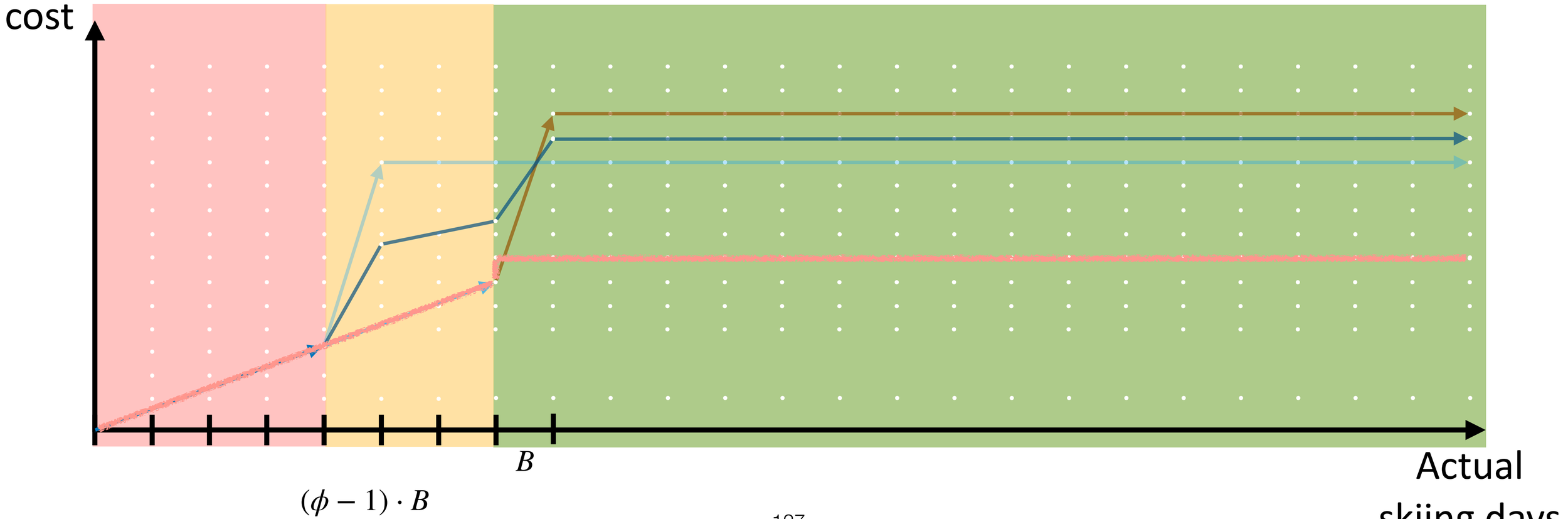
Randomized Ski-rental Algorithm

Deterministic Ski-Rental ALG competitive ratio: $\frac{\longleftrightarrow}{\longleftrightarrow}$

Randomized Ski-Rental ALG competitive ratio: $\text{Max}\{\frac{\longleftrightarrow}{\longleftrightarrow}, \frac{\longleftrightarrow}{\longleftrightarrow}\}$

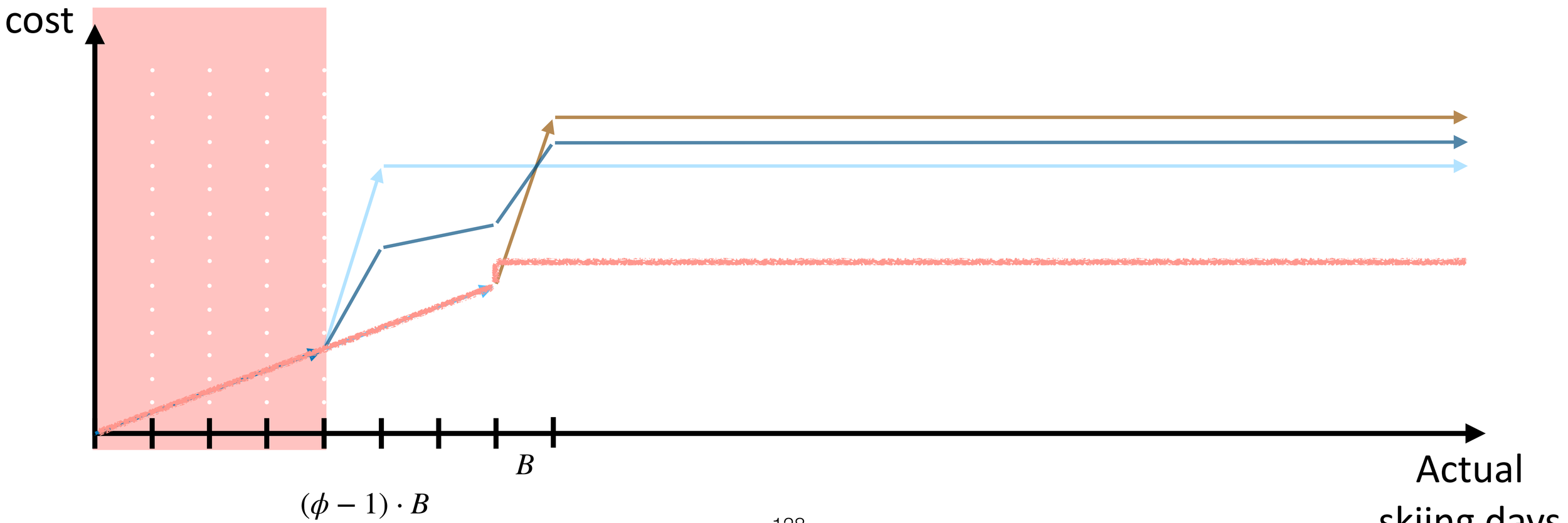


Randomized Ski-rental Algorithm



Randomized Ski-rental Algorithm

If #actual skiing days $< (\phi - 1) \cdot B$: ratio = 1

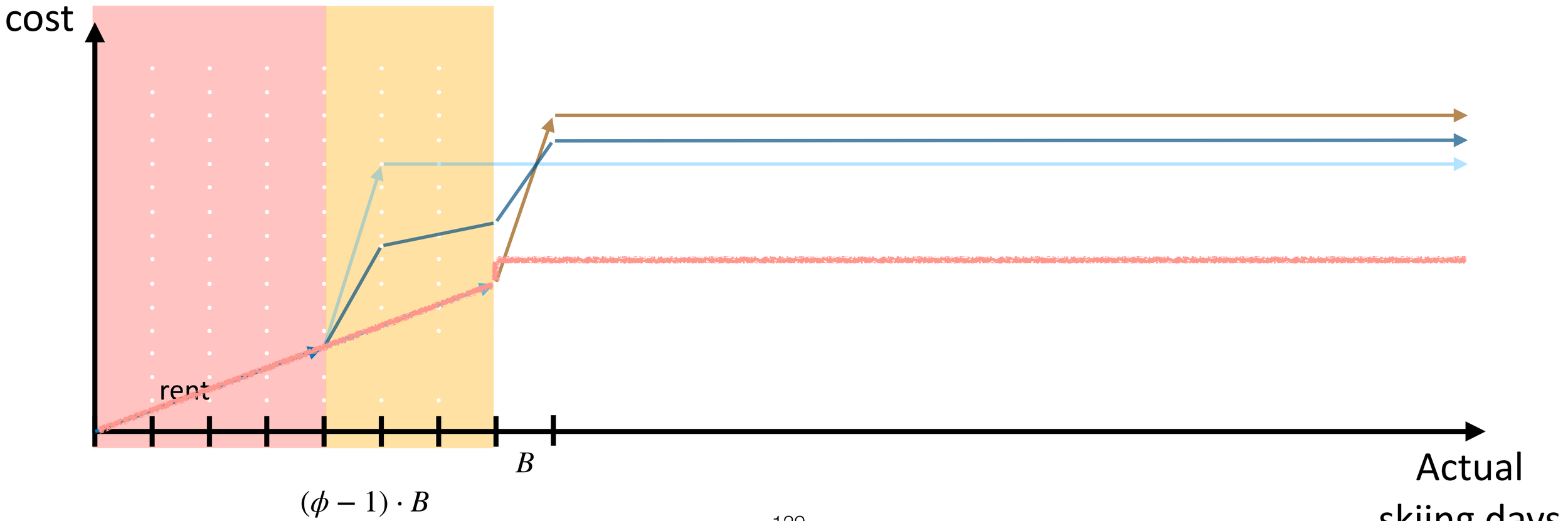


Randomized Ski-rental Algorithm

If #actual skiing days $\in [(\phi - 1) \cdot B, B)$:

ratio = $\frac{1}{2} \cdot \frac{(\phi - 1) \cdot B - 1 + B}{D} + \frac{1}{2} \cdot \frac{D}{D} \leq 1.809$

If #actual skiing days $< (\phi - 1) \cdot B$: ratio = 1



Randomized Ski-rental Algorithm

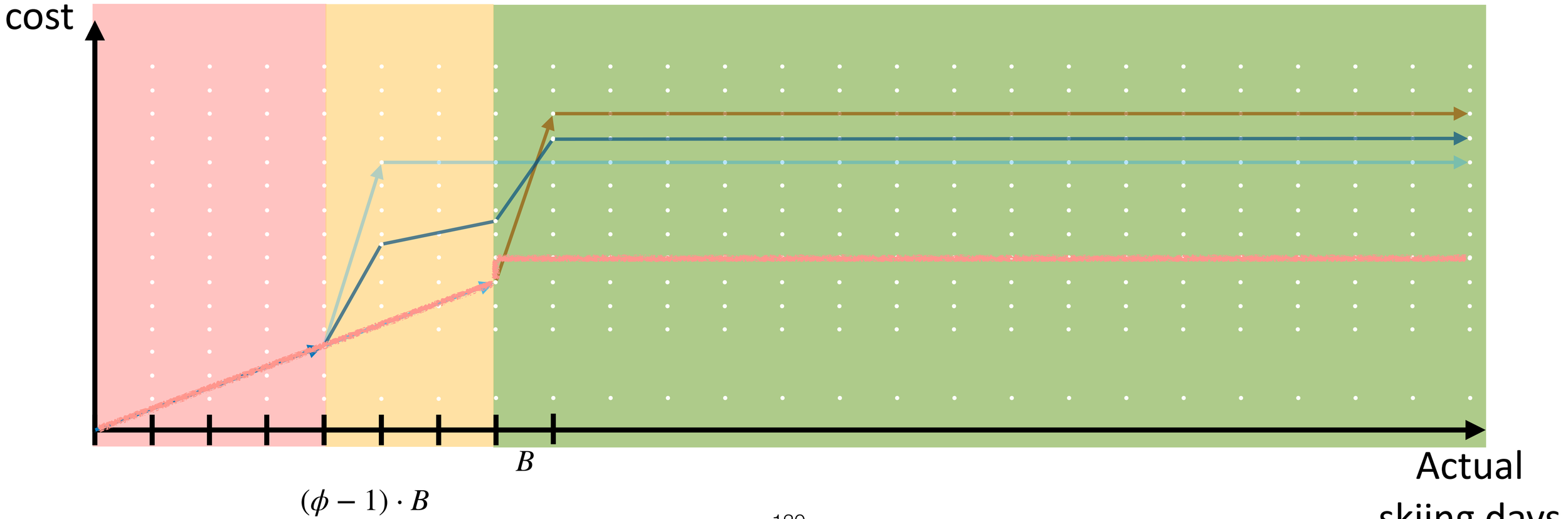
If #actual skiing days $\in [(\phi - 1) \cdot B, B)$:

ratio = $\frac{1}{2} \cdot \frac{(\phi - 1) \cdot B - 1 + B}{D} + \frac{1}{2} \cdot \frac{D}{D} \leq 1.809$

If #actual skiing days $< (\phi - 1) \cdot B$: ratio = 1

If #actual skiing days $\geq B$:

ratio = $\frac{1}{2} \cdot \frac{(\phi - 1) \cdot B - 1 + B}{B} + \frac{1}{2} \cdot \frac{(B - 1) + B)}{B} = 1.809$



Outline

- Research cycle of online algorithms — Using **BinPacking** as an example
  - BinPacking and FirstFit algorithm
  - FirstFit is 2-competitive
  - There is no 1.333-competitive deterministic online algorithm for BinPacking
  - FirstFit is at least 1.667-competitive
  - FirstFit $\leq 1.7 \cdot \mathrm{OPT} + 2$
- Different variants of an online problem — Using **MachineMinimization** as an example
  - There is no $O(1)$-competitive algorithm for the general case
  - For special cases, MachineMinimization admits better competitive algorithms
- Randomization may helps
- Machine-learned advices

Machine-Learned Advice

- Online algorithms deal with optimization under uncertainty (about future input)

Machine-Learned Advice

- Online algorithms deal with optimization under uncertainty (about future input)
  - How if the future information can be predicted or learned by machine-learning?

Machine-Learned Advice

- Online algorithms deal with optimization under uncertainty (about future input)
  - How if the future information can be predicted or learned by machine-learning?
    - Example: weather forecast

Machine-Learned Advice

- Online algorithms deal with optimization under uncertainty (about future input)
  - How if the future information can be predicted or learned by machine-learning?
    - Example: weather forecast
    - These predictions or learned information may not be 100% correct

Machine-Learned Advice

- Online algorithms deal with optimization under uncertainty (about future input)
  - How if the future information can be predicted or learned by machine-learning?
    - Example: weather forecast
    - These predictions or learned information may not be 100% correct
      - Completely trust the predictions may be a disaster

Machine-Learned Advice

- Online algorithms deal with optimization under uncertainty (about future input)
  - How if the future information can be predicted or learned by machine-learning?
    - Example: weather forecast
    - These predictions or learned information may not be 100% correct
      - Completely trust the predictions may be a disaster
      - How can we design online algorithms with this kind of (maybe) untrusted advice?

Machine-Learned Advice

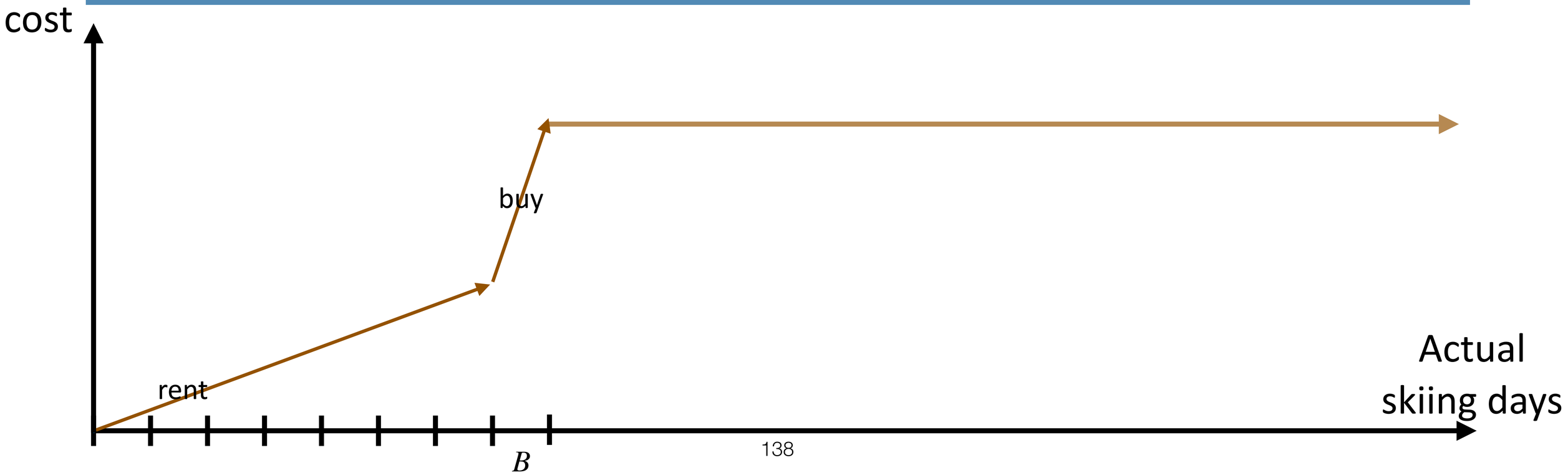
SKI-Rental with advice (a)

If $a \geq B$

Keep renting until the λ-th day // λ is our "trust parameter"

Otherwise ($a < B$)

Keep renting until the B-th day



Machine-Learned Advice

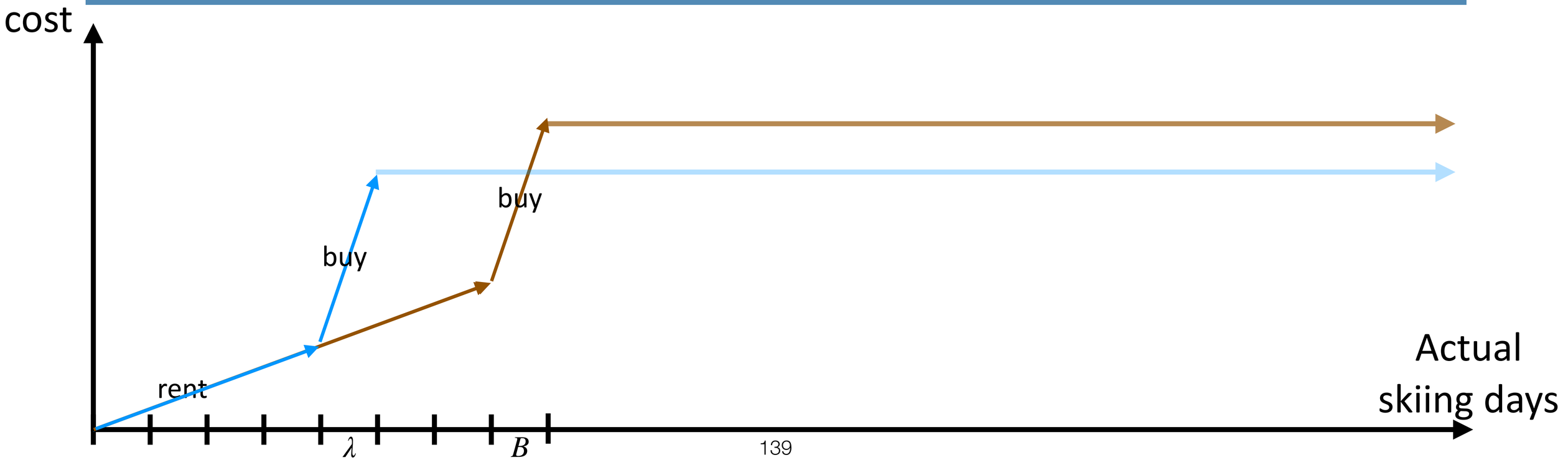
SKI-Rental with advice (a)

If $a \geq B$

Keep renting until the λ-th day // λ is our "trust parameter"

Otherwise ($a < B$)

Keep renting until the B-th day



Machine-Learned Advice

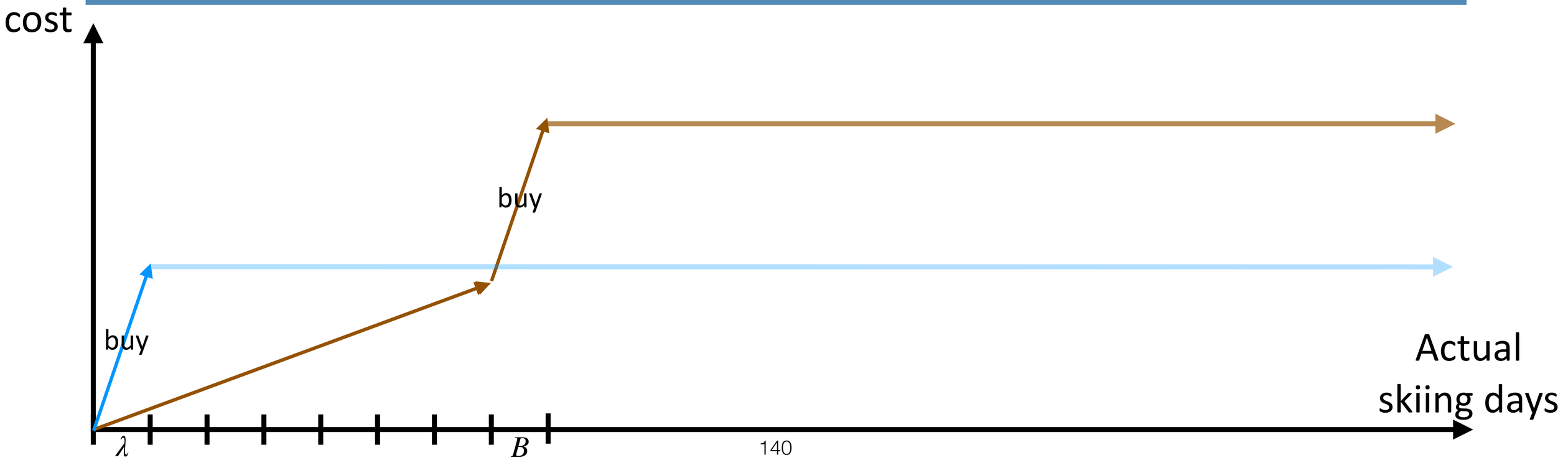
SKI-Rental with advice (a)

If $a \geq B$

Keep renting until the λ-th day // λ is our "trust parameter"

Otherwise ($a < B$)

Keep renting until the B-th day



Machine-Learned Advice

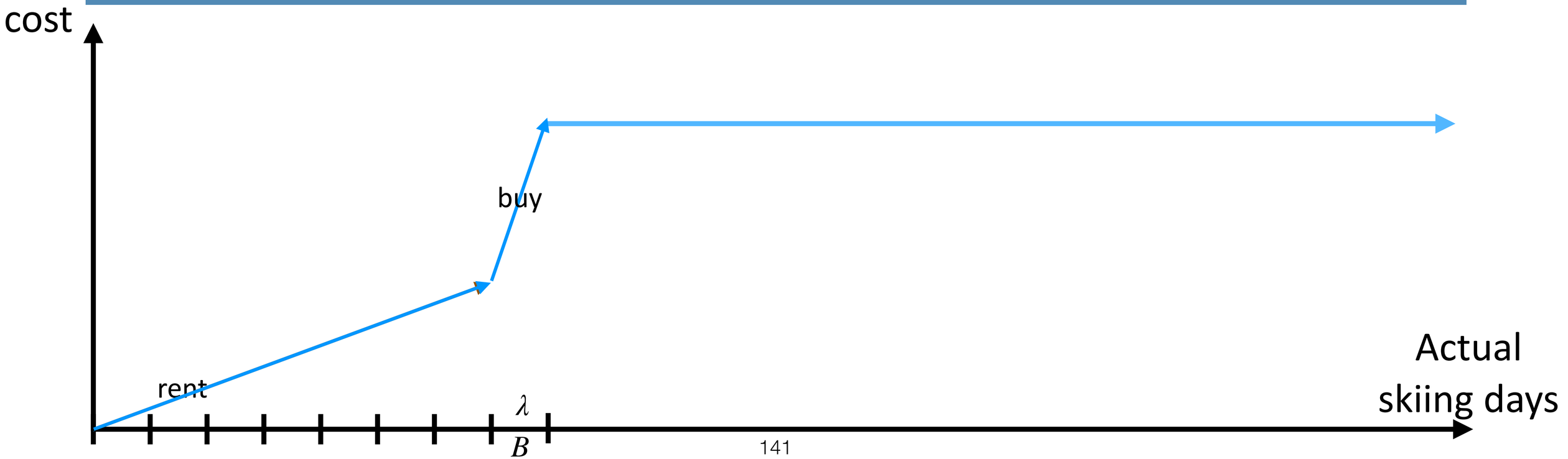
SKI-Rental with advice (a)

If $a \geq B$

Keep renting until the λ-th day // λ is our "trust parameter"

Otherwise ($a < B$)

Keep renting until the B-th day



November 1 (Tue)

November 3 (Thu)

Both okay