First-Come First-Served (FCFS) for Online Slot Allocation (OSA) and Huffman Coding

— SODA 2014 —

Monik Khare

yellowpages.com

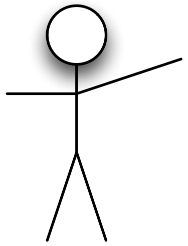
Claire Mathieu

Ecole Normale Superieure

Neal E. Young

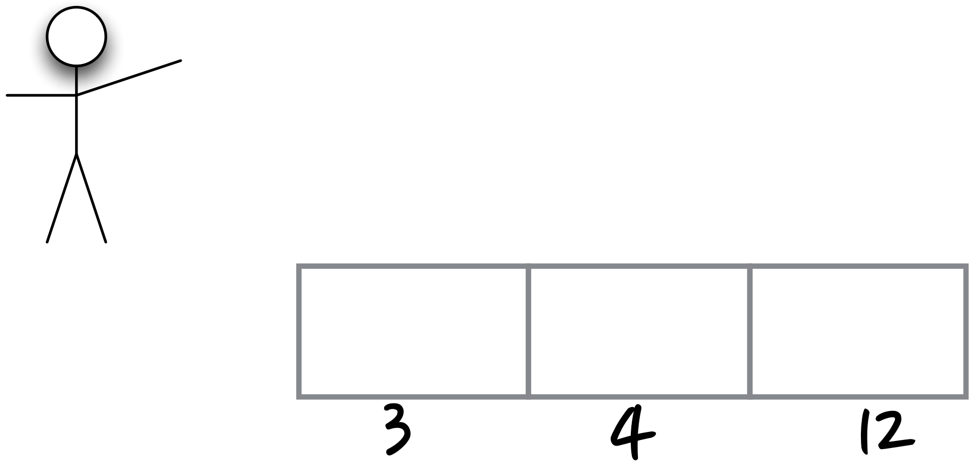
University of California
Riverside

defn of OSA



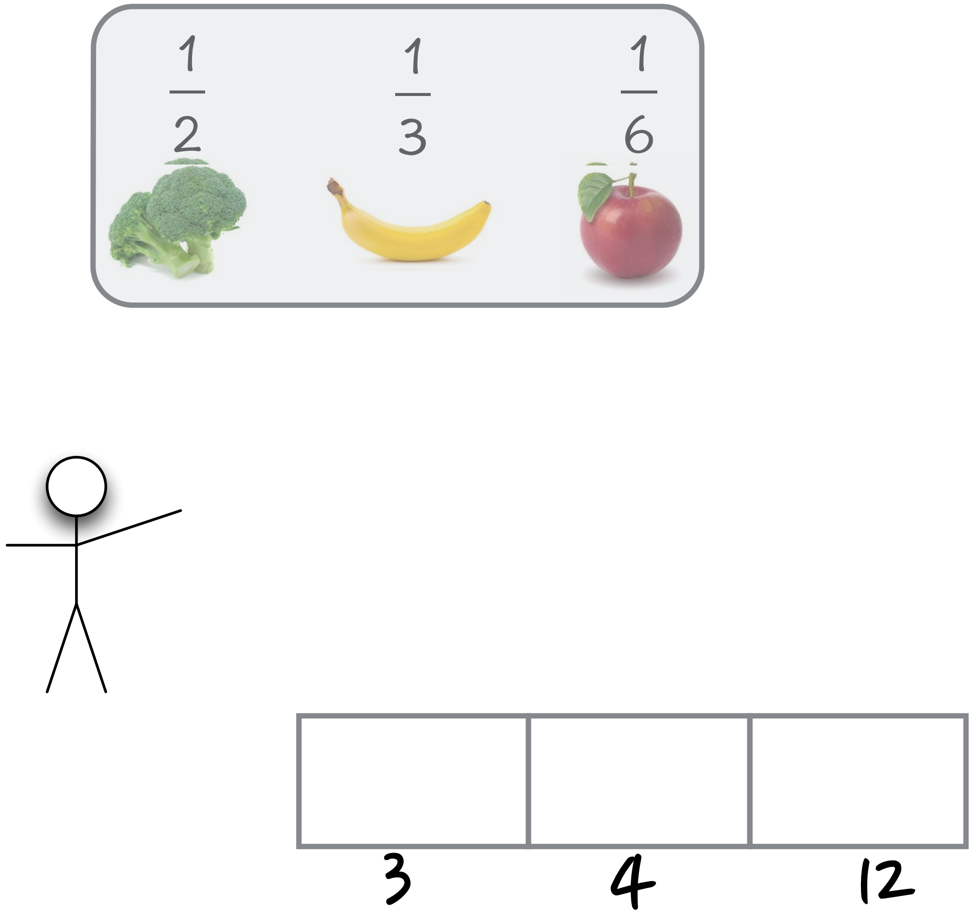
defn of OSA

GIVEN: slots with costs $c(1) \leq c(2) \leq \cdots \leq c(n)$,



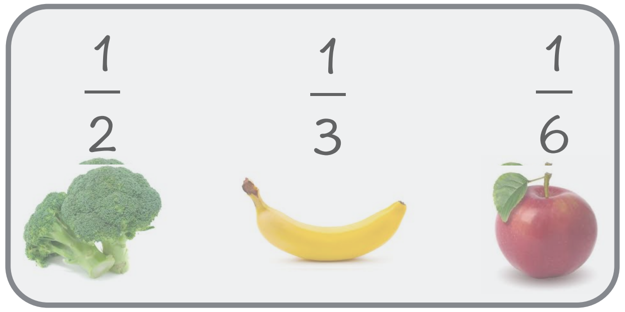
defn of OSA

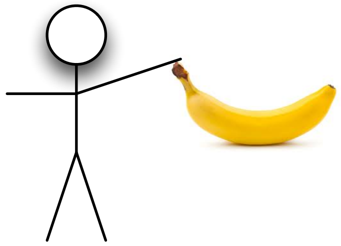
GIVEN: slots with costs $c(1) \leq c(2) \leq \cdots \leq c(n)$,
requests i_1, i_2, i_3, $\cdots$ i.i.d. from unknown distribution p.

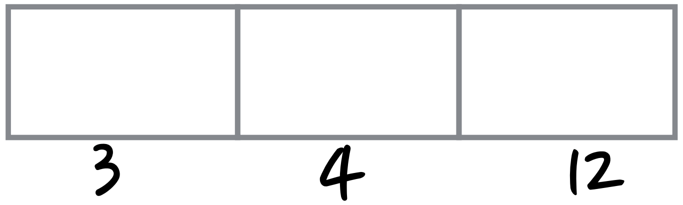


defn of OSA

GIVEN: slots with costs $c(1) \leq c(2) \leq \cdots \leq c(n)$,
requests $i_1, i_2, i_3, \cdots$ i.i.d. from unknown distribution p.



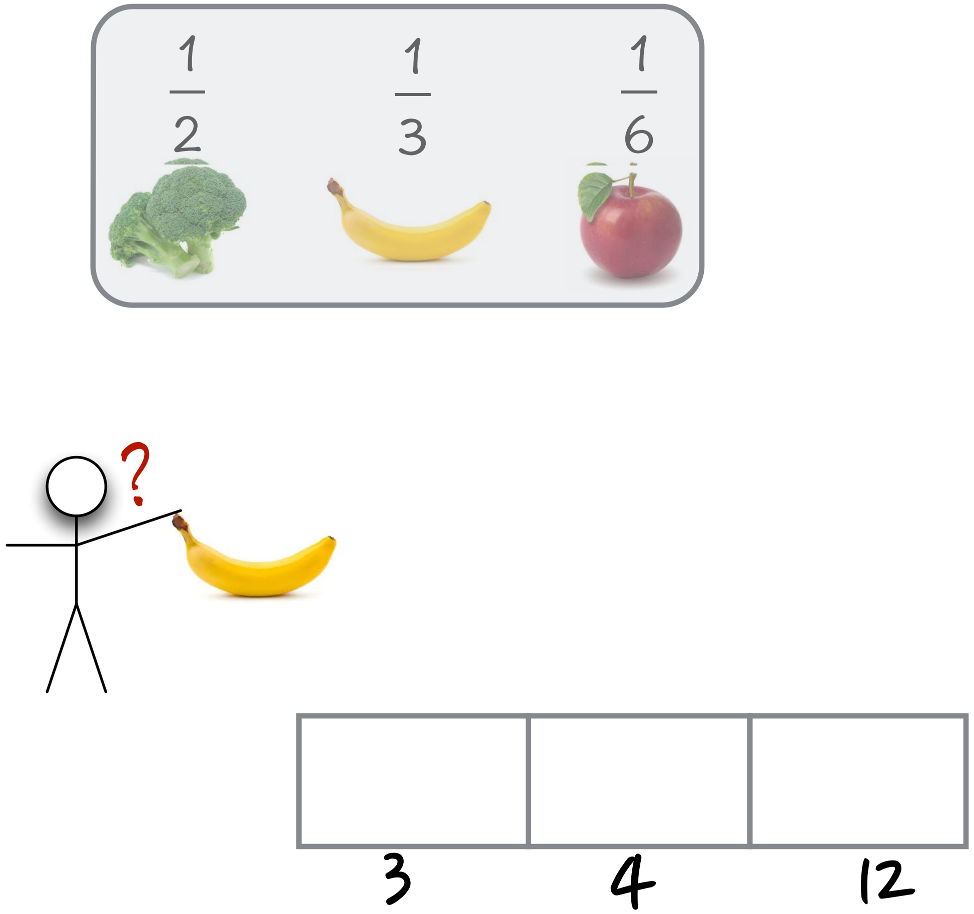




defn of OSA

GIVEN: slots with costs $c(1) \leq c(2) \leq \cdots \leq c(n)$,
requests $i_1, i_2, i_3, \cdots$ i.i.d. from unknown distribution p.

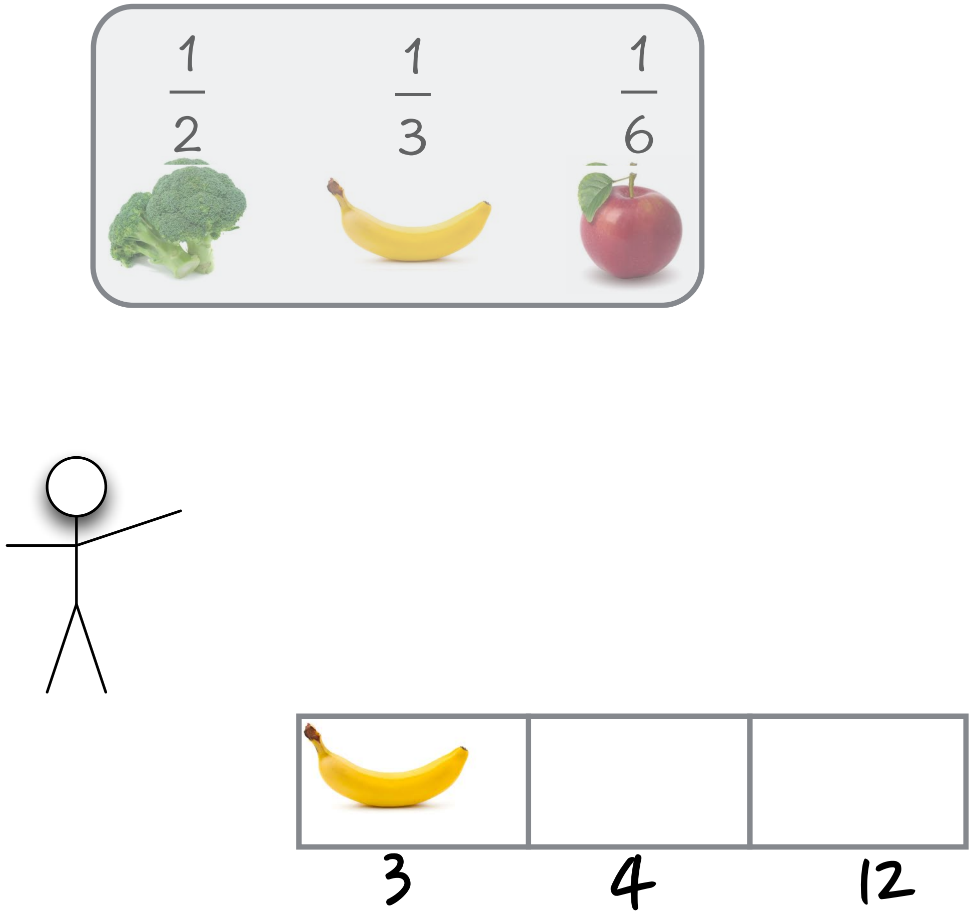
ALLOCATE: on first request of each item i, unique slot j_i for item.



defn of OSA

GIVEN: slots with costs $c(1) \le c(2) \le \cdots \le c(n)$,
requests i_1, i_2, i_3, $\cdots$ i.i.d. from unknown distribution p.

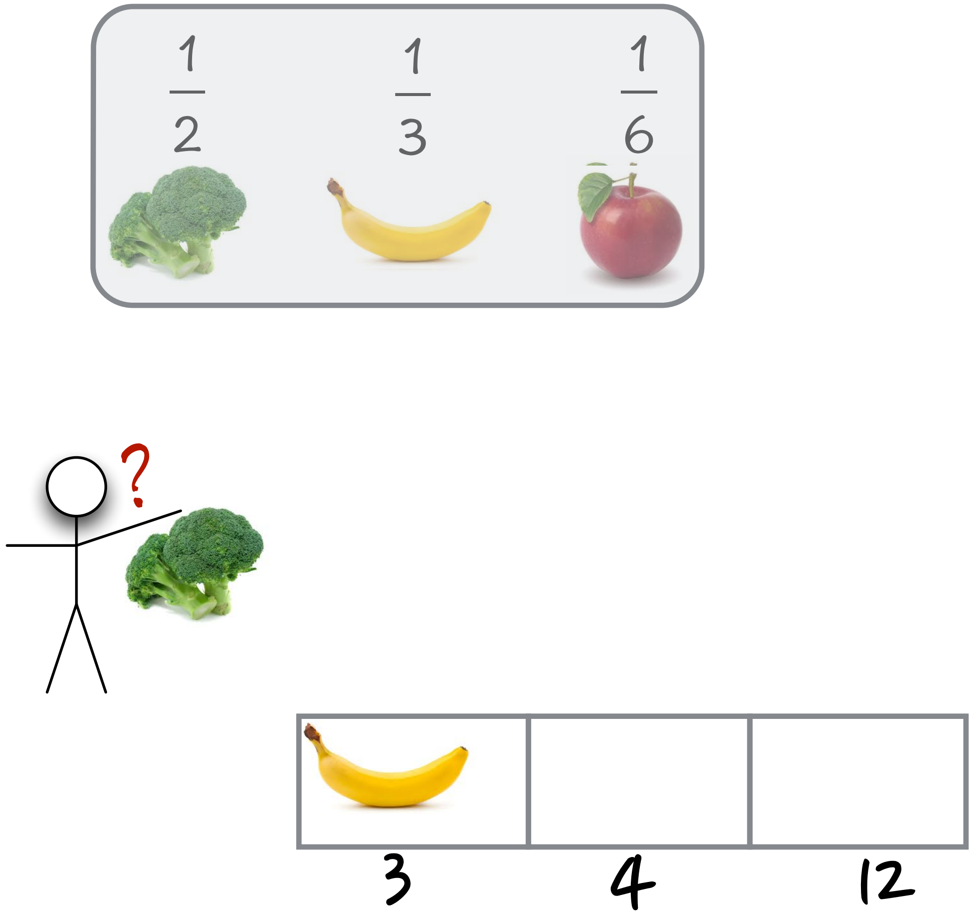
ALLOCATE: on first request of each item i, unique slot j_i for item.



defn of OSA

GIVEN: slots with costs $c(1)\leq c(2)\leq\cdots\leq c(n)$,
requests $i_1,\ i_2,\ i_3,\cdots$ i.i.d. from unknown distribution p.

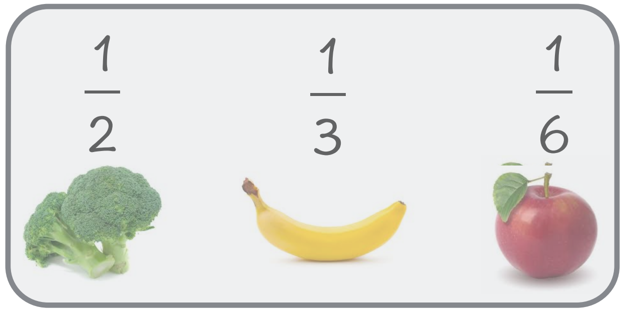
ALLOCATE: on first request of each item i, unique slot j_i for item.

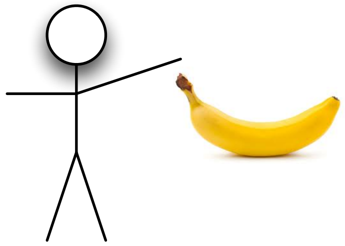


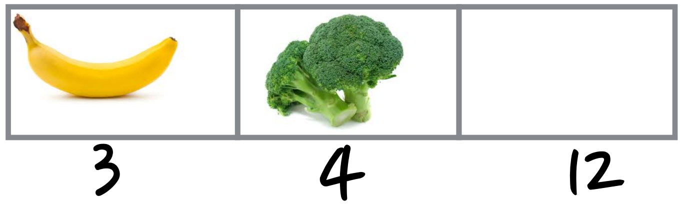
defn of OSA

GIVEN: slots with costs $c(1) \leq c(2) \leq \cdots \leq c(n)$,
requests $i_1, i_2, i_3, \cdots$ i.i.d. from unknown distribution p.

ALLOCATE: on first request of each item i, unique slot j_i for item.



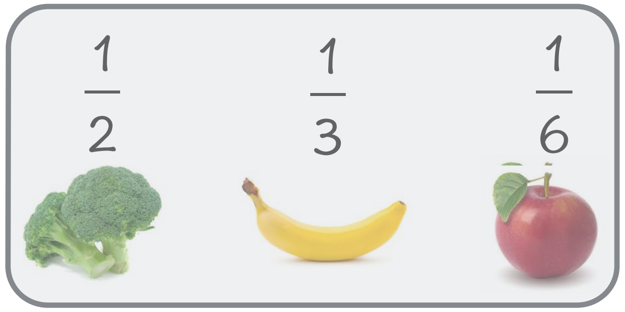


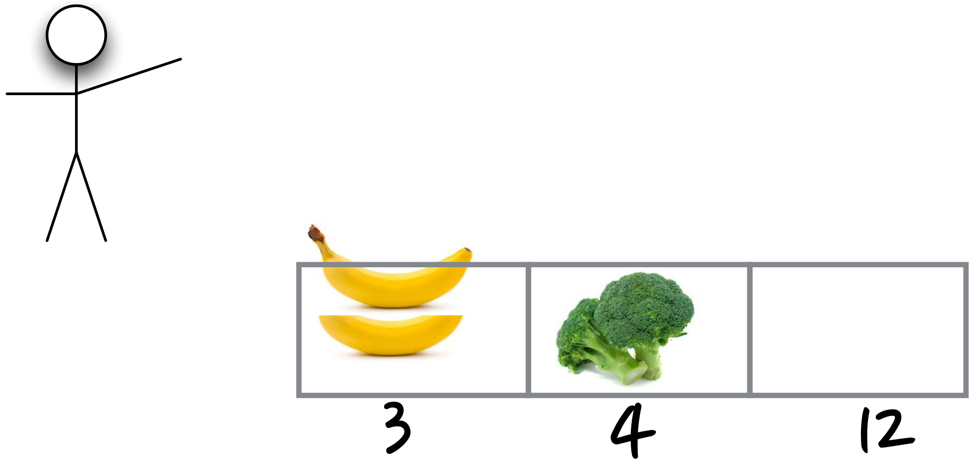


defn of OSA

GIVEN: slots with costs $c(1) \leq c(2) \leq \cdots \leq c(n)$,
requests $i_1, i_2, i_3, \cdots$ i.i.d. from unknown distribution p.

ALLOCATE: on first request of each item i, unique slot j_i for item.

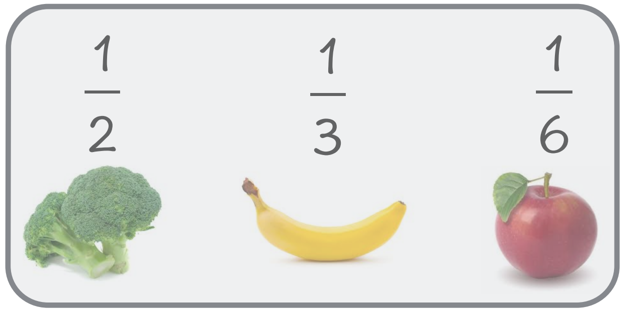


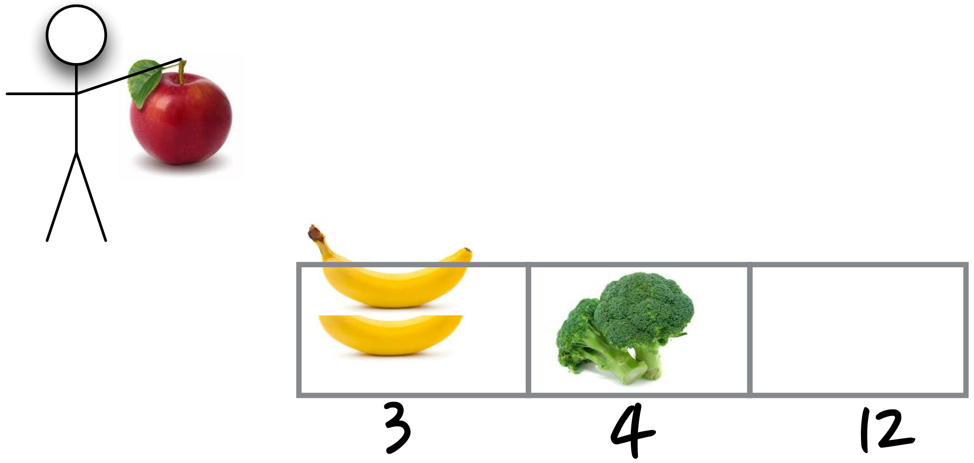


defn of OSA

GIVEN: slots with costs $c(1) \le c(2) \le \cdots \le c(n)$,
requests $i_1, i_2, i_3, \cdots$ i.i.d. from unknown distribution p.

ALLOCATE: on first request of each item i, unique slot j_i for item.

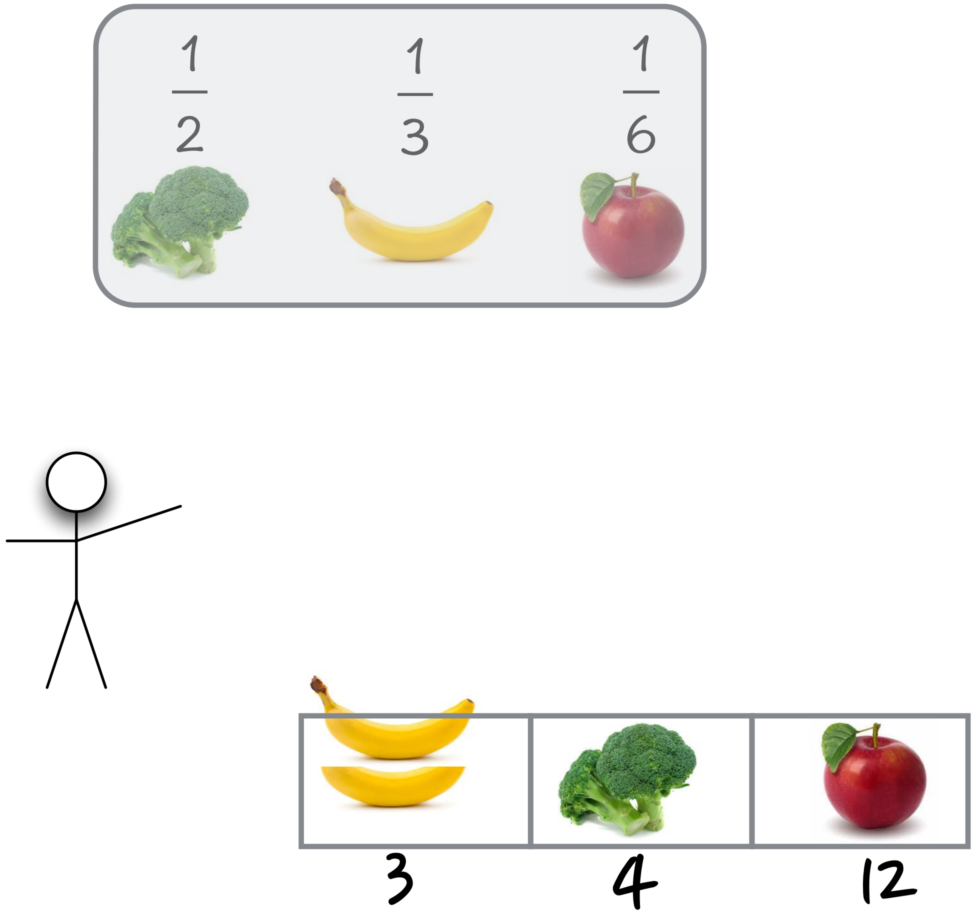




defn of OSA

GIVEN: slots with costs $c(1) \le c(2) \le \cdots \le c(n)$,
requests i_1, i_2, i_3, $\cdots$ i.i.d. from unknown distribution p.

ALLOCATE: on first request of each item i, unique slot j_i for item.

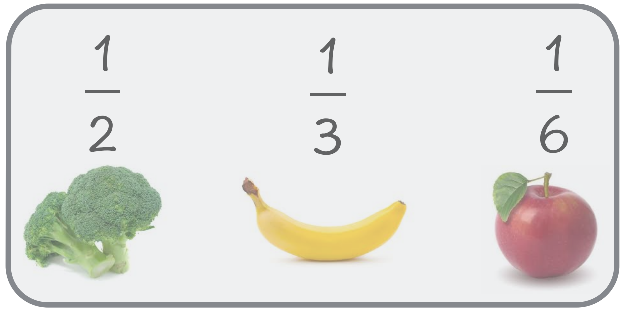


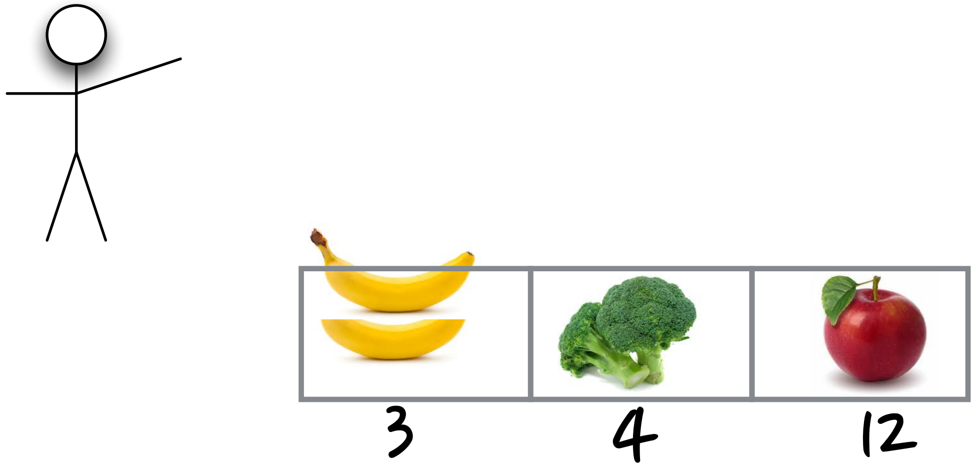
defn of OSA

GIVEN: slots with costs $c(1) \le c(2) \le \cdots \le c(n)$,
requests $i_1, i_2, i_3, \cdots$ i.i.d. from unknown distribution p.

ALLOCATE: on first request of each item i, unique slot j_i for item.

OBJECTIVE: Minimize cost $\sum_{i=1}^{n} p_i \, c(j_i)$



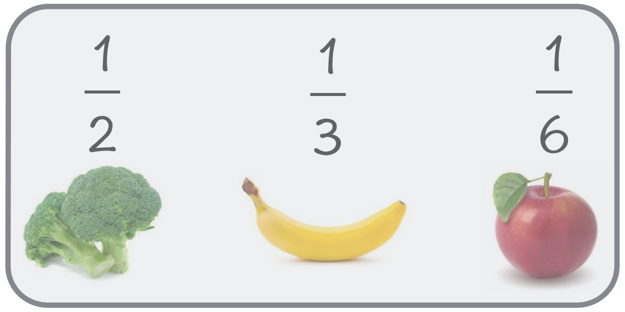


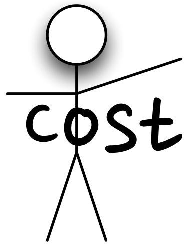
defn of OSA

GIVEN: slots with costs $c(1) \le c(2) \le \cdots \le c(n)$,
requests $i_1, i_2, i_3, \cdots$ i.i.d. from unknown distribution p.

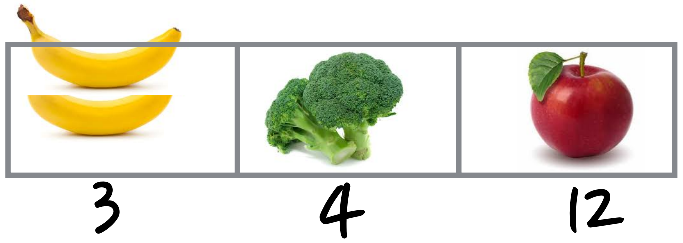
ALLOCATE: on first request of each item i, unique slot j_i for item.

OBJECTIVE: Minimize cost $\sum_{i=1}^{n} p_i \, c(j_i)$





$$\frac{1}{3} \times 3 + \frac{1}{2} \times 4 + \frac{1}{6} \times 12 = 5$$

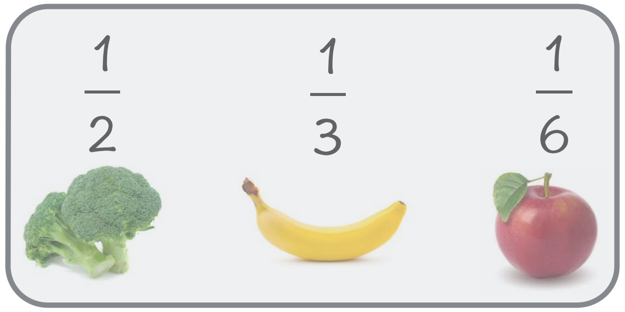


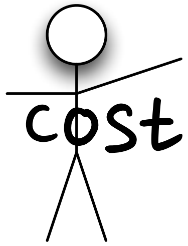
defn of OSA

GIVEN: slots with costs $c(1) \le c(2) \le \cdots \le c(n)$,
requests $i_1, i_2, i_3, \cdots$ i.i.d. from unknown distribution p.

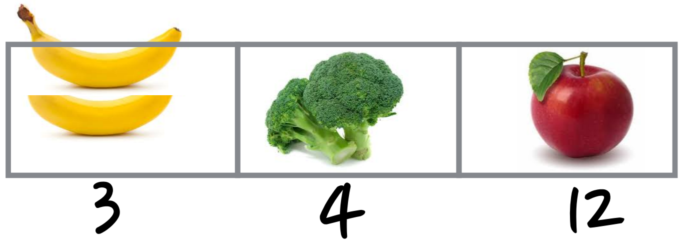
ALLOCATE: on first request of each item i, unique slot j_i for item.

OBJECTIVE: Minimize cost $\sum_{i=1}^{n} p_i \, c(j_i)$





$$\frac{1}{3} \times 3 + \frac{1}{2} \times 4 + \frac{1}{6} \times 12 = 5$$



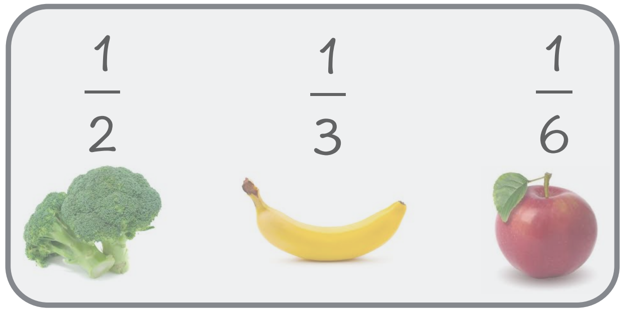
OPT?

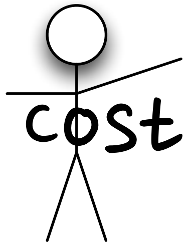
defn of OSA

GIVEN: slots with costs $c(1) \le c(2) \le \cdots \le c(n)$,
requests $i_1, i_2, i_3, \cdots$ i.i.d. from unknown distribution p.

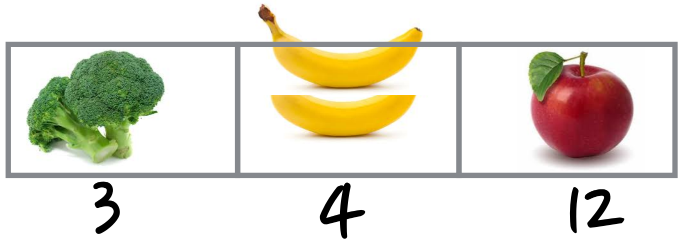
ALLOCATE: on first request of each item i, unique slot j_i for item.

OBJECTIVE: Minimize cost $\sum_{i=1}^{n} p_i \, c(j_i)$





$$\frac{1}{3} \times 3 + \frac{1}{2} \times 4 + \frac{1}{6} \times 12 = 5$$



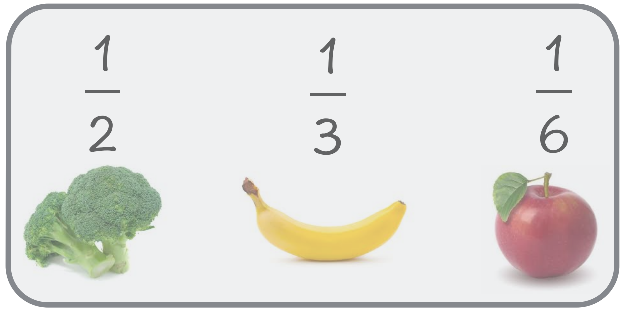
OPT?

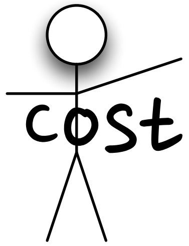
defn of OSA

GIVEN: slots with costs $c(1) \le c(2) \le \cdots \le c(n)$,
requests $i_1, i_2, i_3, \cdots$ i.i.d. from unknown distribution p.

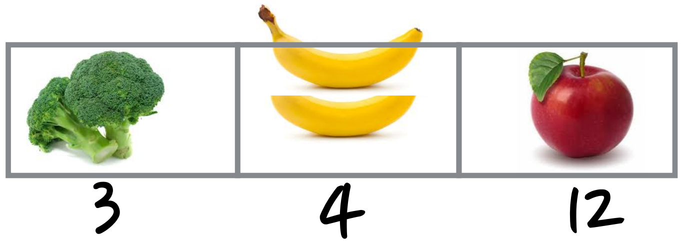
ALLOCATE: on first request of each item i, unique slot j_i for item.

OBJECTIVE: Minimize cost $\sum_{i=1}^{n} p_i \, c(j_i)$





$$\frac{1}{3}\times 3 + \frac{1}{2}\times 4 + \frac{1}{6}\times 12 = 5$$



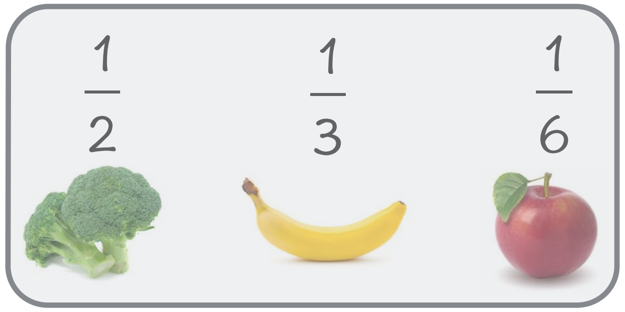
OPT? $\frac{1}{2}\times 3 + \frac{1}{3}\times 4 + \frac{1}{6}\times 12 = \frac{29}{6} < 5$

defn of OSA

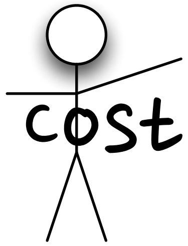
GIVEN: slots with costs $c(1) \le c(2) \le \cdots \le c(n)$,
requests $i_1, i_2, i_3, \cdots$ i.i.d. from unknown distribution p.

ALLOCATE: on first request of each item i, unique slot j_i for item.

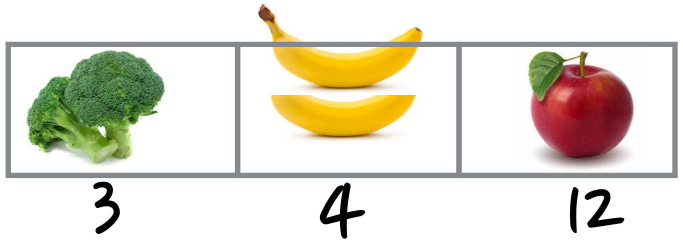
OBJECTIVE: Minimize cost $\sum_{i=1}^{n} p_i \, c(j_i)$



FcFS = use cheapest available slot



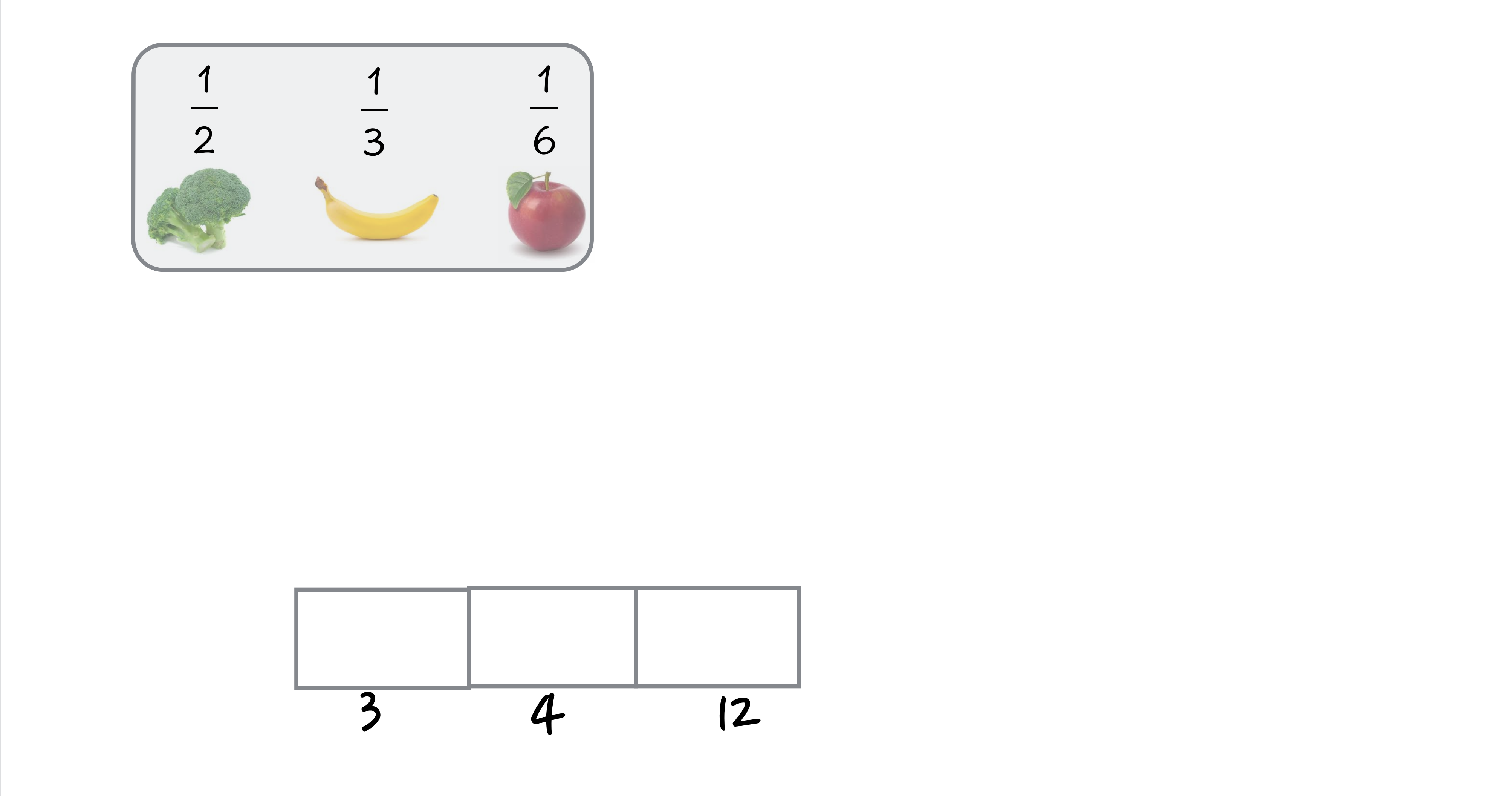
$$\frac{1}{3} \times 3 + \frac{1}{2} \times 4 + \frac{1}{6} \times 12 = 5$$



OPT? $$\frac{1}{2} \times 3 + \frac{1}{3} \times 4 + \frac{1}{6} \times 12 = \frac{29}{6} < 5$$

QUESTION: Is FcFS competitive with offline OPT (which knows p)?

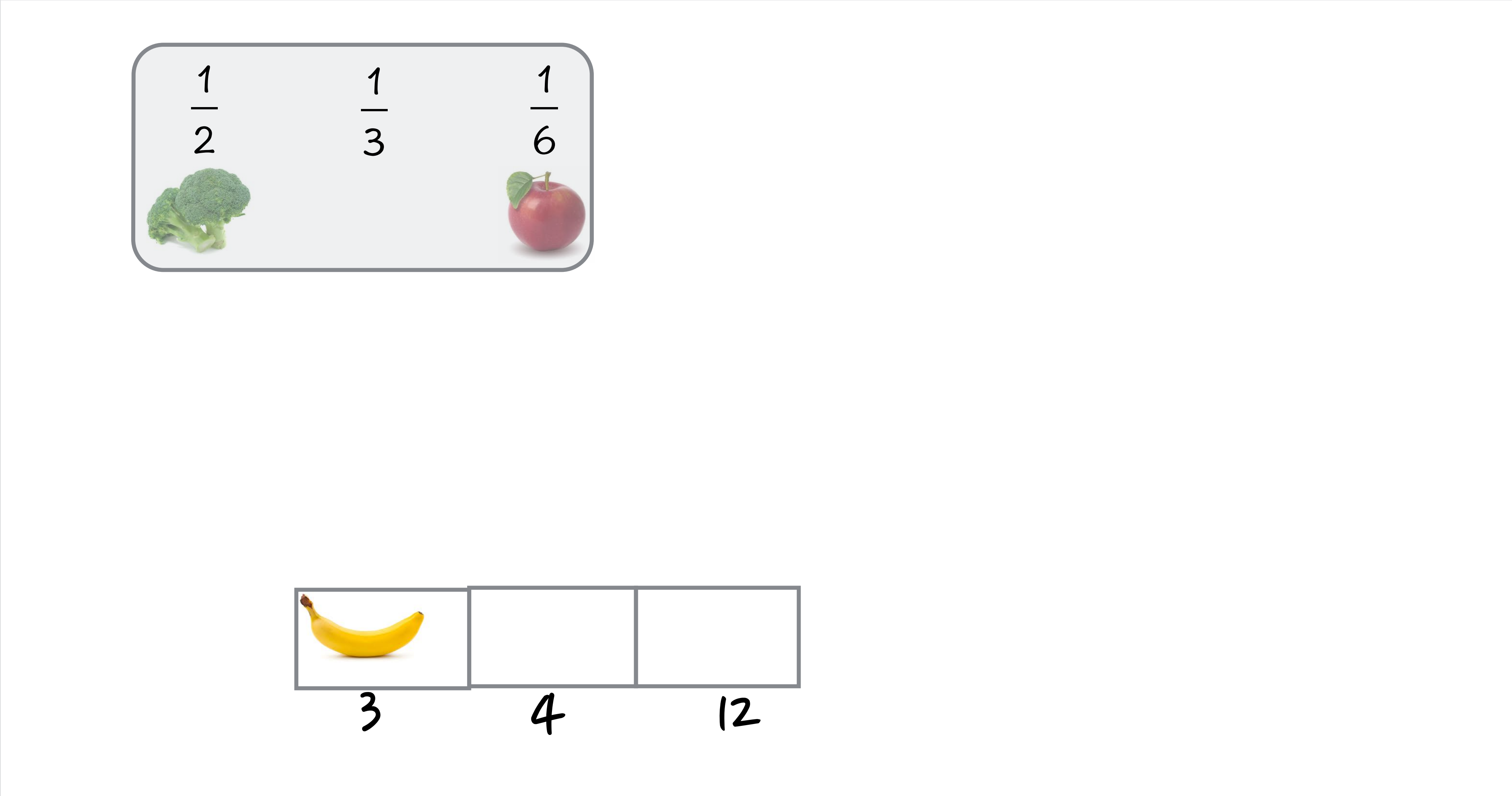
FcFS is just: sampling all items WITHOUT REPLACEMENT from p into slots.



(Because FcFS ignores repeat requests.)

QUESTION: Is FcFS competitive with offline OPT (which knows p)?

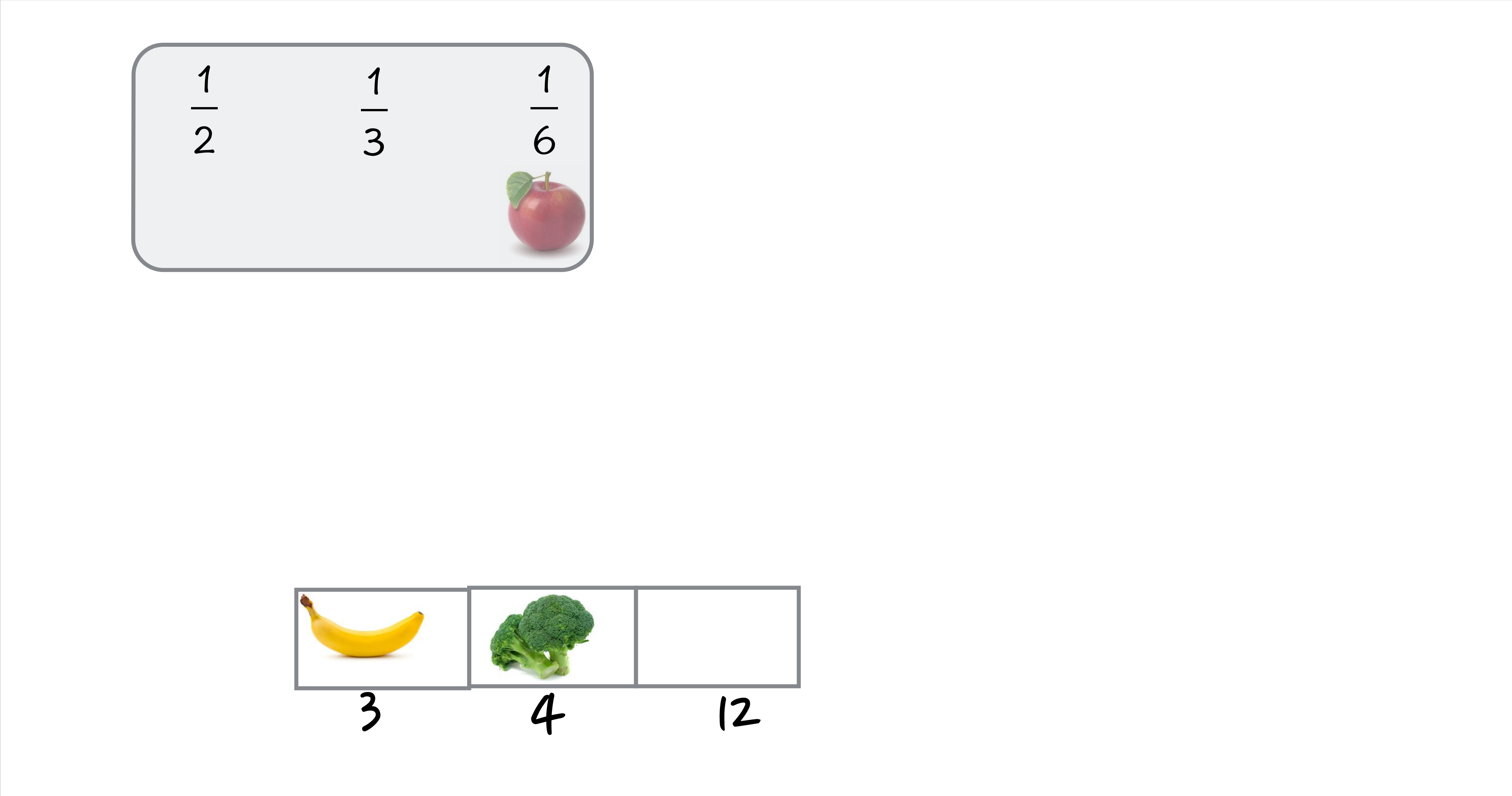
FcFS is just: sampling all items WITHOUT REPLACEMENT from p into slots.



(Because FcFS ignores repeat requests.)

QUESTION: Is FCFS competitive with offline OPT (which knows p)?

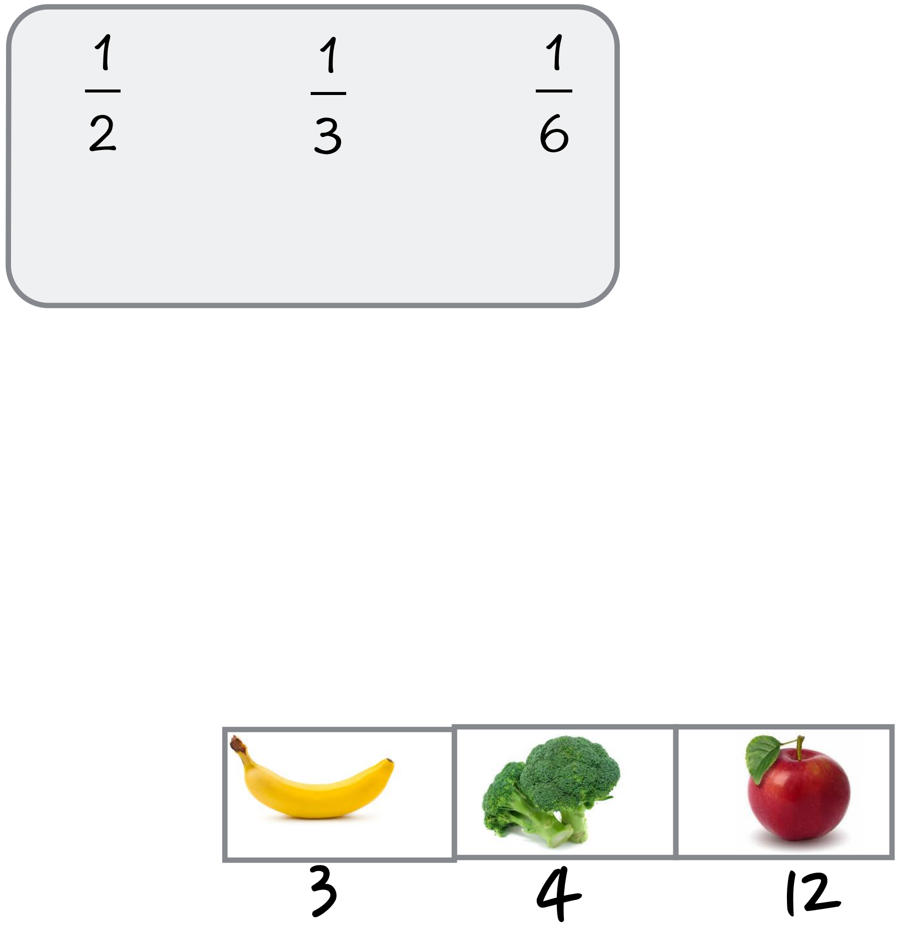
FCFS is just: sampling all items WITHOUT REPLACEMENT from p into slots.



(Because FCFS ignores repeat requests.)

QUESTION: Is FCFS competitive with offline OPT (which knows p)?

FCFS is just: sampling all items WITHOUT REPLACEMENT from p into slots.



(Because FCFS ignores repeat requests.)

main results

THM 1: FCFS is optimally competitive for online Slot Allocation.

TODAY

THM 2: optimal competitive ratios:

(a) Arbitrary slot costs: $1+H_{n-1}$

TODAY (most technically interesting)

(b) concave slot costs: 2

(c) Logarithmic slot costs: 1^*

*asymptotically: FCFS guarantees cost OPT + O(log OPT)..

THM 3: For online Huffman coding, online algorithm with cost

$$\mathrm{OPT} + 2 \log_2 (1 + \mathrm{OPT}) + 2.$$

(some) related work

competitive analysis w. STATIC OPT & unknown item distribution

- List management ~ OSA with linear cost function [34]
- Paging ~ OSA with 0/1 cost function
  (Independent Reference model — IRM) e.g. [1,12]

(some) related work

competitive analysis w. STATIC OPT & unknown item distribution

- List management ~ OSA with linear cost function [34]
- Paging ~ OSA with 0/1 cost function
  (Independent Reference model — IRM) e.g. [1,12]

ADAPTIVE Huffman coding [8,14,26,37,38,39]

- also one-pass, but codewords change adaptively.
- text can be arbitrarily ordered.

“WORST-DISTRIBUTION” COMPETITIVE ANALYSIS:

Oh no! Worst-case analysis is too pessimistic!

Oh no! Average-case analysis is too optimistic!

Show that your algorithm does well against any distribution in a class of distributions.

- competitive paging [23,28,33,42]
  (e.g. Markov paging, diffuse adversary)
- online bin packing [4,19]; online knapsack [30]
- online facility location, Steiner tree [32]
- Secretary problem [6,16]; online auctions [2,9,13]
- adwords [31,21,5]

sampling w/o replacement for poker tournaments

valuing chips = estimating, for sampling without replacement:

Pr[item i ends in slot j]

your expected final payout, given your current chips:

Pr[first place] * (payout for first place)

+ Pr[second place] * (payout for second place)

+ …

+ Pr[last place] * (payout for last place)

random model for your final placement, given current chips:

- round 1: select first-place player by random draw where

  Pr[player i wins first place] = players chips / total chips

- round 2: select second-place player from REMAINING players, again

  Pr[player i wins second place] = players chips / total chips of remaining players

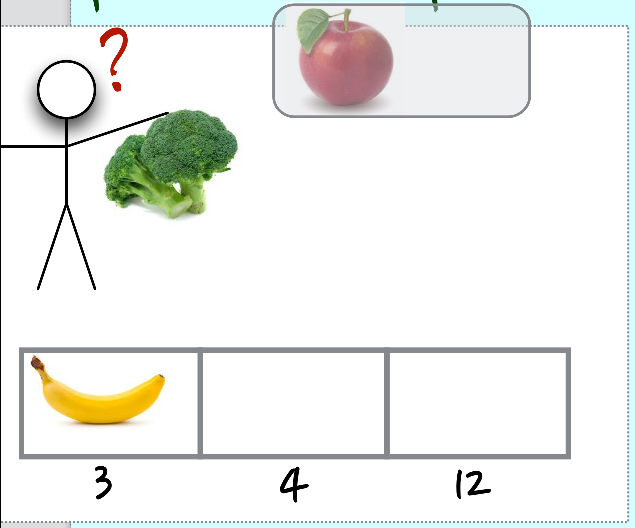
- etc… = sampling players without replacement, using current chip counts as probabilities

  You finish in j'th place in tournament <==> You are the j'th sample

THEOREM 1: FCFS is optimally competitive among all online algorithms.

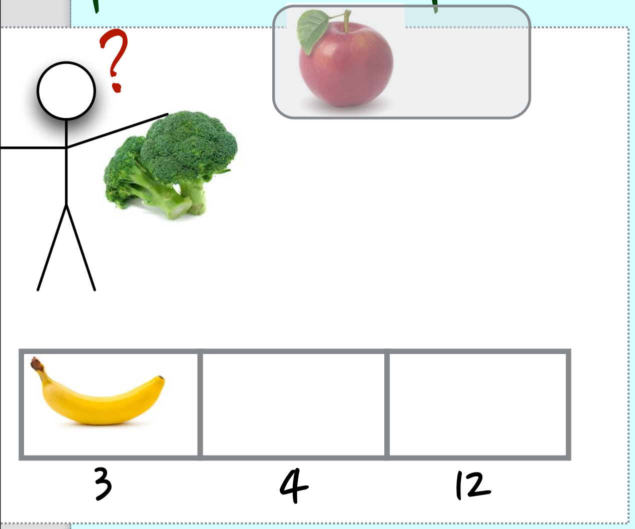
THEOREM 1: FCFS is optimally competitive among all online algorithms.

proof attempt 1



THEOREM 1: FCFS is optimally competitive among all online algorithms.

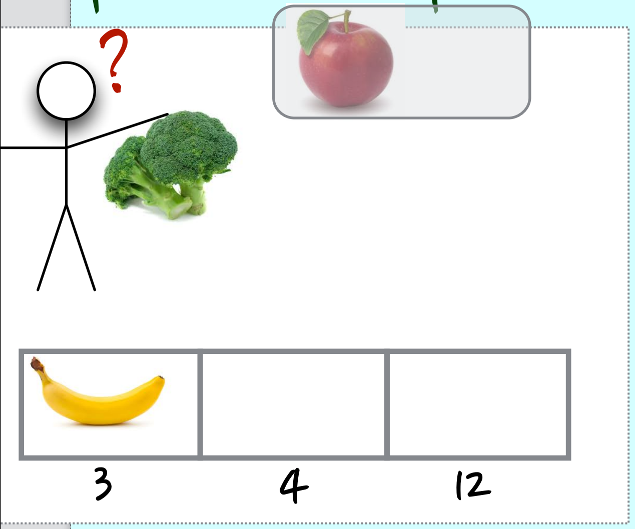
proof attempt 1



when allocating slot for broccoli, the subproblem that remains is: allocate slots to broccoli and apple. For this subproblem, all that matters is which slots remain, and the relative frequencies of broccoli and apple. Frequency of banana is irrelevant. Given that broccoli was sampled before apple, broccoli is more likely to have higher frequency, so you should put it in cheapest available slot.

THEOREM 1: FCFS is optimally competitive among all online algorithms.

proof attempt 1



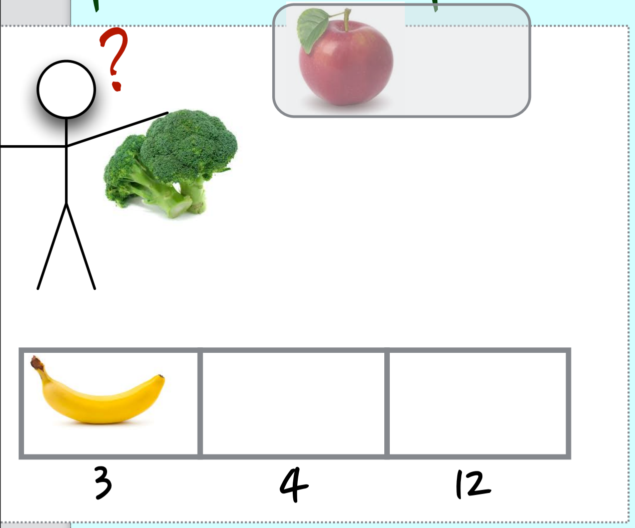
When allocating slot for broccoli, the subproblem that remains is: allocate slots to broccoli and apple. For this subproblem, all that matters is which slots remain, and the relative frequencies of broccoli and apple. Frequency of banana is irrelevant. Given that broccoli was sampled before apple, broccoli is more likely to have higher frequency, so you should put it in cheapest available slot.

CAREFUL! (1) What does "more likely to have higher frequency" mean? p is fixed!

(2) real objective is to minimize competitive ratio (not absolute cost)

THEOREM 1: FCFS is optimally competitive among all online algorithms.

proof attempt 1



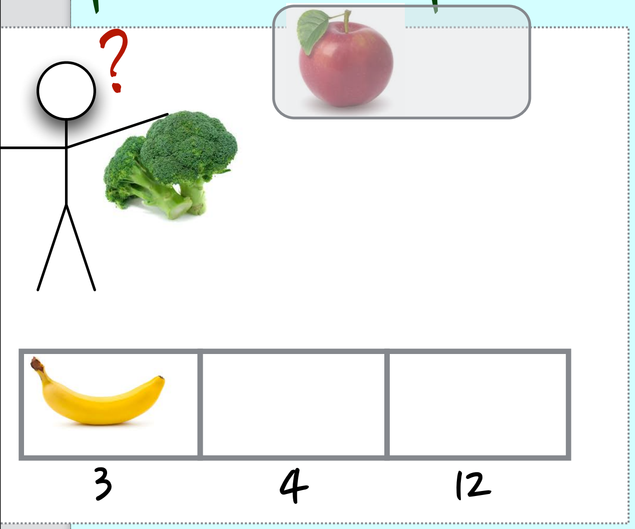
When allocating slot for broccoli, the subproblem that remains is: allocate slots to broccoli and apple. For this subproblem, all that matters is which slots remain, and the relative frequencies of broccoli and apple. Frequency of banana is irrelevant. Given that broccoli was sampled before apple, broccoli is more likely to have higher frequency, so you should put it in cheapest available slot.

CAREFUL! (1) What does "more likely to have higher frequency" mean? p is fixed!
(2) real objective is to minimize competitive ratio (not absolute cost)

FIX: Prove stronger result: FCFS best among WEAKLY ONLINE algorithms.

THEOREM 1: FCFS is optimally competitive among all online algorithms.

proof attempt 1



when allocating slot for broccoli, the subproblem that remains is: allocate slots to broccoli and apple. For this subproblem, all that matters is which slots remain, and the relative frequencies of broccoli and apple. Frequency of banana is irrelevant. Given that broccoli was sampled before apple, broccoli is more likely to have higher frequency, so you should put it in cheapest available slot.

CAREFUL! (1) what does "more likely to have higher frequency" mean? p is fixed!
(2) real objective is to minimize competitive ratio (not absolute cost)

FIX: Prove stronger result: FCFS best among WEAKLY ONLINE algorithms.

WEAKLY ONLINE = alg. KNOWS p but chooses next slot just BEFORE next request.

Now "more likely" is well-defined... (Rest of proof is technical but not surprising.)

THEOREM 2: Optimal competitive ratios for online Slot Allocation:

(a) arbitrary slot costs: $1 + H_{n-1}$

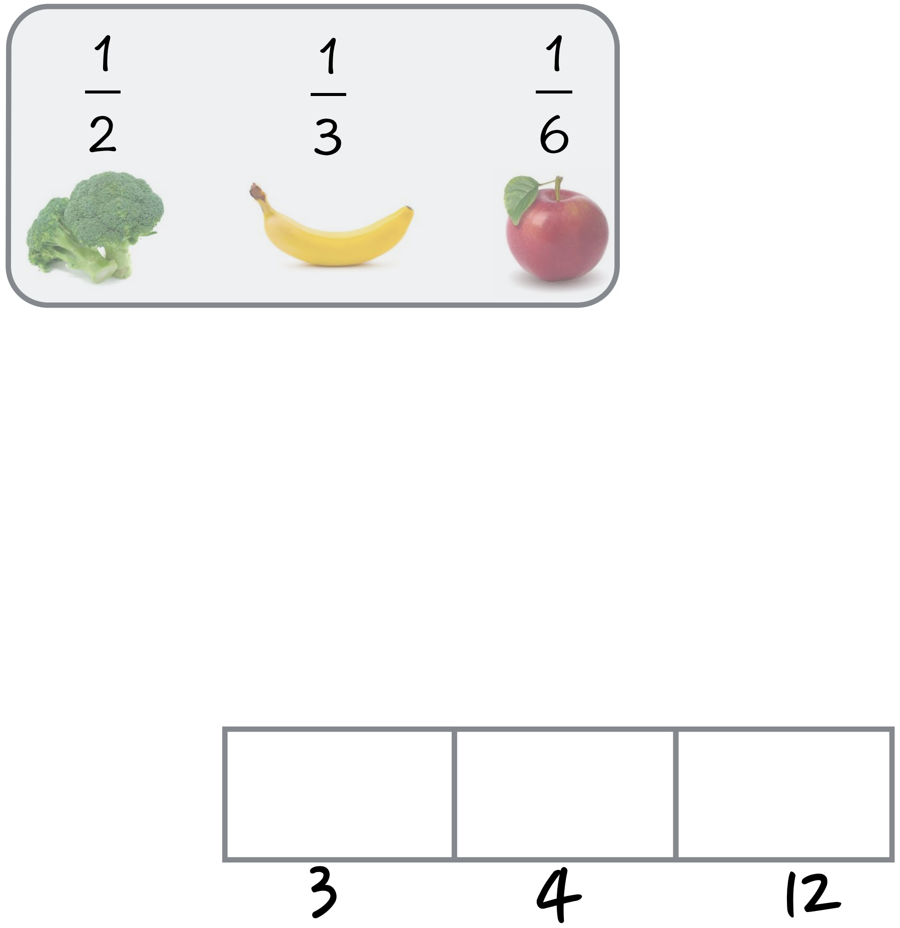
NEXT (UPPER BOUND ONLY)

(b) concave slot costs: 2

(c) logarithmic slot costs: 1^*

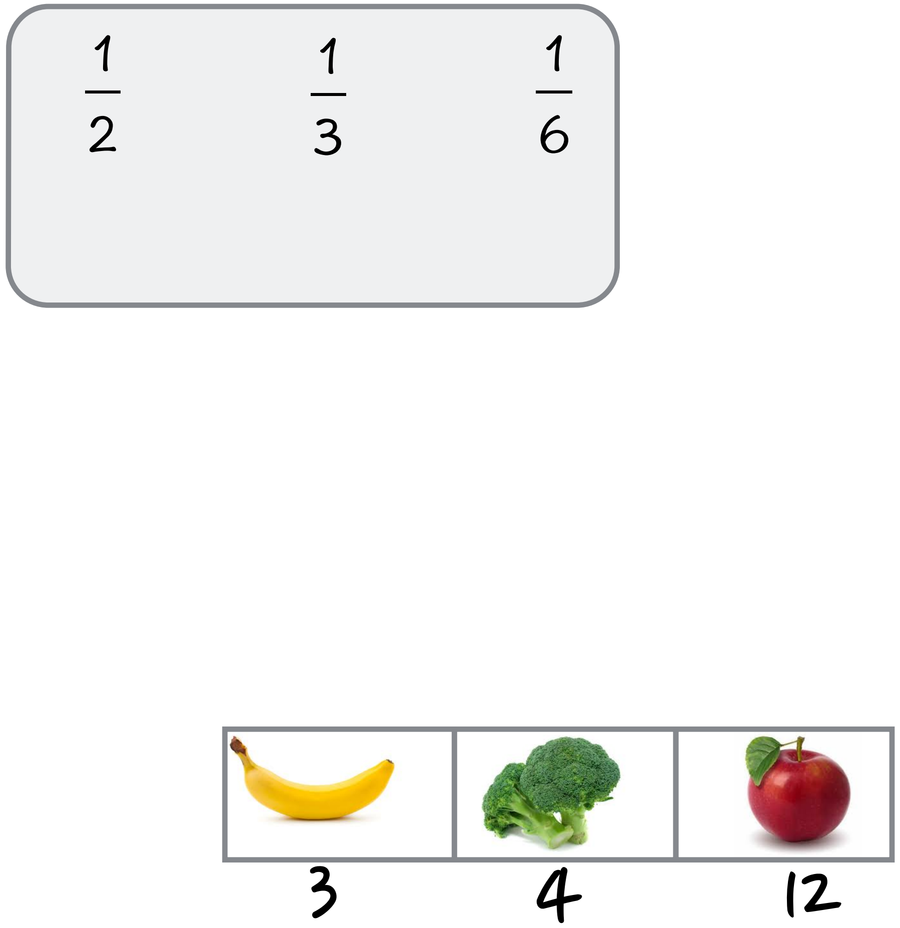
*asymptotically: FCFS guarantees cost $\mathrm{OPT} + O(\log \mathrm{OPT})$.

FCFS is just: sampling all items WITHOUT REPLACEMENT from p into slots.



OPT is just: allocate highest-cost slots to lowest-probability items

FCFS is just: sampling all items WITHOUT REPLACEMENT from p into slots.



OPT is just: allocate highest-cost slots to lowest-probability items

THEOREM 2(a) upper bound: FCFS is $(1+H_{n-1})$-competitive. Proof idea

THEOREM 2(a) upper bound: FCFS is $(1+H_{n-1})$-competitive.

proof idea

1. Example: five slots of cost 0, three slots of cost 1: 🟦🟦🟦🟦🟦🟥🟥🟥

   Denote five largest probabilities L (large), three smallest probabilities S (small).

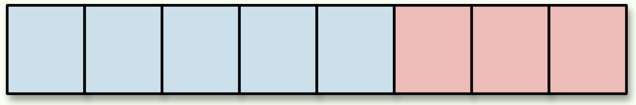
2. optimal solution = | L | L | L | L | L | S | S | S | , OPT cost is S+S+S.

3. we bound FCFS's expected cost for large items by $H_5 \times \mathrm{OPT}$.

THEOREM 2(a) upper bound: FCFS is $(1+H_{n-1})$-competitive.

Proof idea

1. Example: five slots of cost 0, three slots of cost 1:

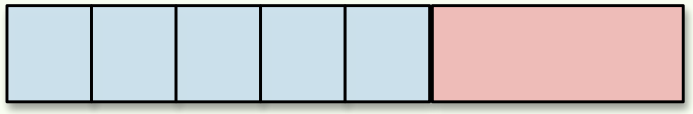


Denote five largest probabilities L (large), three smallest probabilities S (small).

2. Optimal solution = L L L L L S S S , OPT cost is S+S+S.

3. We bound FCFS's expected cost for large items by $H_5 \times$OPT.

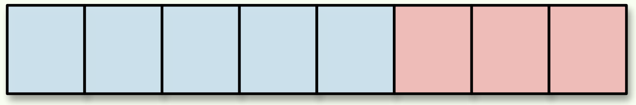
4. EXPOSURE view of FCFS:



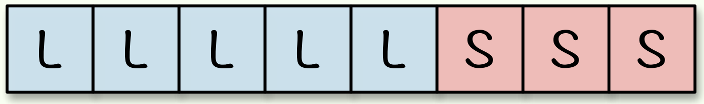
i. Sample items without replacement into slots, but keep them hidden.

THEOREM 2(a) upper bound: FcFS is $(1+H_{n-1})$-competitive.

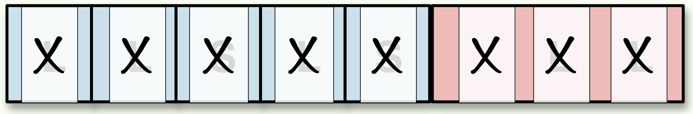
Proof idea

1. Example: five slots of cost 0, three slots of cost 1: 

   Denote five largest probabilities L (large), three smallest probabilities S (small).

2. Optimal solution =  L L L L L S S S, OPT cost is S+S+S.

3. We bound FcFS's expected cost for large items by $H_5 \times$OPT.

4. EXPOSURE view of FcFS:  X X X X X X X X

   i. Sample items without replacement into slots, but keep them hidden.

THEOREM 2(a) upper bound: FCFS is $(1+H_{n-1})$-competitive.

proof idea

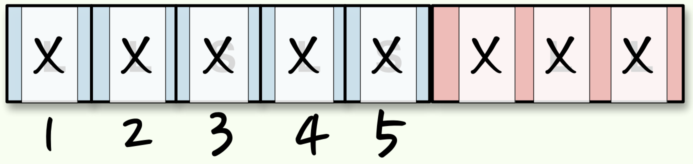
1. Example: five slots of cost 0, three slots of cost 1 : 🟦🟦🟦🟦🟦🟥🟥🟥

   Denote five largest probabilities L (large), three smallest probabilities S (small).

2. optimal solution = | L | L | L | L | L | S | S | S |, OPT cost is S+S+S.

3. we bound FCFS's expected cost for large items by $H_5 \times \mathrm{OPT}$.

4. EXPOSURE view of FCFS:



   i. Sample items without replacement into slots, but keep them hidden.

   ii. Expose requests 1-5 one by one. In each step where a SMALL item comes up, ALSO expose the LAST not-yet-exposed LARGE item. (call these COSTLY.)

THEOREM 2(a) upper bound: FcFS is $(1+H_{n-1})$-competitive.

proof idea

1. Example: five slots of cost 0, three slots of cost 1 :

🟦	🟦	🟦	🟦	🟦	🟥	🟥	🟥

Denote five largest probabilities L (large), three smallest probabilities S (small).

2. optimal solution =

L	L	L	L	L	S	S	S

, OPT cost is S+S+S.

3. we bound FcFS's expected cost for large items by $H_5 \times$OPT.

4. EXPOSURE view of FcFS:

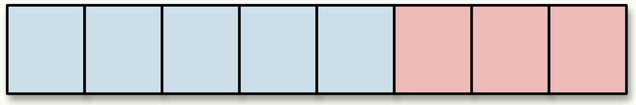
L	X	X	X	X	X	X	X
1	2	3	4	5			

i. Sample items without replacement into slots, but keep them hidden.

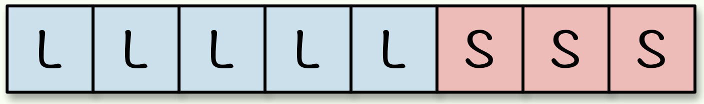
ii. Expose requests 1-5 one by one. In each step where a SMALL item comes up, ALSO expose the LAST not-yet-exposed LARGE item. (call these COSTLY.)

THEOREM 2(a) upper bound: FCFS is $(1+H_{n-1})$-competitive.

proof idea

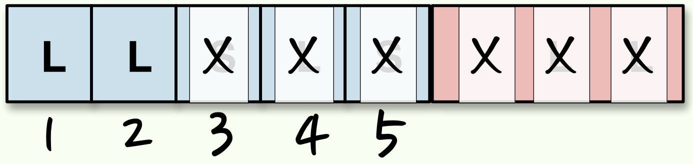
1. Example: five slots of cost 0, three slots of cost 1 : 

   Denote five largest probabilities L (large), three smallest probabilities S (small).

2. Optimal solution = 

, OPT cost is S+S+S.

3. We bound FCFS's expected cost for large items by $H_5 \times$OPT.

4. EXPOSURE view of FCFS: 

   i. Sample items without replacement into slots, but keep them hidden.

   ii. Expose requests 1-5 one by one. In each step where a SMALL item comes up, ALSO expose the LAST not-yet-exposed LARGE item. (call these COSTLY.)

THEOREM 2(a) upper bound: FCFS is $(1+H_{n-1})$-competitive. proof idea

1. Example: five slots of cost 0, three slots of cost 1:

🟦	🟦	🟦	🟦	🟦	🟥	🟥	🟥

   Denote five largest probabilities L (large), three smallest probabilities S (small).

2. optimal solution =

L	L	L	L	L	S	S	S

   , OPT cost is S+S+S.

3. we bound FCFS's expected cost for large items by $H_5 \times \mathrm{OPT}$.

4. EXPOSURE view of FCFS:

L	L	S	X	X	X	X	X
1	2	3	4	5			

   i. Sample items without replacement into slots, but keep them hidden.

   ii. Expose requests 1-5 one by one. In each step where a SMALL item comes up, ALSO expose the LAST not-yet-exposed LARGE item. (call these COSTLY.)

THEOREM 2(a) upper bound: FCFS is $(1+H_{n-1})$-competitive.

proof idea

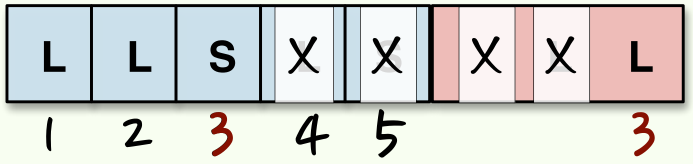
1. Example: five slots of cost 0, three slots of cost 1: 🟦🟦🟦🟦🟦🟥🟥🟥

   Denote five largest probabilities L (large), three smallest probabilities S (small).

2. optimal solution = | L | L | L | L | L | S | S | S |, OPT cost is S+S+S.

3. we bound FCFS's expected cost for large items by $H_5 \times$OPT.

4. EXPOSURE view of FCFS:



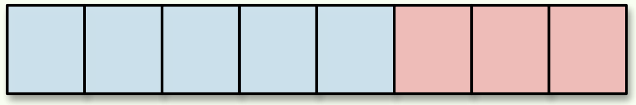
   i. Sample items without replacement into slots, but keep them hidden.

   ii. Expose requests 1-5 one by one. In each step where a SMALL item comes up, ALSO expose the LAST not-yet-exposed LARGE item. (call these COSTLY.)

THEOREM 2(a) upper bound: FCFS is $(1+H_{n-1})$-competitive.

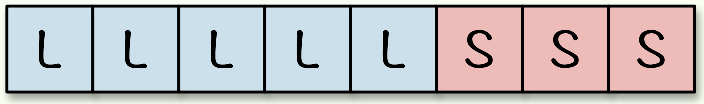
proof idea

1. Example: five slots of cost 0, three slots of cost 1:



Denote five largest probabilities L (large), three smallest probabilities S (small).

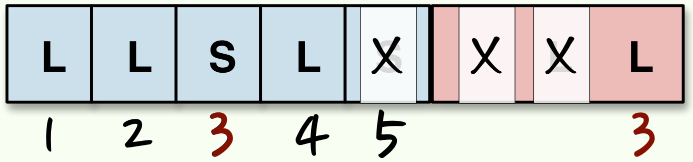
2. optimal solution =



, OPT cost is S+S+S.

3. We bound FCFS's expected cost for large items by $H_5 \times$OPT.

4. EXPOSURE view of FCFS:



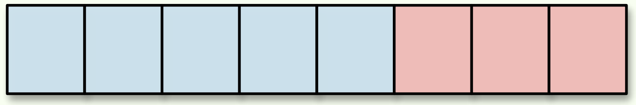
i. Sample items without replacement into slots, but keep them hidden.

ii. Expose requests 1-5 one by one. In each step where a SMALL item comes up, ALSO expose the LAST not-yet-exposed LARGE item. (call these COSTLY.)

THEOREM 2(a) upper bound: FCFS is $(1+H_{n-1})$-competitive.

proof idea

1. Example: five slots of cost 0, three slots of cost 1:



Denote five largest probabilities L (large), three smallest probabilities S (small).

2. Optimal solution =

L	L	L	L	L	S	S	S

, OPT cost is S+S+S.

3. we bound FCFS's expected cost for large items by $H_5 \times$OPT.

4. EXPOSURE view of FCFS:

L	L	S	L	S	X	X	L
1	2	3	4	5			3

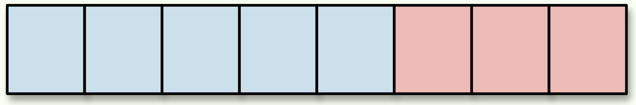
i. Sample items without replacement into slots, but keep them hidden.

ii. Expose requests 1-5 one by one. In each step where a SMALL item comes up, ALSO expose the LAST not-yet-exposed LARGE item. (call these COSTLY.)

THEOREM 2(a) upper bound: FCFS is $(1+H_{n-1})$-competitive.

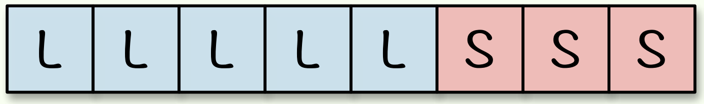
Proof idea

1. Example: five slots of cost 0, three slots of cost 1:



Denote five largest probabilities L (large), three smallest probabilities S (small).

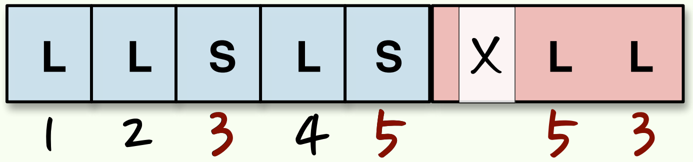
2. optimal solution =



, OPT cost is S+S+S.

3. we bound FCFS's expected cost for large items by $H_5 \times \mathrm{OPT}$.

4. EXPOSURE view of FCFS:



i. Sample items without replacement into slots, but keep them hidden.

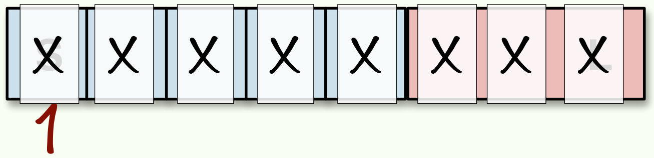
ii. Expose requests 1-5 one by one. In each step where a SMALL item comes up, ALSO expose the LAST not-yet-exposed LARGE item. (call these COSTLY.)

THEOREM 2(a) upper bound: FCFS is $(1+H_{n-1})$-competitive.

proof idea
(continued)

2. optimal solution = | L | L | L | L | L | S | S | S | OPT = S+S+S...

3. whenever a small item is exposed, expose the last not-yet-exposed large item (call it COSTLY):



THEOREM 2(a) upper bound: FCFS is $(1+H_{n-1})$-competitive.

proof idea

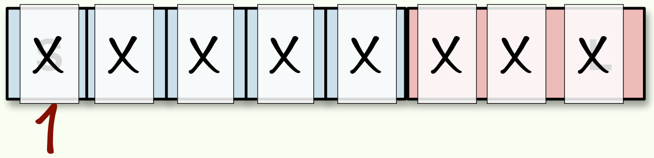
(continued)

2. optimal solution =

L	L	L	L	L	S	S	S

OPT = S+S+S...

3. whenever a small item is exposed, expose the last not-yet-exposed large item (call it **COSTLY**):



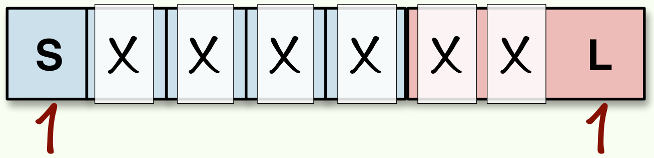
4. REQUEST 1: Pr[request 1 small] is at most $\dfrac{S+S+S}{\text{sum of all probabilities}} = \dfrac{OPT}{\text{sum of all probabilities}}$

THEOREM 2(a) upper bound: FCFS is $(1+H_{n-1})$-competitive.

proof idea (continued)

2. optimal solution = | L | L | L | L | L | S | S | S | OPT = S+S+S...

3. whenever a small item is exposed, expose the last not-yet-exposed large item (call it COSTLY):



S X X X X X X L

1 1

4. REQUEST 1: Pr[request 1 small] is at most $\frac{\text{S+S+S}}{\text{sum of all probabilities}} = \frac{\text{OPT}}{\text{sum of all probabilities}}$

5. LEMMA: If request 1 is small, then the costly large item exposed has expected probability at most

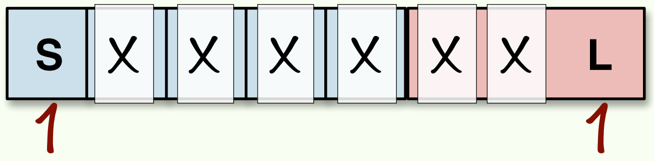
$$\text{average of large probabilities} = \frac{\text{sum of large probabilities}}{5}$$

THEOREM 2(a) upper bound: FCFS is $(1+H_{n-1})$-competitive.

proof idea (continued)

2. optimal solution = L L L L L S S S OPT = S+S+S...

3. whenever a small item is exposed, expose the last not-yet-exposed large item (call it COSTLY):



4. REQUEST 1: Pr[request 1 small] is at most $\dfrac{\text{S+S+S}}{\text{sum of all probabilities}} = \dfrac{\text{OPT}}{\text{sum of all probabilities}}$

5. LEMMA: If request 1 is small, then the costly large item exposed has expected probability at most

$$\text{average of large probabilities} = \frac{\text{sum of large probabilities}}{5}$$

6. IMPLIES: Expected contribution of step 1 to cost of COSTLY large items is at most

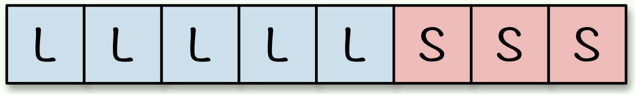
$$\text{Pr[item 1 small]} \times E[\text{revealed large item cost}] \leq \frac{\text{OPT}}{\text{sum of all probabilities}} \times \frac{\text{sum of large probabilities}}{5}$$

$$\leq \frac{\text{OPT}}{5}$$

THEOREM 2(a) upper bound: FCFS is $(1+H_{n-1})$-competitive.

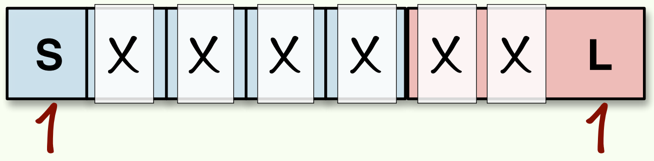
proof idea
(continued)

2. optimal solution =



OPT = S+S+S...

3. whenever a small item is exposed, expose the last not-yet-exposed large item (call it COSTLY):



4. REQUEST 1: Pr[request 1 small] is at most $\frac{\text{S+S+S}}{\text{sum of all probabilities}} = \frac{\text{OPT}}{\text{sum of all probabilities}}$

5. LEMMA: If request 1 is small, then the costly large item exposed has expected probability at most

$$\text{average of large probabilities} = \frac{\text{sum of large probabilities}}{5}$$

6. IMPLIES: Expected contribution of step 1 to cost of COSTLY large items is at most

$$\text{Pr[item 1 small]} \times E\text{[revealed large item cost]} \leq \frac{\text{OPT}}{\text{sum of all probabilities}} \times \frac{\text{sum of large probabilities}}{5}$$

$$\leq \frac{\text{OPT}}{5}$$

7. Second step: $\frac{\text{OPT}}{4}$; third: $\frac{\text{OPT}}{3}$; fourth: $\frac{\text{OPT}}{2}$; fifth $\frac{\text{OPT}}{1}$. Total $H_5 \times \text{OPT}$. QED ?

summary

THM 1: FcFS is optimally competitive for online Slot Allocation.

THM 2: optimal competitive ratios:

(a) Arbitrary slot costs: $1+H_{n-1}$

(b) concave slot costs: 2

(c) Logarithmic slot costs: 1^*

*asymptotically: FcFS guarantees cost OPT + O(log OPT)..

THM 3: For Huffman coding, online algorithm with cost

$$\mathrm{OPT} + 2\log_2(1 + \mathrm{OPT}) + 2.$$

sampling w/o replacement for poker tournaments

valuing chips = estimating, for sampling without replacement:

Pr[item i ends in slot j]

sampling w/o replacement for poker tournaments

valuing chips = estimating, for sampling without replacement:

Pr[item i ends in slot j]

your expected final payout, given your current chips:

```
      Pr[   first place ] * (payout for first place)
+     Pr[second place ] * (payout for second place)
+     ...
+     Pr[    last place ] * (payout for last place)
```

sampling w/o replacement for poker tournaments

valuing chips = estimating, for sampling without replacement:

Pr[item i ends in slot j]

your expected final payout, given your current chips:

Pr[first place] * (payout for first place)
+ Pr[second place] * (payout for second place)
+ …
+ Pr[last place] * (payout for last place)

random model for your final placement, given current chips:

- round 1: select first-place player by random draw where

  Pr[player i wins first place] = players chips / total chips

- round 2: select second-place player from REMAINING players, again

  Pr[player i wins second place] = players chips / total chips of remaining players

- etc… = sampling players without replacement, using current chip counts as probabilities

You finish in j'th place in tournament <==> You are the j'th sample

GIVEN: slots with costs $c(1) \le c(2) \le \cdots \le c(n)$, requests i

ALLOCATE: slot j

OBJECTIVE: Minimize cost $\sum_{i=1}^{n} p_i \, c(j_i)$

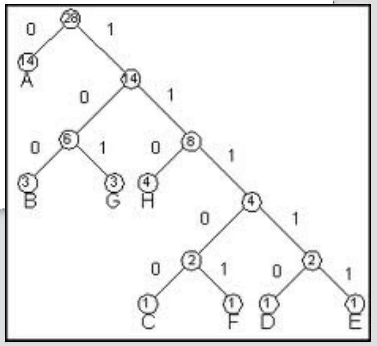
defn of OSA

DEFN OF ONLINE HUFFMAN CODING

GIVEN: letters i_1, i_2, i_3, $\cdots$ i.i.d. from unknown distribution p.

ALLOCATE: codeword j_i for each letter i on first occurrence.

OBJECTIVE: Minimize cost $\sum_{i=1}^{n} p_i \, c(j_i)$ $\quad (c(j) \approx \log_2 j)$



First-Come First-Served (FCFS) for Online Slot Allocation (OSA) and Huffman Coding

— SODA 2014 —

Monik Khare

yellowpages.com

Claire Mathieu

Ecole Normale Superieure

Neal E. Young

University of California
Riverside