

Introduction to Probability

Lecture 12: Online Algorithms

Mateja Jamnik, [Thomas Sauerwald](#)

University of Cambridge, Department of Computer Science and Technology
email: {mateja.jamnik,thomas.sauerwald}@cl.cam.ac.uk

Easter 2024



Stopping Problem 1: Dice Game

Stopping Problem 2: The Secretary Problem

A Generalisation: The Odds Algorithm (non-examinable)

The End...

Introduction: Dice Game



Dice Game

Introduction: Dice Game



Dice Game

- We throw a fair, six-sided dice n times

Introduction: Dice Game



Dice Game

- We throw a fair, six-sided dice n times
- After each throw, you can either STOP or CONTINUE
- You win if you STOP at the last 6 within the n throws

Introduction: Dice Game



Dice Game

- We throw a fair, six-sided dice n times
- After each throw, you can either STOP or CONTINUE
- You win if you STOP at the last 6 within the n throws

Example ($n = 10$)

Introduction: Dice Game



Dice Game

- We throw a fair, six-sided dice n times
- After each throw, you can either STOP or CONTINUE
- You win if you STOP at the last 6 within the n throws

Example ($n = 10$)

- 3, 5, 6, 4, 2, 3, 1, 2, 6, 5
STOP

Introduction: Dice Game



Dice Game

- We throw a fair, six-sided dice n times
- After each throw, you can either STOP or CONTINUE
- You win if you STOP at the last 6 within the n throws

Example ($n = 10$)

- 3, 5, 6, 4, 2, 3, 1, 2, 6, 5 \Rightarrow LOSE!
STOP

Introduction: Dice Game



Dice Game

- We throw a fair, six-sided dice n times
- After each throw, you can either STOP or CONTINUE
- You win if you STOP at the last 6 within the n throws

Example ($n = 10$)

- 3, 5, 6, 4, 2, 3, 1, 2, 6, 5 \Rightarrow LOSE!
STOP
- 3, 5, 6, 4, 2, 3, 1, 2, 6, 5
STOP

Introduction: Dice Game



Dice Game

- We throw a fair, six-sided dice n times
- After each throw, you can either STOP or CONTINUE
- You win if you STOP at the last 6 within the n throws

Example ($n = 10$)

- 3, 5, 6, 4, 2, 3, 1, 2, 6, 5 \Rightarrow LOSE!
STOP
- 3, 5, 6, 4, 2, 3, 1, 2, 6, 5 \Rightarrow LOSE!
STOP
- 3, 5, 6, 4, 2, 3, 1, 2, 6, 5
STOP

Introduction: Dice Game



Dice Game

- We throw a fair, six-sided dice n times
- After each throw, you can either STOP or CONTINUE
- You win if you STOP at the last 6 within the n throws

Example ($n = 10$)

- 3, 5, 6, 4, 2, 3, 1, 2, 6, 5 \Rightarrow LOSE!
STOP
- 3, 5, 6, 4, 2, 3, 1, 2, 6, 5 \Rightarrow LOSE!
STOP
- 3, 5, 6, 4, 2, 3, 1, 2, 6, 5 \Rightarrow WIN!
STOP

Introduction: Dice Game



Dice Game

- We throw a fair, six-sided dice n times
- After each throw, you can either STOP or CONTINUE
- You win if you STOP at the last 6 within the n throws

What is the optimal strategy for maximising the probability of winning?

Example ($n = 10$)

- 3, 5, 6, 4, 2, 3, 1, 2, 6, 5 \Rightarrow LOSE!
STOP
- 3, 5, 6, 4, 2, 3, 1, 2, 6, 5 \Rightarrow LOSE!
STOP
- 3, 5, 6, 4, 2, 3, 1, 2, 6, 5 \Rightarrow WIN!
STOP

Introduction: Dice Game



Dice Game

- We throw a fair, six-sided dice n times
- After each throw, you can either STOP or CONTINUE
- You win if you STOP at the last 6 within the n throws

What is the optimal strategy for maximising the probability of winning?

Example ($n = 10$)

- 3, 5, 6, 4, 2, 3, 1, 2, 6, 5 \Rightarrow LOSE!
STOP
- 3, 5, 6, 4, 2, 3, 1, 2, 6, 5 \Rightarrow LOSE!
STOP
- 3, 5, 6, 4, 2, 3, 1, 2, 6, 5 \Rightarrow WIN!
STOP

This boils down to finding a threshold from which we STOP as soon as a 6 is thrown.

Dice Game (Solution)

\mathbf{P} [obtain exactly one 6 in last k throws] =

Dice Game (Solution)

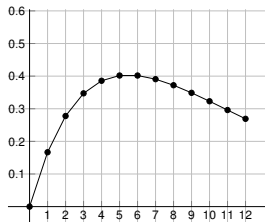
$$\mathbf{P}[\text{obtain exactly one 6 in last } k \text{ throws}] = \binom{k}{1} \cdot \frac{1}{6} \cdot \left(\frac{5}{6}\right)^{k-1}$$

Dice Game (Solution)

$$\mathbf{P}[\text{obtain exactly one 6 in last } k \text{ throws}] = \binom{k}{1} \cdot \frac{1}{6} \cdot \left(\frac{5}{6}\right)^{k-1} = \frac{k}{6} \cdot \left(\frac{5}{6}\right)^{k-1}$$

Dice Game (Solution)

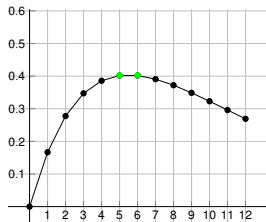
$$\mathbf{P}[\text{obtain exactly one 6 in last } k \text{ throws}] = \binom{k}{1} \cdot \frac{1}{6} \cdot \left(\frac{5}{6}\right)^{k-1} = \frac{k}{6} \cdot \left(\frac{5}{6}\right)^{k-1}$$



We obtain a **unimodal** distribution

Dice Game (Solution)

$$\mathbf{P}[\text{obtain exactly one 6 in last } k \text{ throws}] = \binom{k}{1} \cdot \frac{1}{6} \cdot \left(\frac{5}{6}\right)^{k-1} = \frac{k}{6} \cdot \left(\frac{5}{6}\right)^{k-1}$$

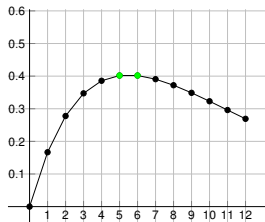


We obtain a **unimodal** distribution

- This is **maximised** for $k = 6$ (or $k = 5$)

Dice Game (Solution)

$$\mathbf{P}[\text{obtain exactly one 6 in last } k \text{ throws}] = \binom{k}{1} \cdot \frac{1}{6} \cdot \left(\frac{5}{6}\right)^{k-1} = \frac{k}{6} \cdot \left(\frac{5}{6}\right)^{k-1}$$

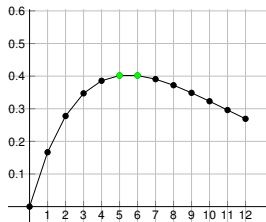


We obtain a **unimodal** distribution

- This is **maximised** for $k = 6$ (or $k = 5$) \Rightarrow **best strategy**: wait until we have 6 (5) throws left, and then STOP at the first 6

Dice Game (Solution)

$$\mathbf{P}[\text{obtain exactly one 6 in last } k \text{ throws}] = \binom{k}{1} \cdot \frac{1}{6} \cdot \left(\frac{5}{6}\right)^{k-1} = \frac{k}{6} \cdot \left(\frac{5}{6}\right)^{k-1}$$

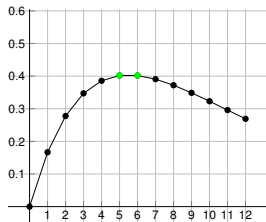


We obtain a **unimodal** distribution

- This is **maximised** for $k = 6$ (or $k = 5$) \Rightarrow **best strategy**: wait until we have 6 (5) throws left, and then STOP at the first 6
- **Probability of success** is:

Dice Game (Solution)

$$\mathbf{P}[\text{obtain exactly one 6 in last } k \text{ throws}] = \binom{k}{1} \cdot \frac{1}{6} \cdot \left(\frac{5}{6}\right)^{k-1} = \frac{k}{6} \cdot \left(\frac{5}{6}\right)^{k-1}$$



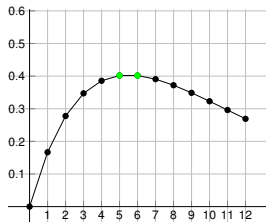
We obtain a **unimodal** distribution

- This is **maximised** for $k = 6$ (or $k = 5$) \Rightarrow **best strategy**: wait until we have 6 (5) throws left, and then STOP at the first 6
- **Probability of success** is:

$$\left(\frac{5}{6}\right)^5$$

Dice Game (Solution)

$$\mathbf{P}[\text{obtain exactly one 6 in last } k \text{ throws}] = \binom{k}{1} \cdot \frac{1}{6} \cdot \left(\frac{5}{6}\right)^{k-1} = \frac{k}{6} \cdot \left(\frac{5}{6}\right)^{k-1}$$



We obtain a **unimodal** distribution

- This is **maximised** for $k = 6$ (or $k = 5$) \Rightarrow **best strategy**: wait until we have 6 (5) throws left, and then STOP at the first 6
- **Probability of success** is:

$$\left(\frac{5}{6}\right)^5 \approx 0.40.$$

Illustration of the Three Possibilities

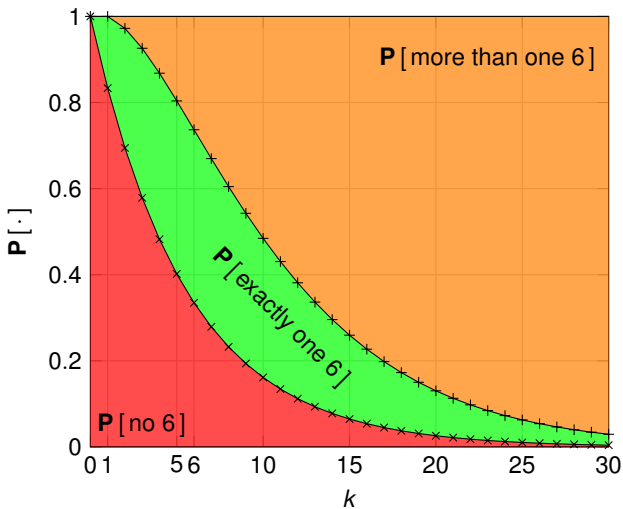
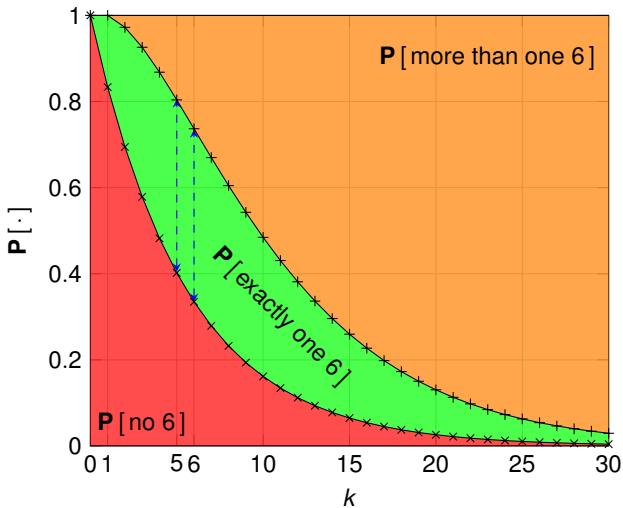


Illustration of the Three Possibilities



Stopping Problem 1: Dice Game

Stopping Problem 2: The Secretary Problem

A Generalisation: The Odds Algorithm (non-examinable)

The End...

The Secretary Problem

The Problem

- We are interviewing n candidates for **one job** in a sequential, **random** order

The Secretary Problem

The Problem

- We are interviewing n candidates for **one job** in a sequential, **random** order
- A candidate must be accepted (STOP) or rejected **immediately** after the interview and cannot be recalled

The Secretary Problem

The Problem

- We are interviewing n candidates for **one job** in a sequential, **random** order
- A candidate must be accepted (STOP) or rejected **immediately** after the interview and cannot be recalled
- **Goal:** **maximise** the probability of hiring the **best** candidate

The Secretary Problem

The Problem

- We are interviewing n candidates for **one job** in a sequential, **random** order
- A candidate must be accepted (STOP) or rejected **immediately** after the interview and cannot be recalled
- **Goal:** **maximise** the probability of hiring the **best** candidate

also known as **marriage problem** (Kepler 1613),
hiring problem or **best choice problem**.

The Secretary Problem

The Problem

- We are interviewing n candidates for **one job** in a sequential, **random** order
- A candidate must be accepted (STOP) or rejected **immediately** after the interview and cannot be recalled
- **Goal:** **maximise** the probability of hiring the **best** candidate

also known as **marriage problem** (Kepler 1613),
hiring problem or **best choice problem**.

Further Remarks

The Secretary Problem

The Problem

- We are interviewing n candidates for **one job** in a sequential, **random** order
- A candidate must be accepted (STOP) or rejected **immediately** after the interview and cannot be recalled
- **Goal:** **maximise** the probability of hiring the **best** candidate

also known as **marriage problem** (Kepler 1613),
hiring problem or **best choice problem**.

Further Remarks

- After seeing candidate i , we only know the **relative order** among the first i candidates.

The Secretary Problem

The Problem

- We are interviewing n candidates for **one job** in a sequential, **random** order
- A candidate must be accepted (STOP) or rejected **immediately** after the interview and cannot be recalled
- **Goal:** **maximise** the probability of hiring the **best** candidate

also known as **marriage problem** (Kepler 1613),
hiring problem or **best choice problem**.

Further Remarks

- After seeing candidate i , we only know the **relative order** among the first i candidates.
- ⇒ For our problem we may as well assume that the only information we have when interviewing candidate i is whether that candidate is best among $\{1, \dots, i\}$ or not.

Illustration ($n = 20$)

unknown permutation:

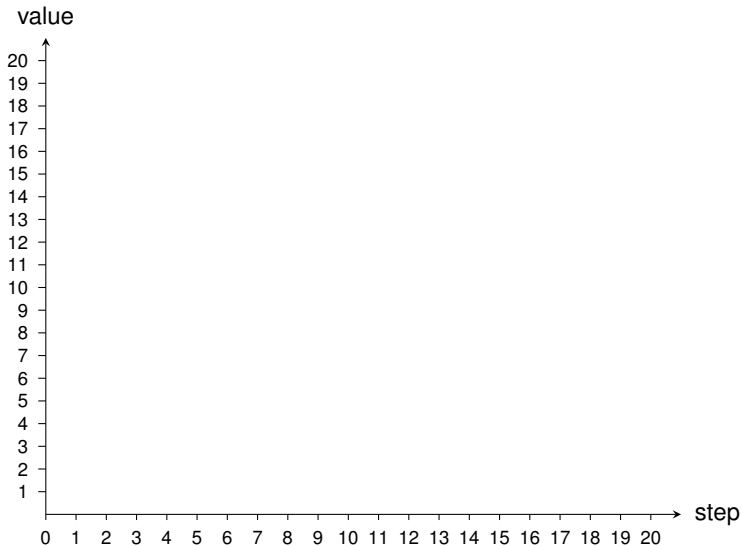


Illustration ($n = 20$)

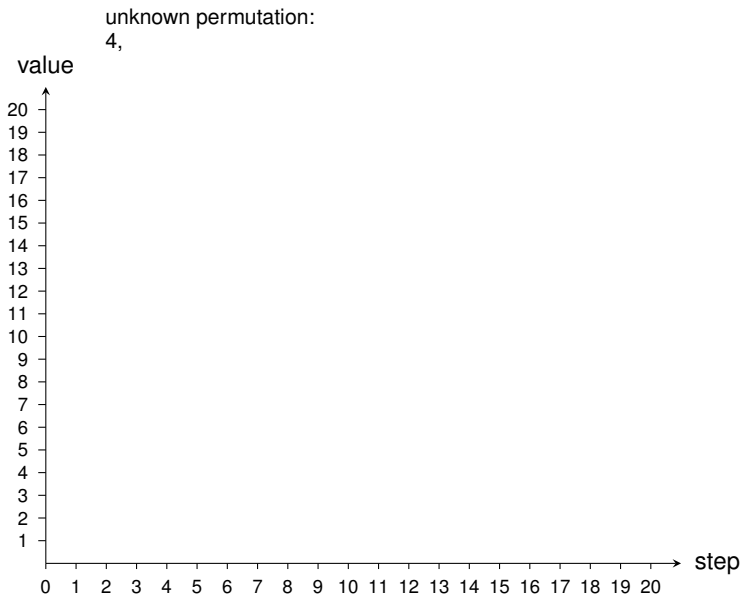


Illustration ($n = 20$)

unknown permutation:

4,

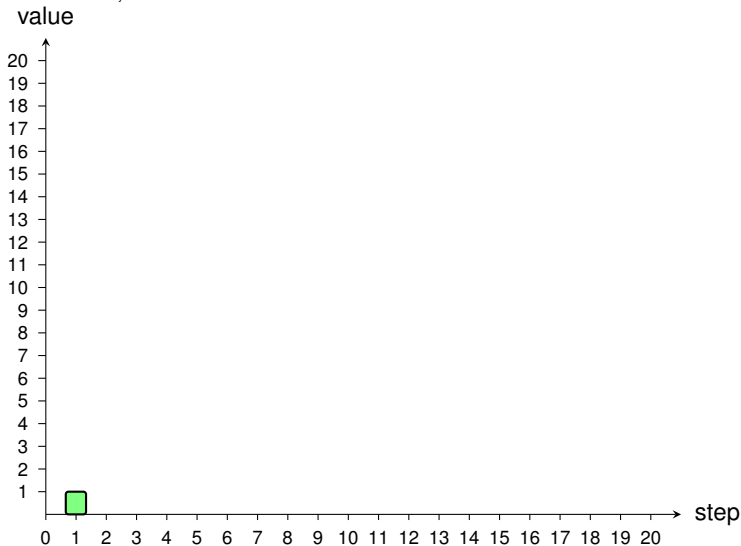


Illustration ($n = 20$)

unknown permutation:

4, 7,

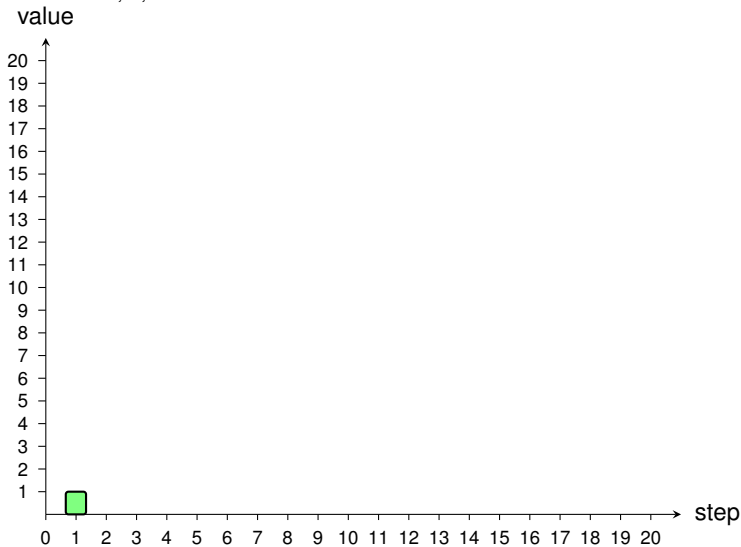


Illustration ($n = 20$)

unknown permutation:
4, 7,

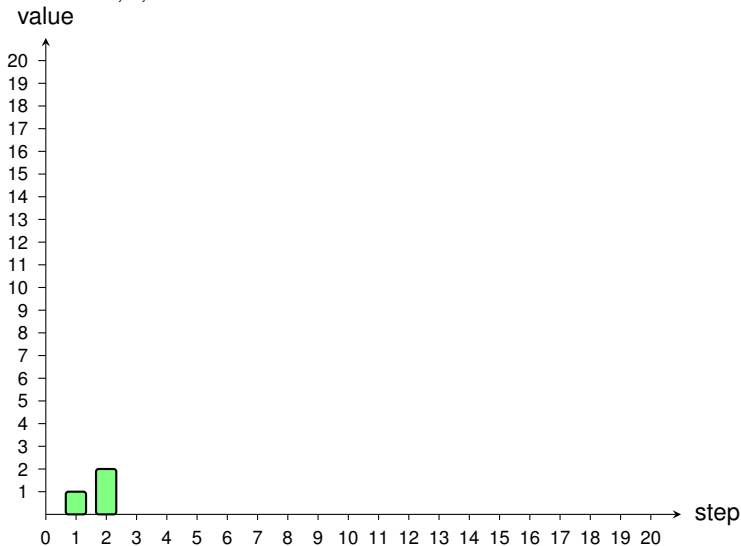


Illustration ($n = 20$)

unknown permutation:
4, 7, 8,

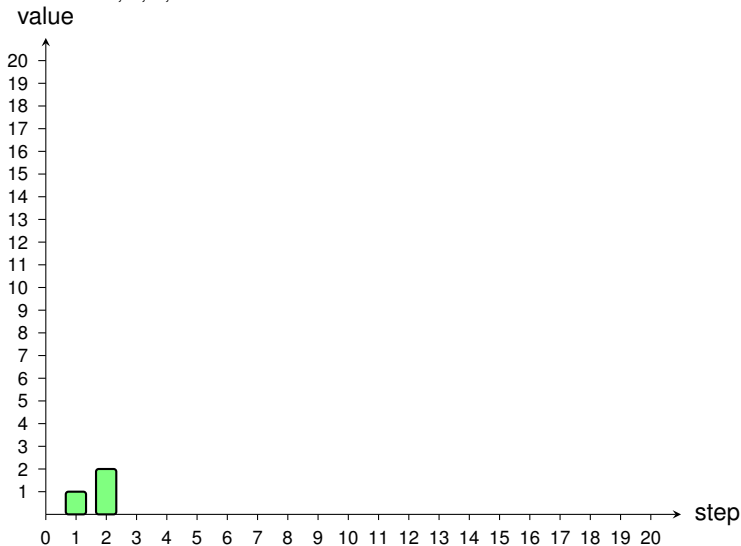


Illustration ($n = 20$)

unknown permutation:
4, 7, 8,

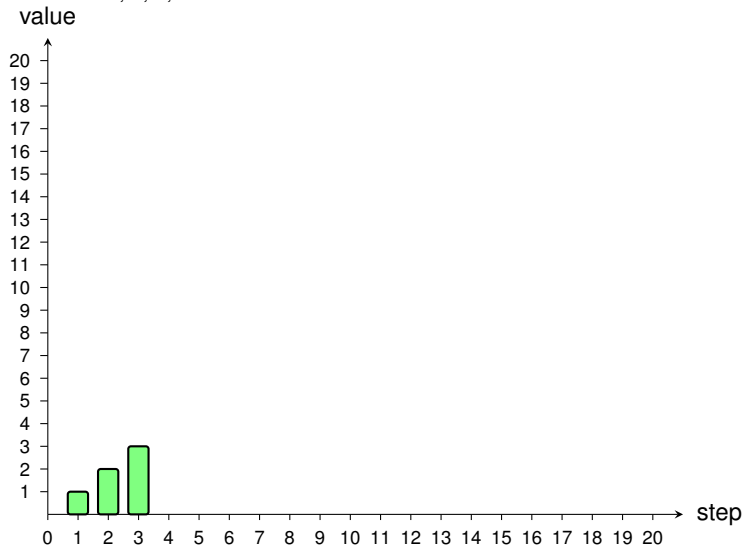


Illustration ($n = 20$)

unknown permutation:
4, 7, 8, 6,

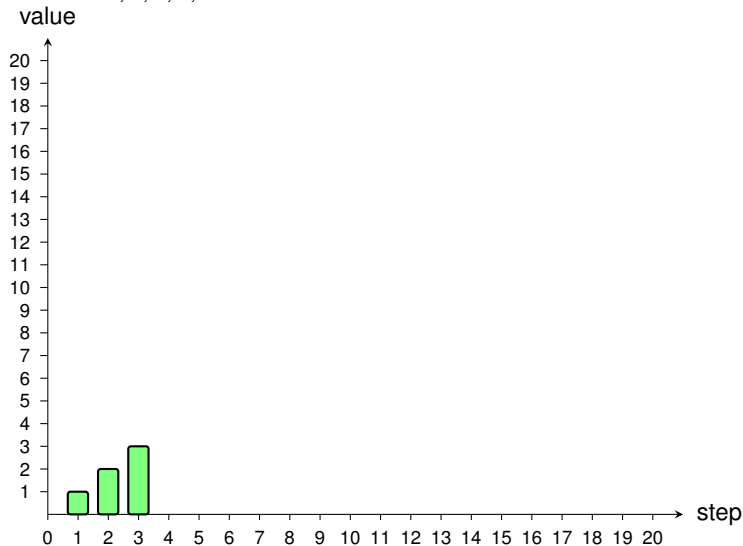


Illustration ($n = 20$)

unknown permutation:
4, 7, 8, 6,

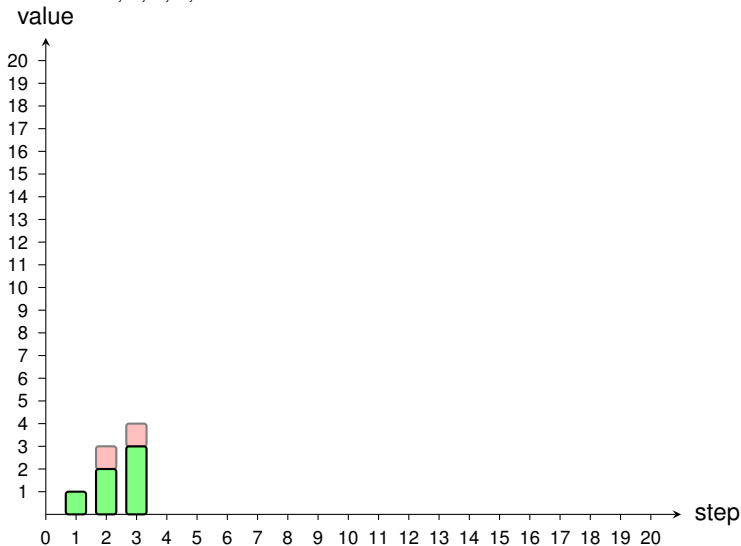


Illustration ($n = 20$)

unknown permutation:

4, 7, 8, 6,

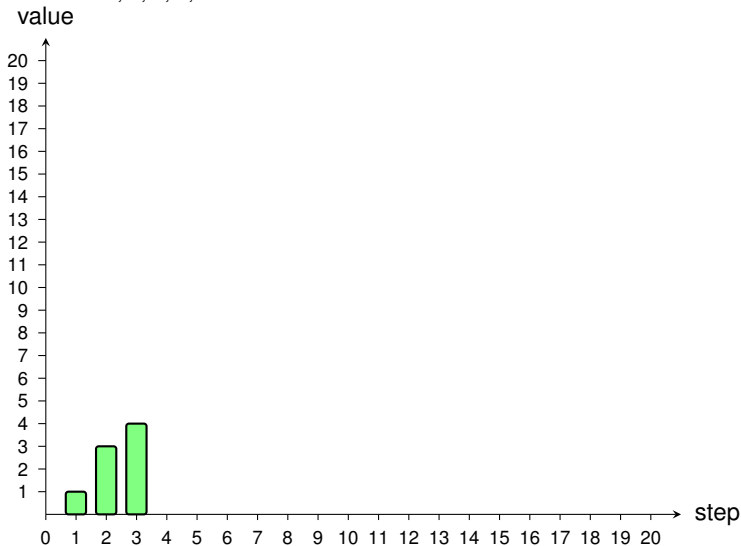


Illustration ($n = 20$)

unknown permutation:
4, 7, 8, 6,

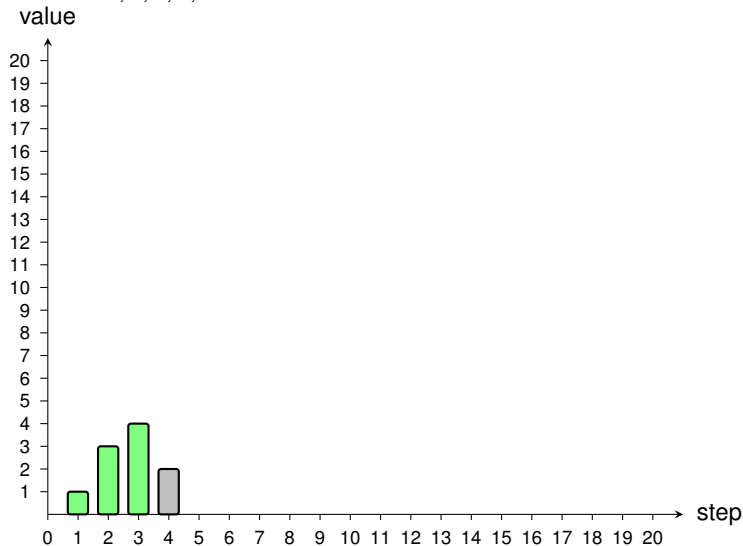


Illustration ($n = 20$)

unknown permutation:
4, 7, 8, 6, 18,

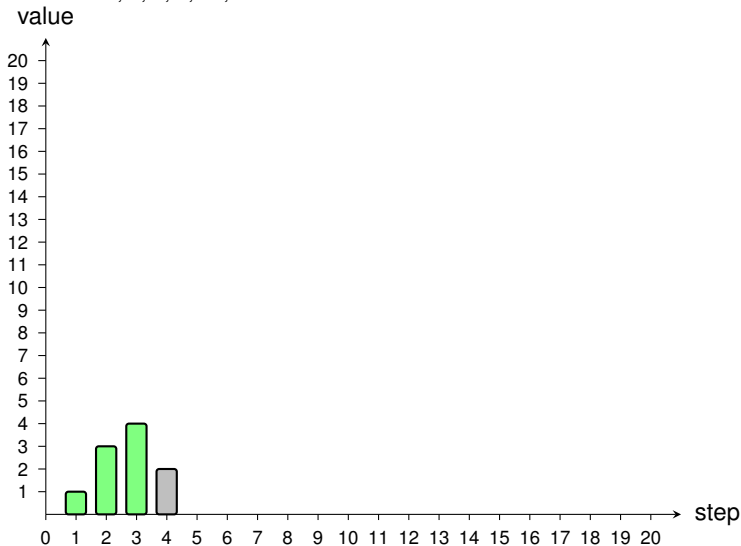


Illustration ($n = 20$)

unknown permutation:
4, 7, 8, 6, 18,

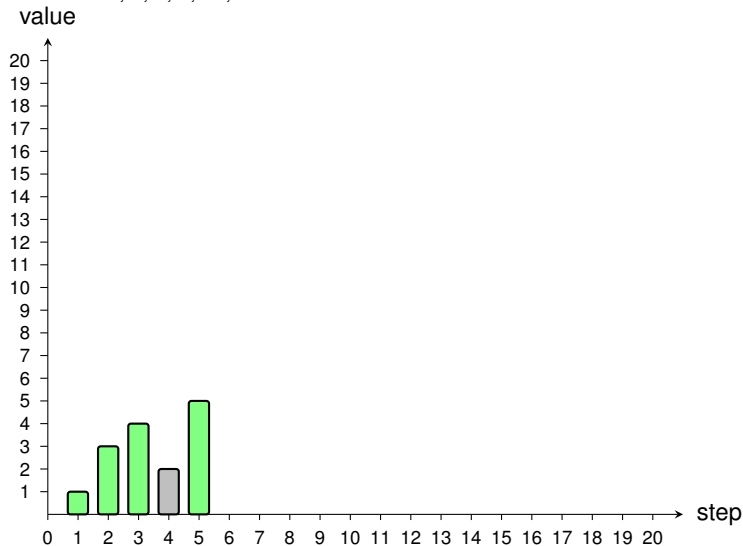


Illustration ($n = 20$)

unknown permutation:
4, 7, 8, 6, 18, 11,

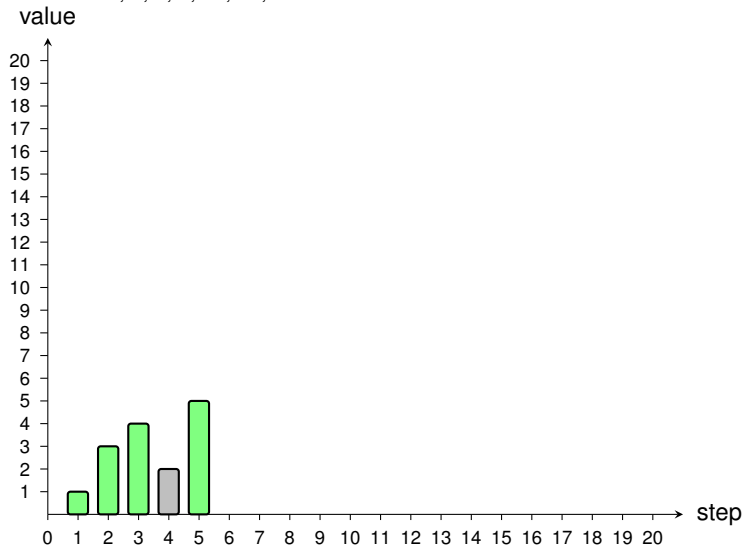


Illustration ($n = 20$)

unknown permutation:
4, 7, 8, 6, 18, 11,

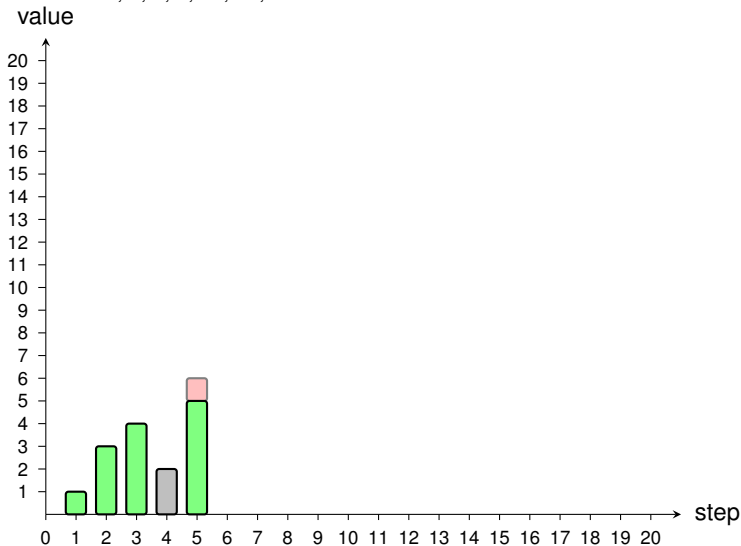


Illustration ($n = 20$)

unknown permutation:
4, 7, 8, 6, 18, 11,

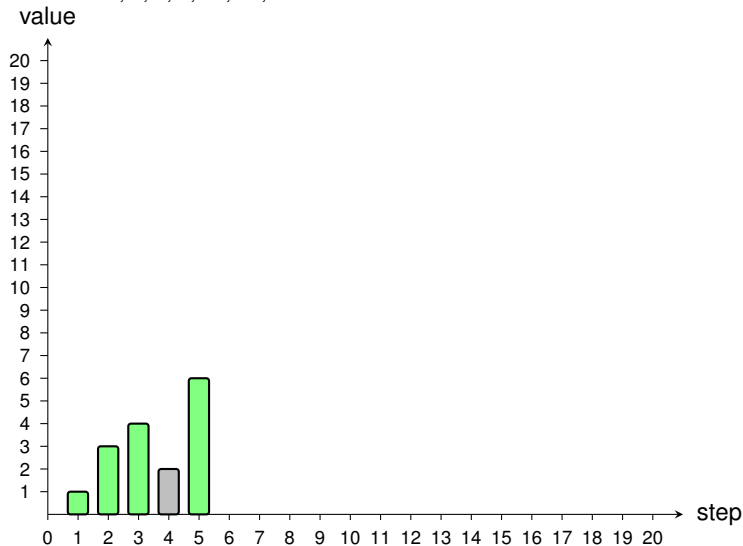


Illustration ($n = 20$)

unknown permutation:
4, 7, 8, 6, 18, 11,

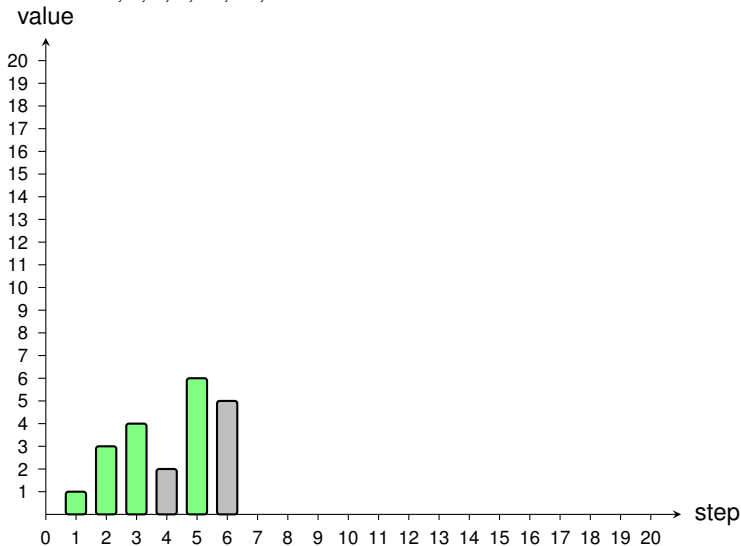


Illustration ($n = 20$)

unknown permutation:

4, 7, 8, 6, 18, 11, 3,

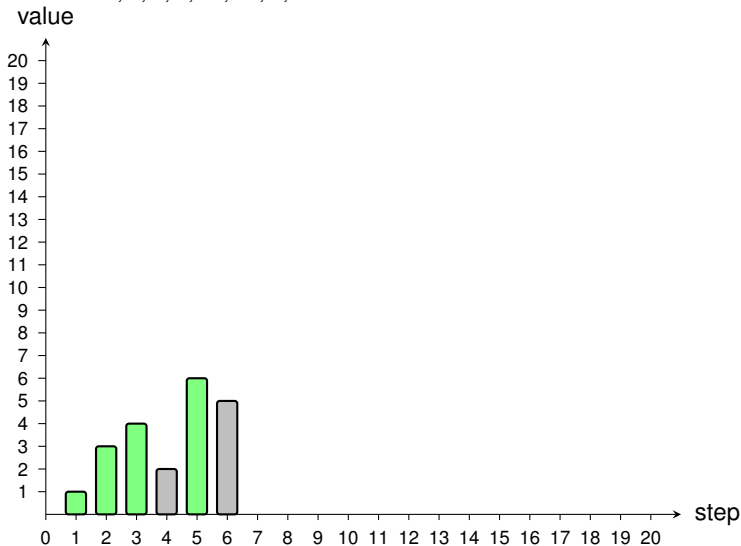


Illustration ($n = 20$)

unknown permutation:
4, 7, 8, 6, 18, 11, 3,

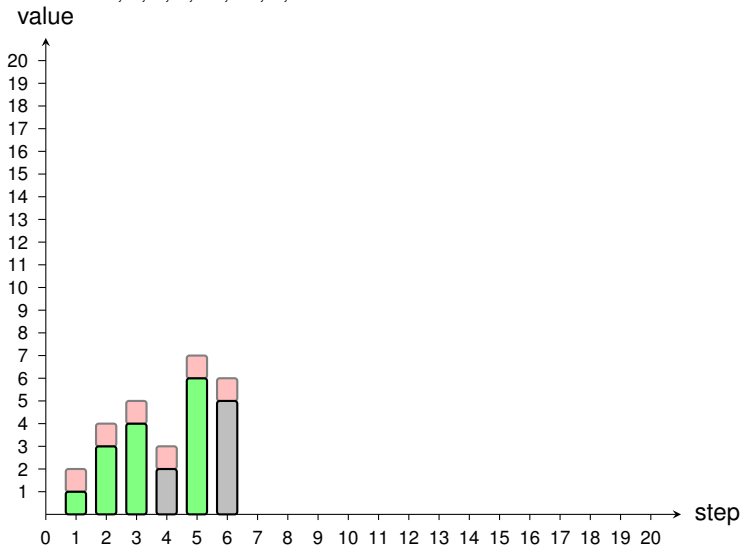


Illustration ($n = 20$)

unknown permutation:

4, 7, 8, 6, 18, 11, 3,

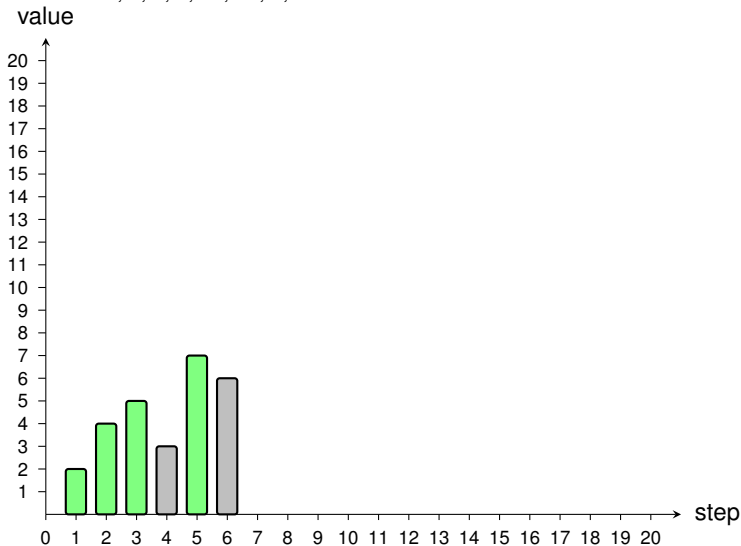


Illustration ($n = 20$)

unknown permutation:

4, 7, 8, 6, 18, 11, 3,

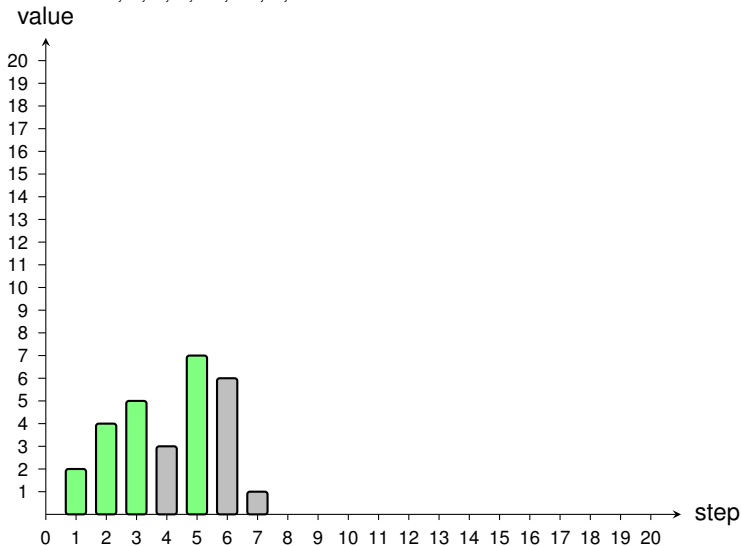


Illustration ($n = 20$)

unknown permutation:
4, 7, 8, 6, 18, 11, 3, 5,

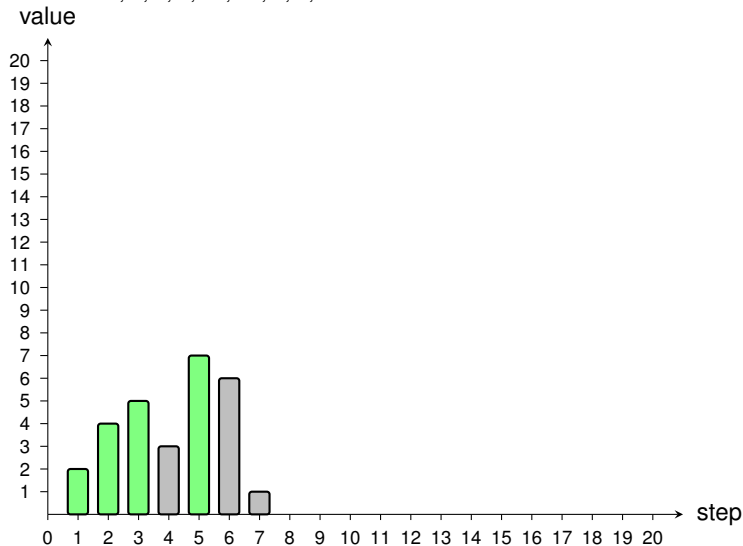


Illustration ($n = 20$)

unknown permutation:
4, 7, 8, 6, 18, 11, 3, 5,

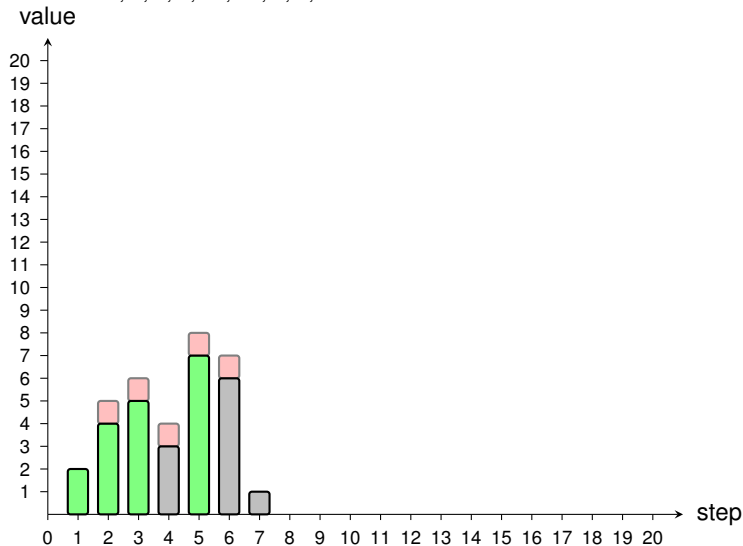


Illustration ($n = 20$)

unknown permutation:
4, 7, 8, 6, 18, 11, 3, 5,

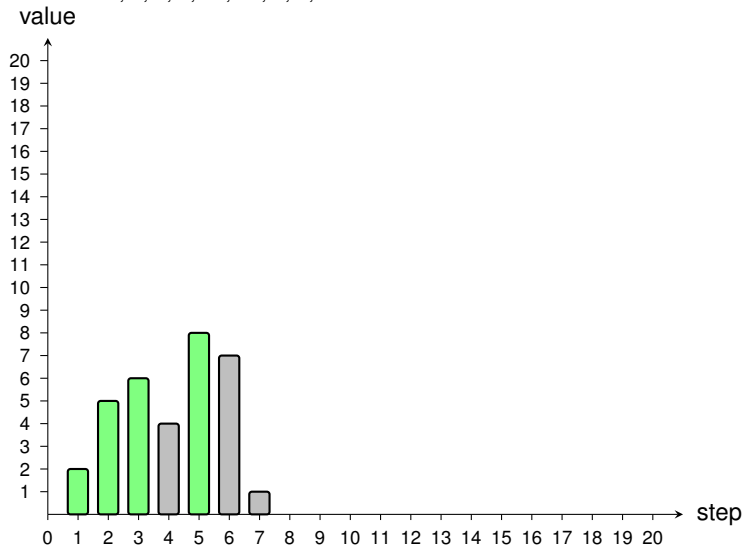


Illustration ($n = 20$)

unknown permutation:
4, 7, 8, 6, 18, 11, 3, 5,

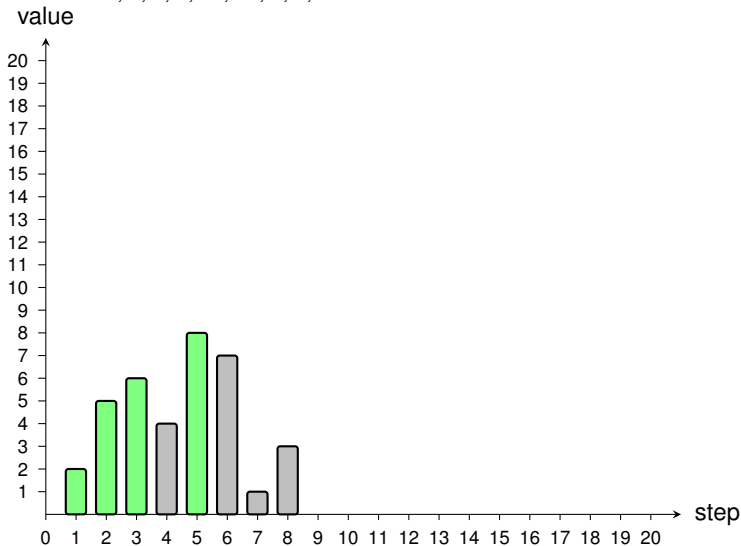


Illustration ($n = 20$)

unknown permutation:
4, 7, 8, 6, 18, 11, 3, 5, 9,

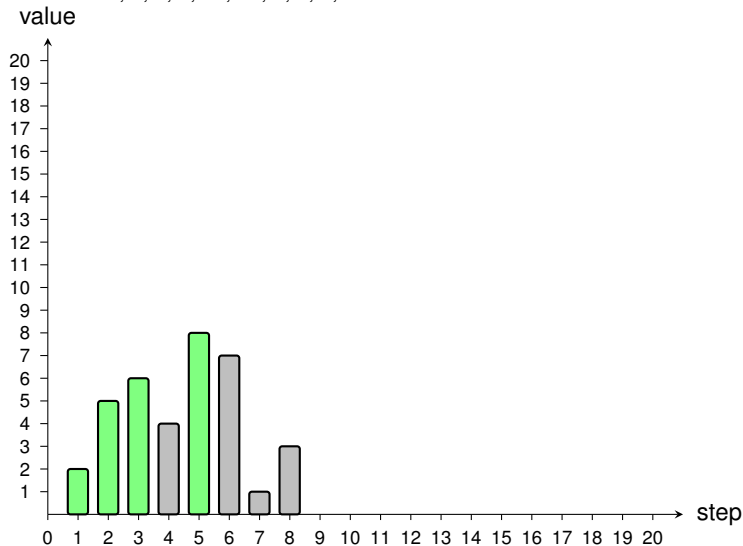


Illustration ($n = 20$)

unknown permutation:
4, 7, 8, 6, 18, 11, 3, 5, 9,

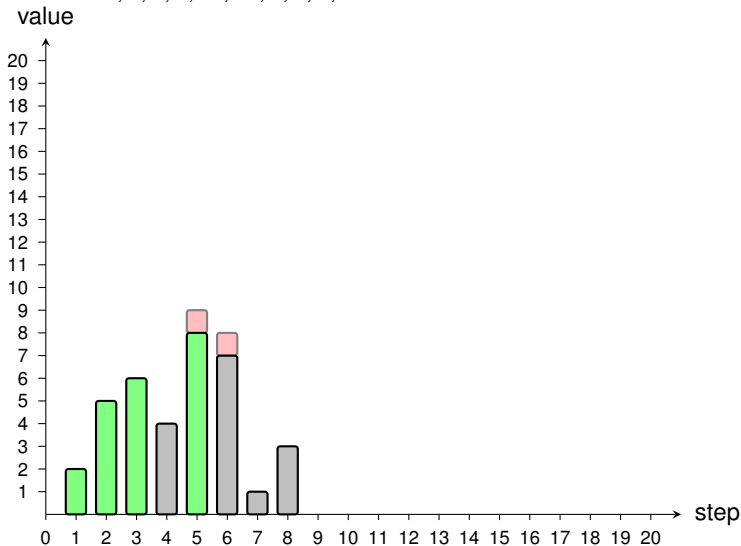


Illustration ($n = 20$)

unknown permutation:
4, 7, 8, 6, 18, 11, 3, 5, 9,

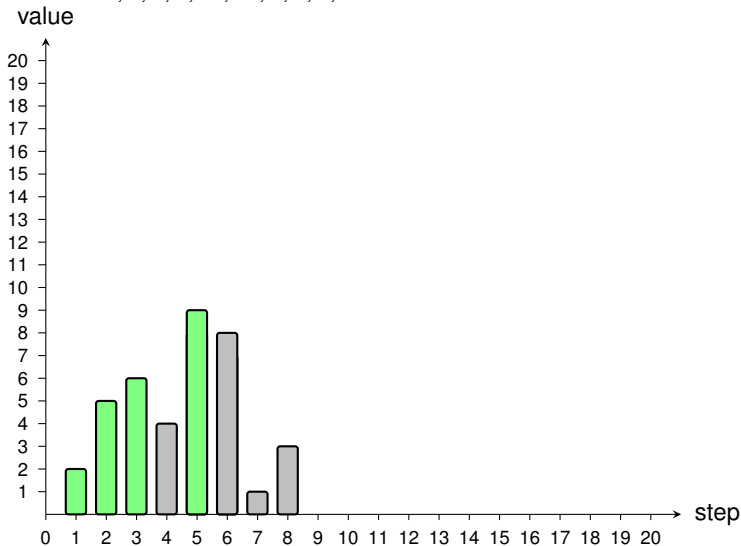


Illustration ($n = 20$)

unknown permutation:
4, 7, 8, 6, 18, 11, 3, 5, 9,

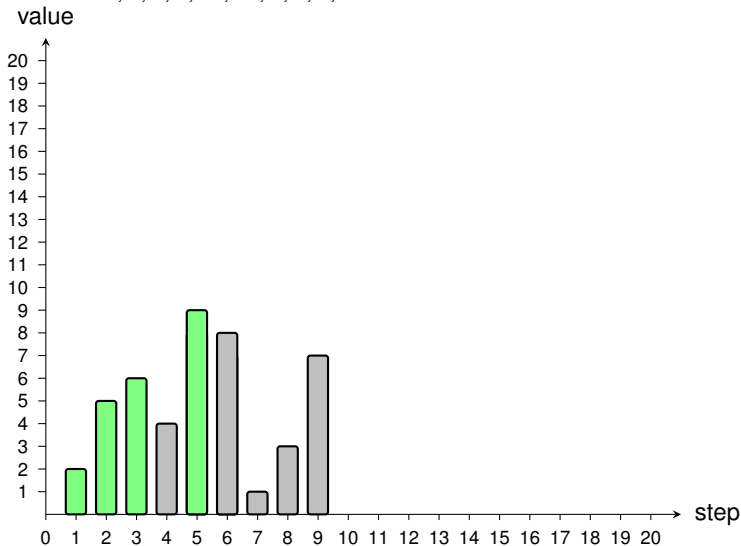


Illustration ($n = 20$)

unknown permutation:

4, 7, 8, 6, 18, 11, 3, 5, 9, 13,

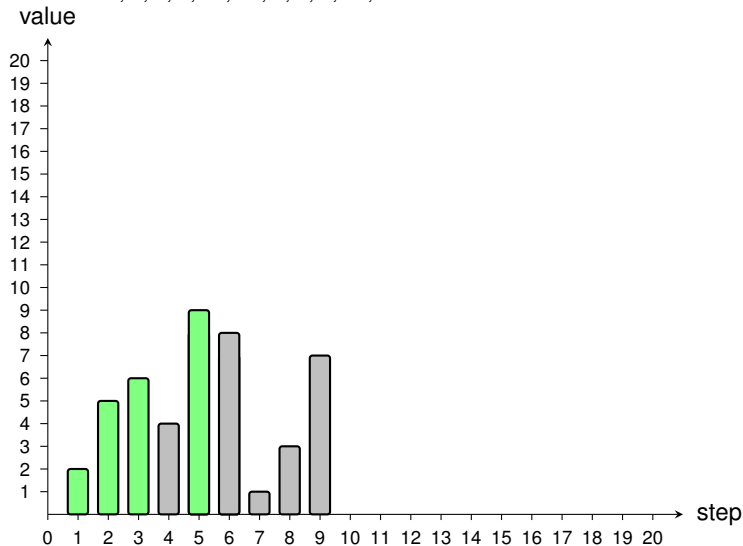


Illustration ($n = 20$)

unknown permutation:

4, 7, 8, 6, 18, 11, 3, 5, 9, 13,

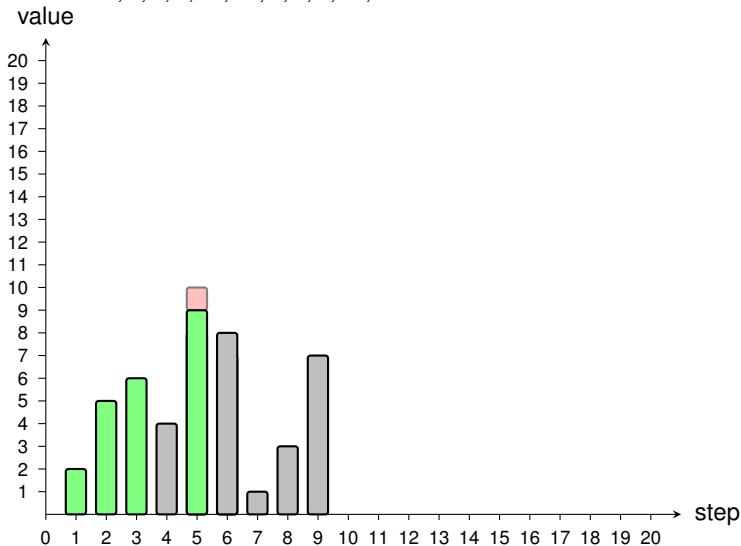


Illustration ($n = 20$)

unknown permutation:

4, 7, 8, 6, 18, 11, 3, 5, 9, 13,

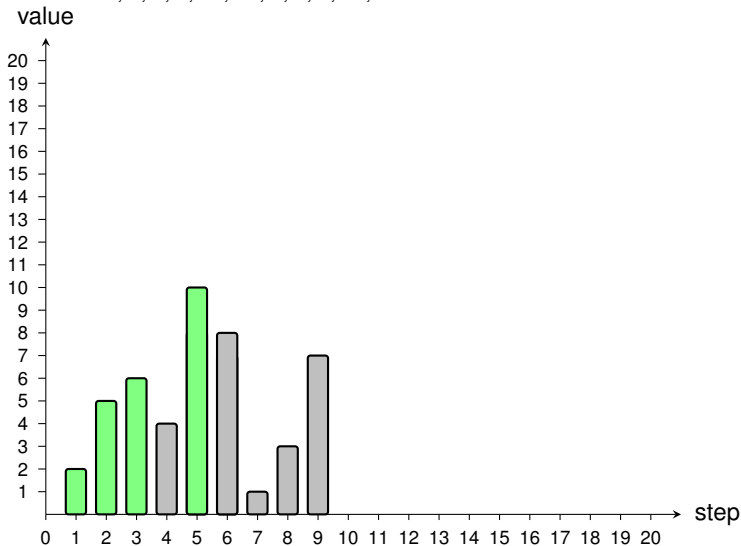


Illustration ($n = 20$)

unknown permutation:

4, 7, 8, 6, 18, 11, 3, 5, 9, 13,

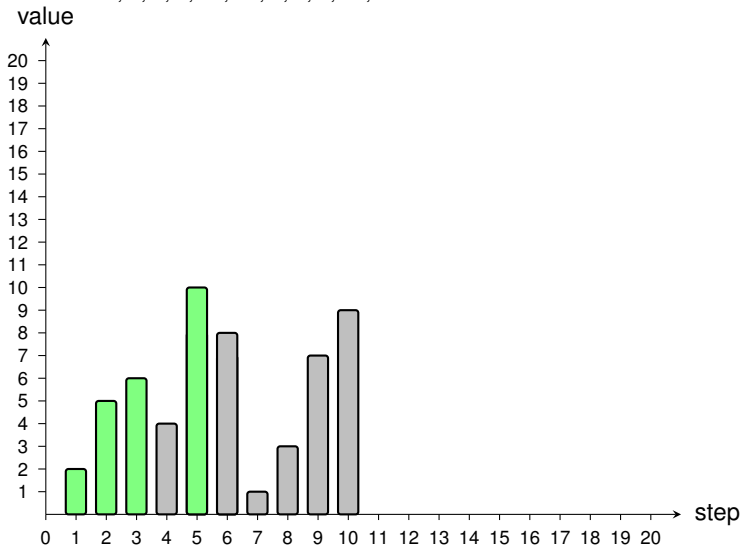


Illustration ($n = 20$)

unknown permutation:

4, 7, 8, 6, 18, 11, 3, 5, 9, 13, 17,

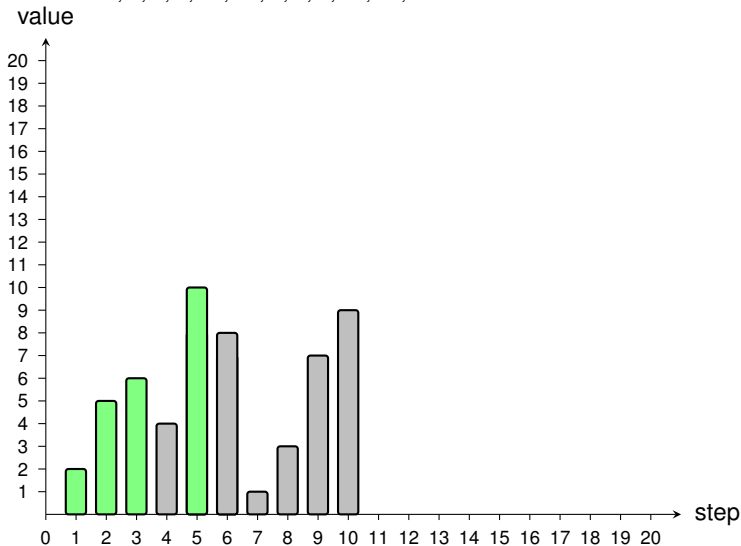


Illustration ($n = 20$)

unknown permutation:

4, 7, 8, 6, 18, 11, 3, 5, 9, 13, 17,

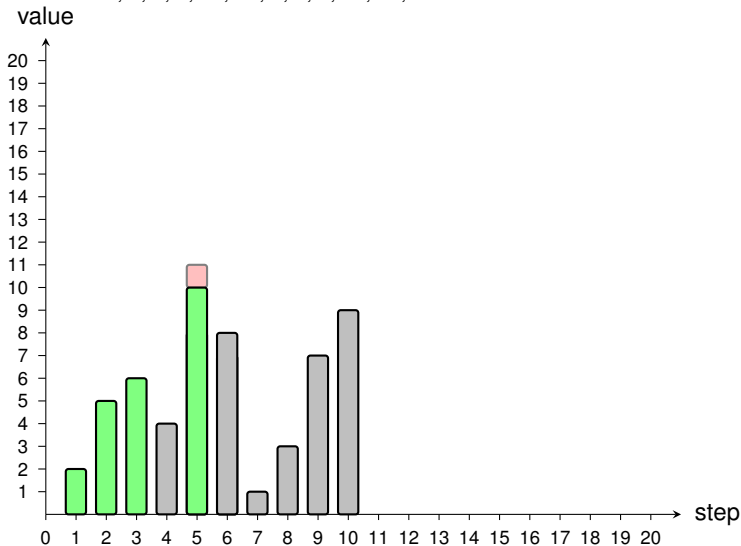


Illustration ($n = 20$)

unknown permutation:

4, 7, 8, 6, 18, 11, 3, 5, 9, 13, 17,

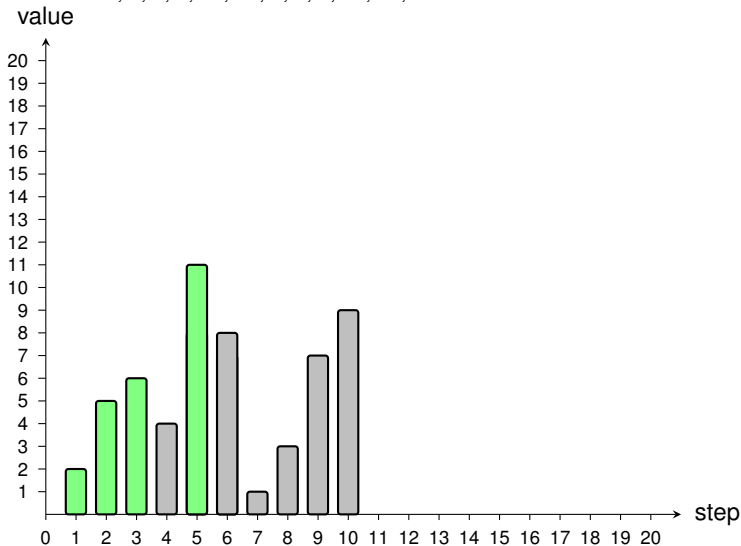


Illustration ($n = 20$)

unknown permutation:

4, 7, 8, 6, 18, 11, 3, 5, 9, 13, 17,

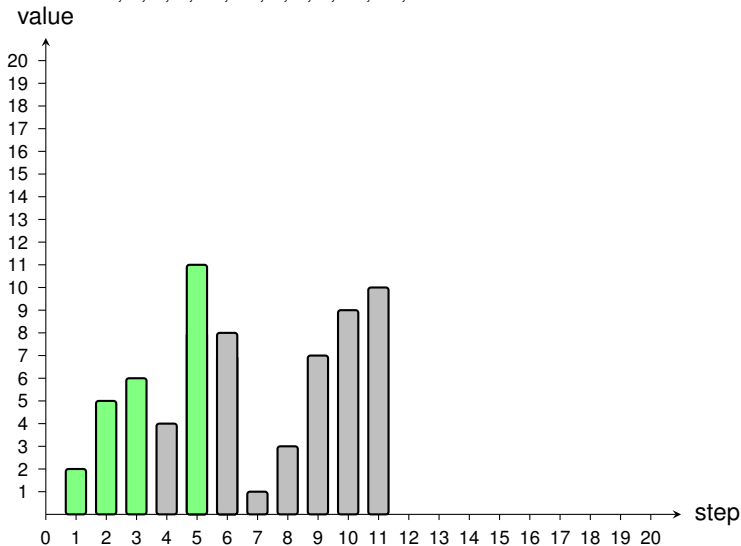


Illustration ($n = 20$)

unknown permutation:

4, 7, 8, 6, 18, 11, 3, 5, 9, 13, 17, 2,

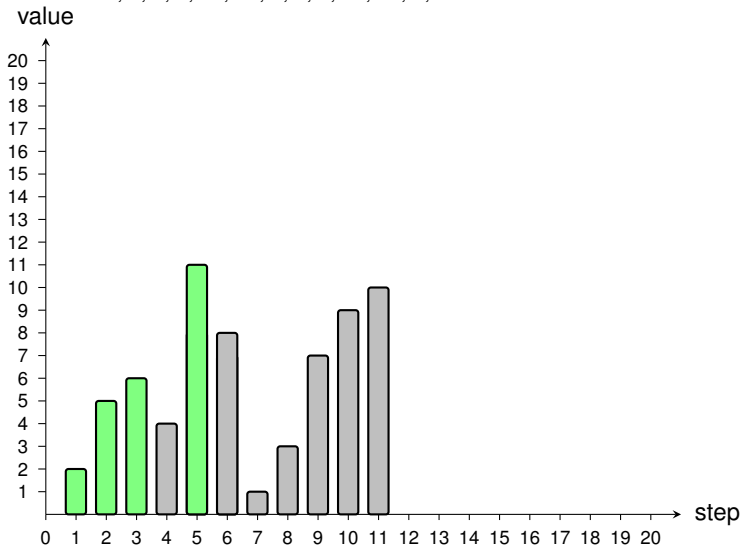


Illustration ($n = 20$)

unknown permutation:

4, 7, 8, 6, 18, 11, 3, 5, 9, 13, 17, 2,

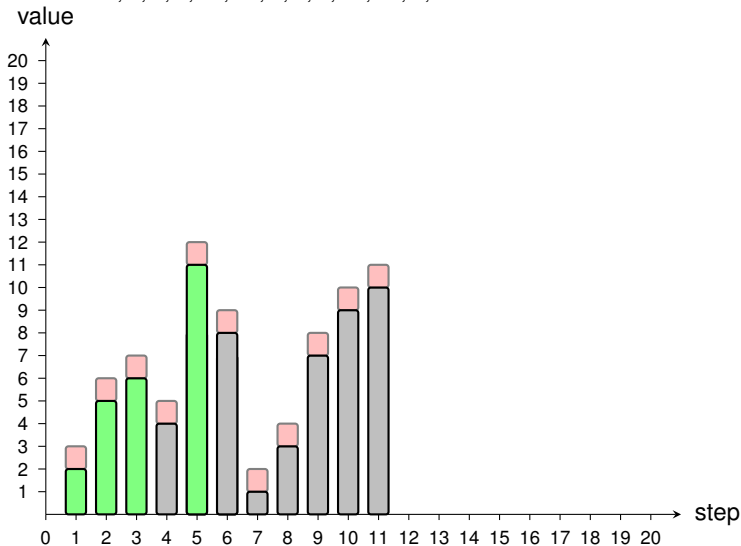


Illustration ($n = 20$)

unknown permutation:

4, 7, 8, 6, 18, 11, 3, 5, 9, 13, 17, 2,

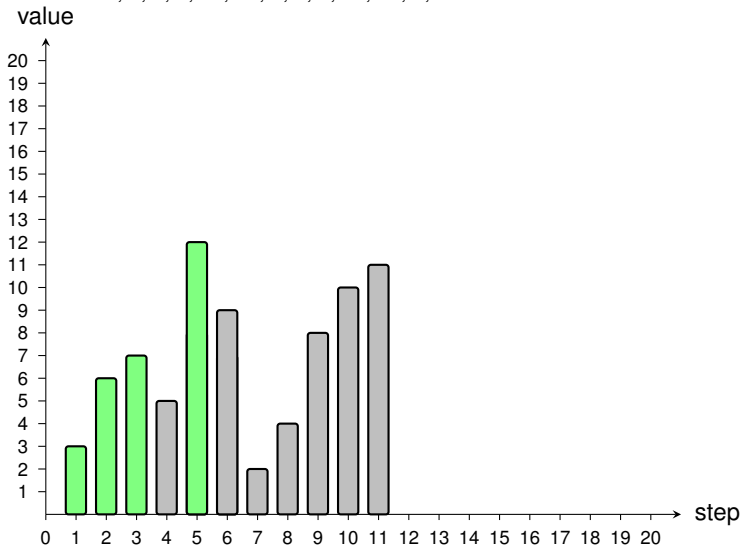


Illustration ($n = 20$)

unknown permutation:

4, 7, 8, 6, 18, 11, 3, 5, 9, 13, 17, 2,

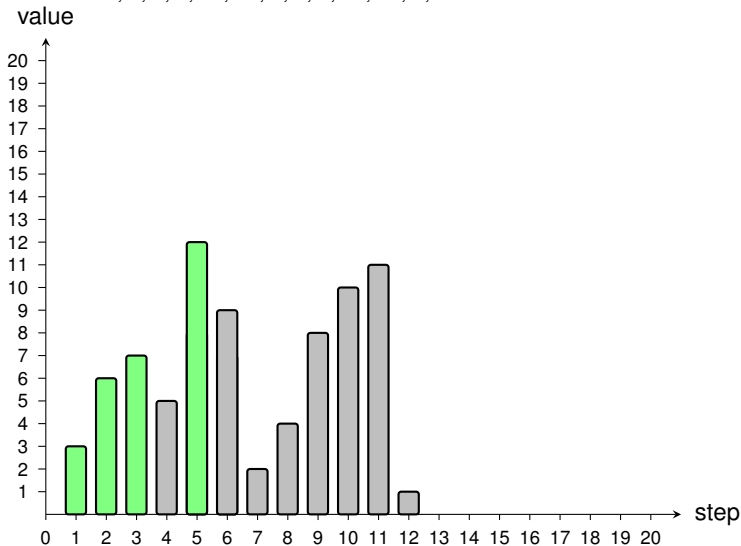


Illustration ($n = 20$)

unknown permutation:

4, 7, 8, 6, 18, 11, 3, 5, 9, 13, 17, 2, 20,

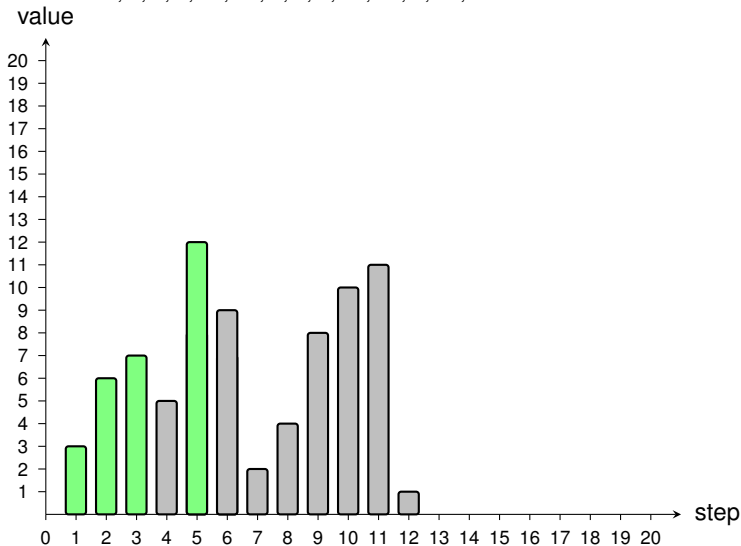


Illustration ($n = 20$)

unknown permutation:

4, 7, 8, 6, 18, 11, 3, 5, 9, 13, 17, 2, 20,

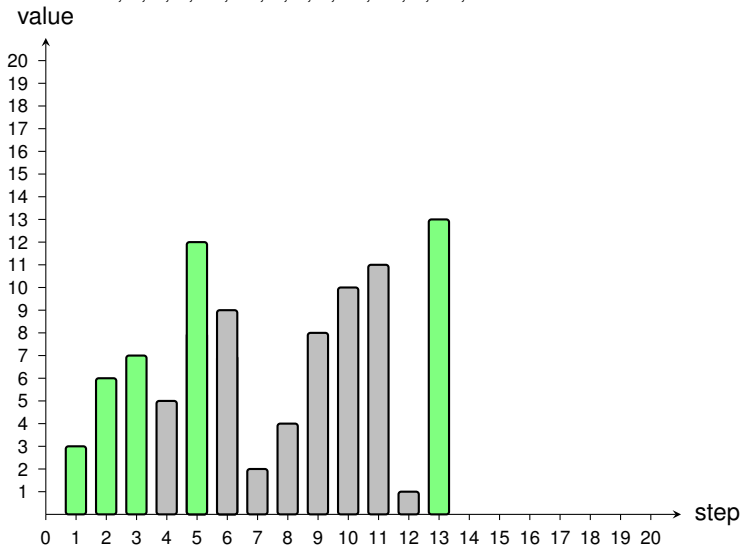


Illustration ($n = 20$)

unknown permutation:

4, 7, 8, 6, 18, 11, 3, 5, 9, 13, 17, 2, 20, 14,

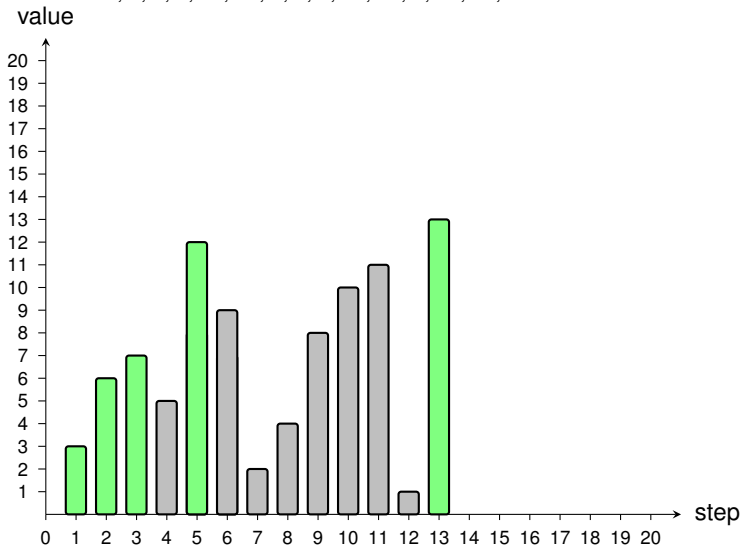


Illustration ($n = 20$)

unknown permutation:

4, 7, 8, 6, 18, 11, 3, 5, 9, 13, 17, 2, 20, 14,

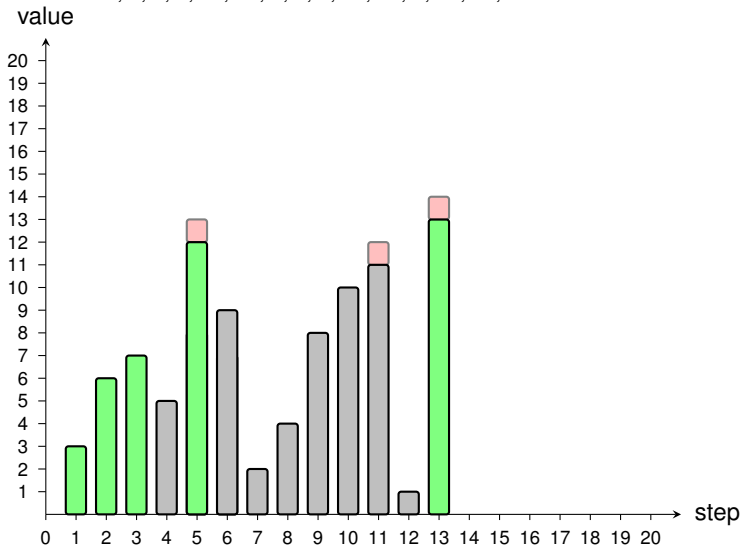


Illustration ($n = 20$)

unknown permutation:

4, 7, 8, 6, 18, 11, 3, 5, 9, 13, 17, 2, 20, 14,

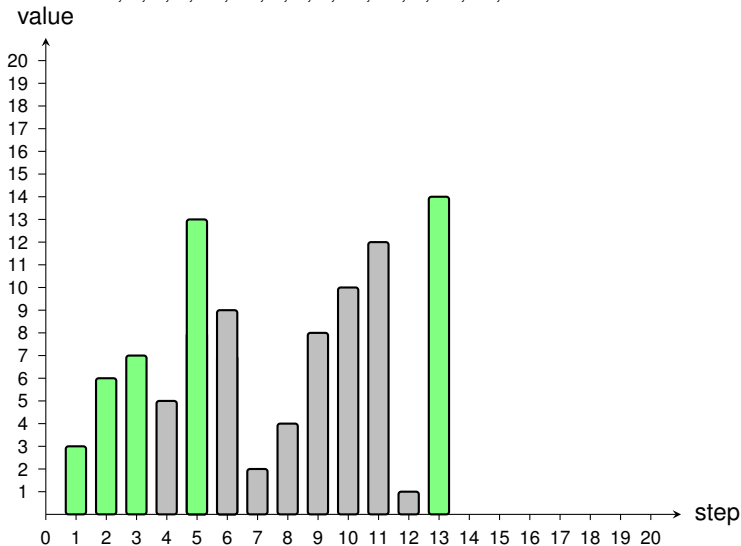


Illustration ($n = 20$)

unknown permutation:

4, 7, 8, 6, 18, 11, 3, 5, 9, 13, 17, 2, 20, 14,

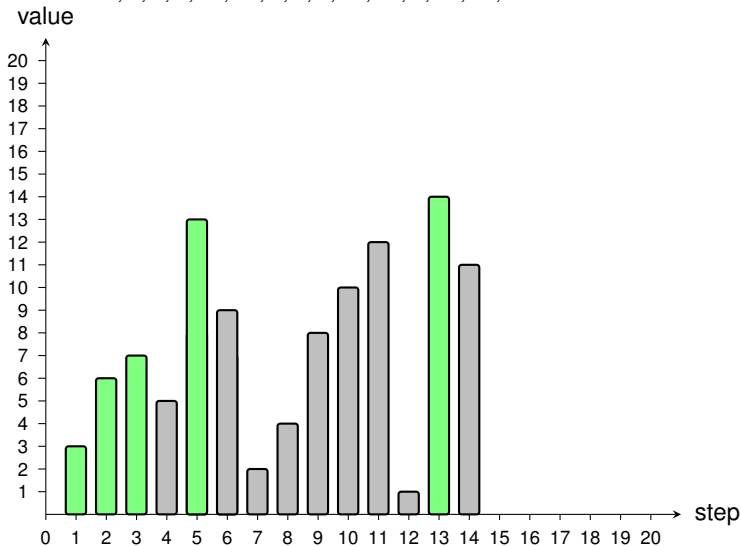


Illustration ($n = 20$)

unknown permutation:

4, 7, 8, 6, 18, 11, 3, 5, 9, 13, 17, 2, 20, 14, 12,

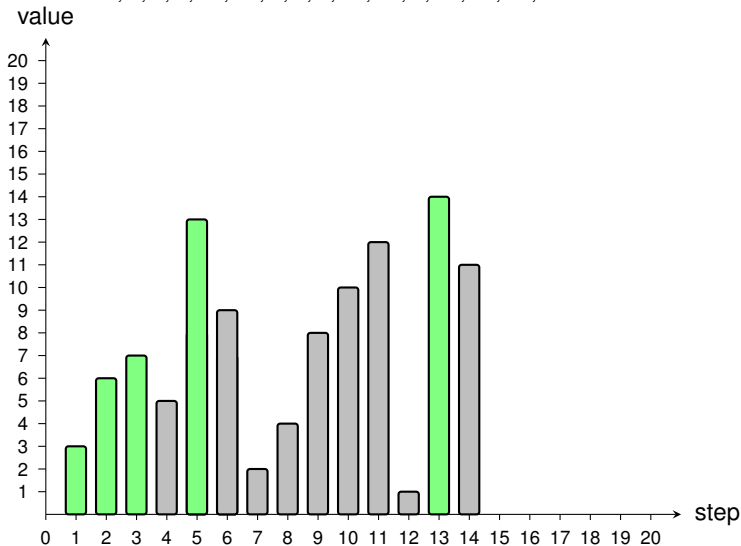


Illustration ($n = 20$)

unknown permutation:

4, 7, 8, 6, 18, 11, 3, 5, 9, 13, 17, 2, 20, 14, 12,

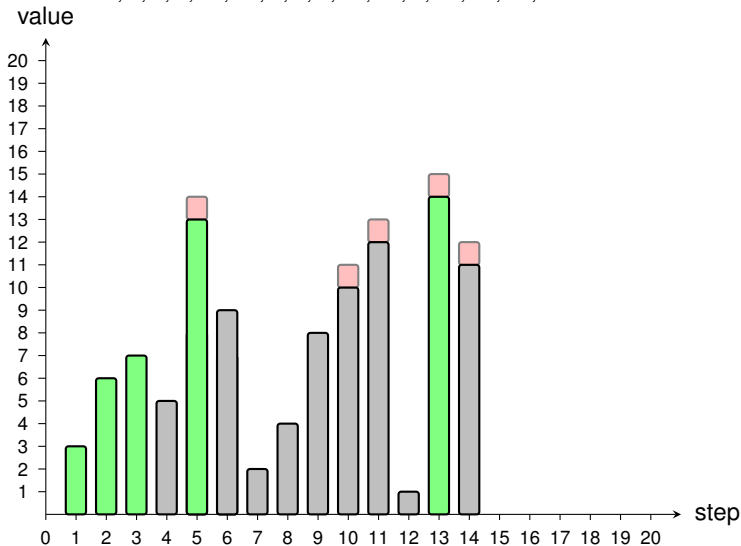


Illustration ($n = 20$)

unknown permutation:

4, 7, 8, 6, 18, 11, 3, 5, 9, 13, 17, 2, 20, 14, 12,

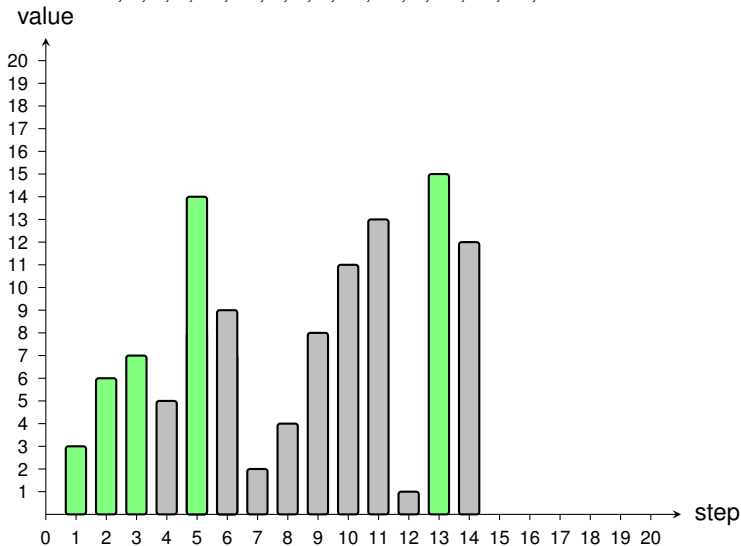


Illustration ($n = 20$)

unknown permutation:

4, 7, 8, 6, 18, 11, 3, 5, 9, 13, 17, 2, 20, 14, 12,

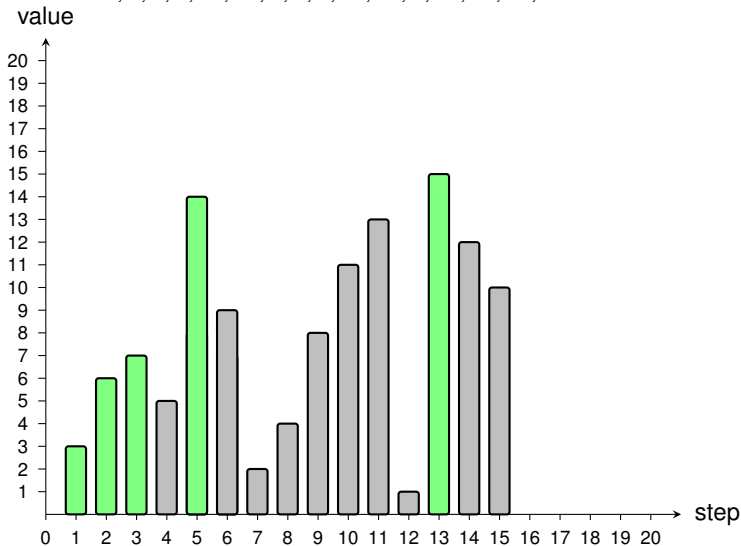


Illustration ($n = 20$)

unknown permutation:

4, 7, 8, 6, 18, 11, 3, 5, 9, 13, 17, 2, 20, 14, 12, 15,

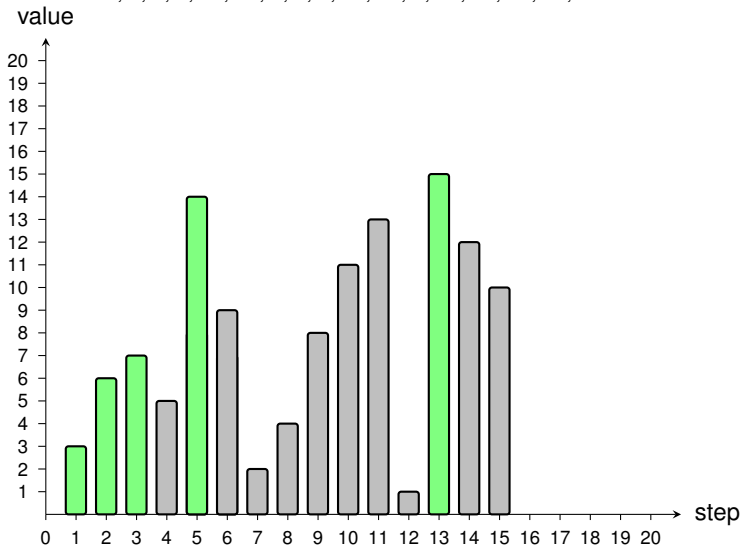


Illustration ($n = 20$)

unknown permutation:

4, 7, 8, 6, 18, 11, 3, 5, 9, 13, 17, 2, 20, 14, 12, 15,

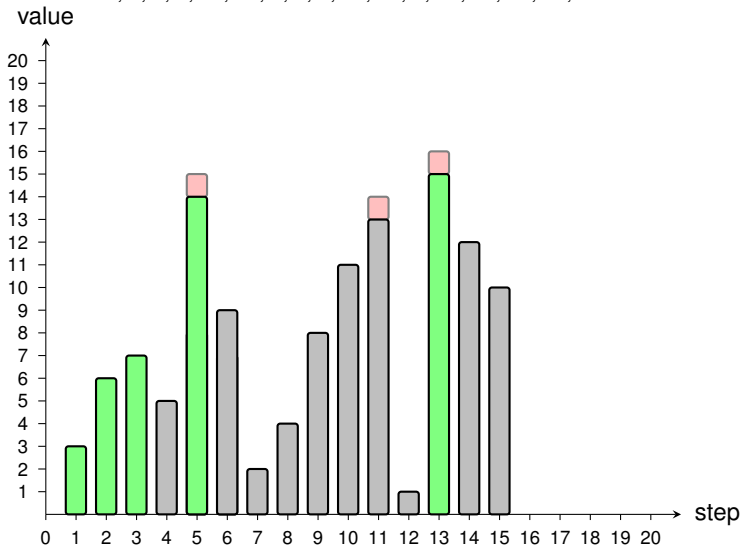


Illustration ($n = 20$)

unknown permutation:

4, 7, 8, 6, 18, 11, 3, 5, 9, 13, 17, 2, 20, 14, 12, 15,

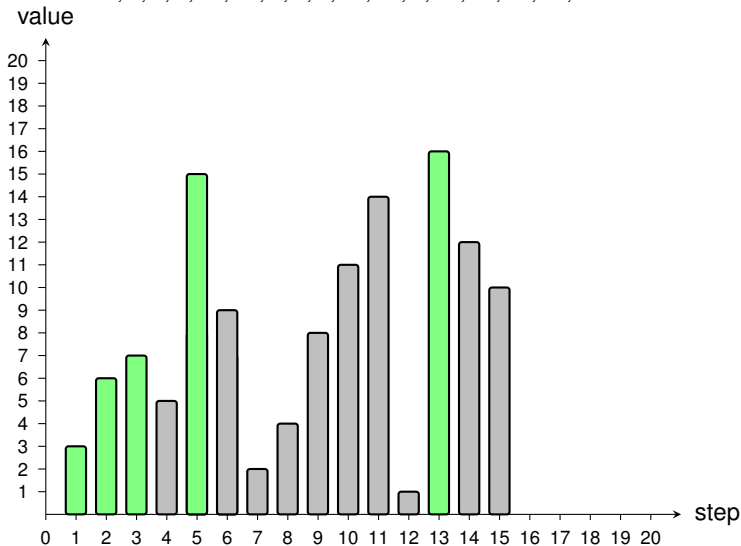


Illustration ($n = 20$)

unknown permutation:

4, 7, 8, 6, 18, 11, 3, 5, 9, 13, 17, 2, 20, 14, 12, 15,

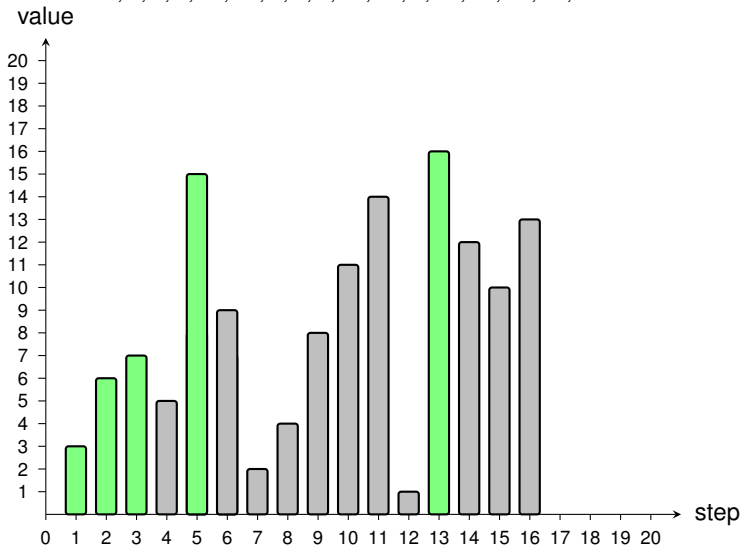


Illustration ($n = 20$)

unknown permutation:

4, 7, 8, 6, 18, 11, 3, 5, 9, 13, 17, 2, 20, 14, 12, 15, 10,

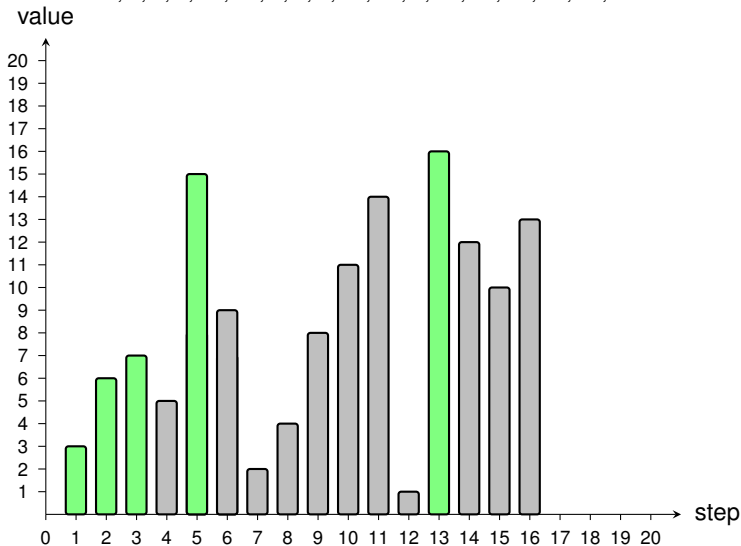


Illustration ($n = 20$)

unknown permutation:

4, 7, 8, 6, 18, 11, 3, 5, 9, 13, 17, 2, 20, 14, 12, 15, 10,

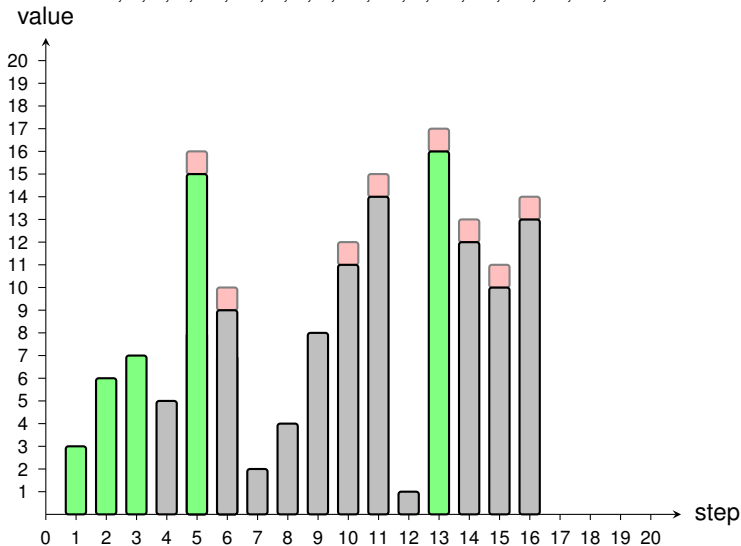


Illustration ($n = 20$)

unknown permutation:

4, 7, 8, 6, 18, 11, 3, 5, 9, 13, 17, 2, 20, 14, 12, 15, 10,

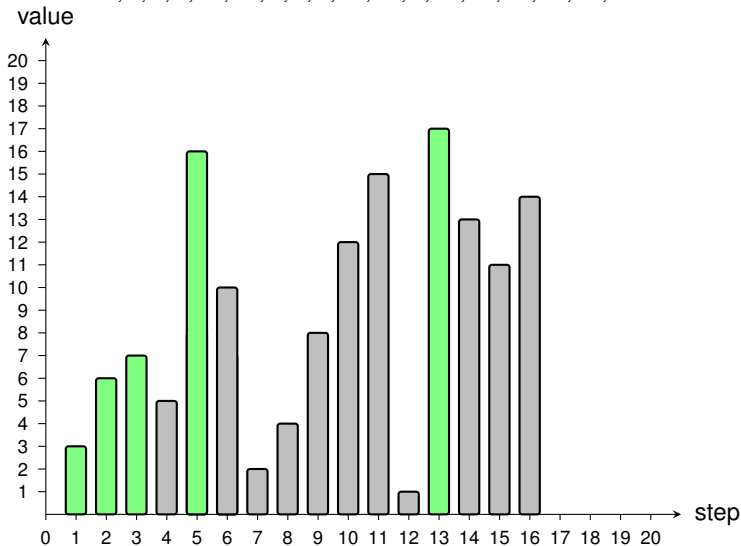


Illustration ($n = 20$)

unknown permutation:

4, 7, 8, 6, 18, 11, 3, 5, 9, 13, 17, 2, 20, 14, 12, 15, 10,

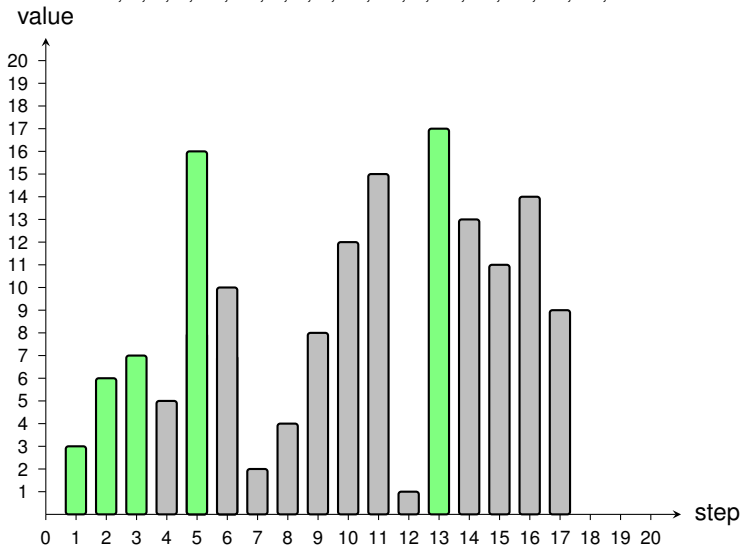


Illustration ($n = 20$)

unknown permutation:

4, 7, 8, 6, 18, 11, 3, 5, 9, 13, 17, 2, 20, 14, 12, 15, 10, 16,

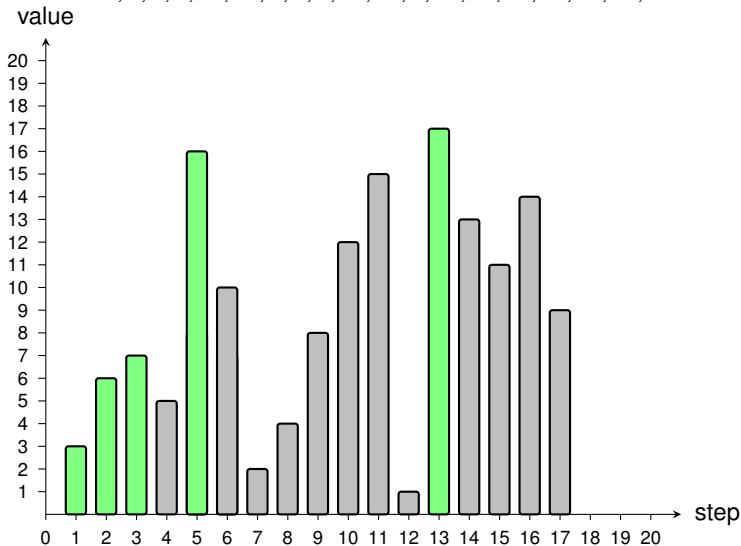


Illustration ($n = 20$)

unknown permutation:

4, 7, 8, 6, 18, 11, 3, 5, 9, 13, 17, 2, 20, 14, 12, 15, 10, 16,

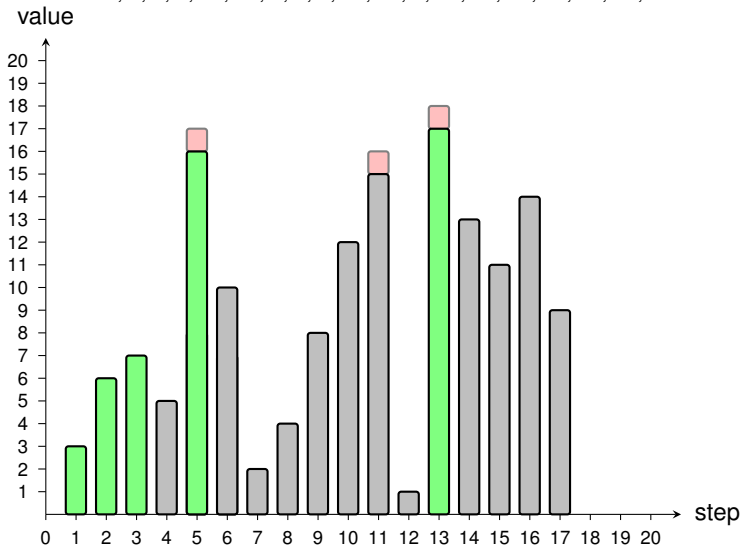


Illustration ($n = 20$)

unknown permutation:

4, 7, 8, 6, 18, 11, 3, 5, 9, 13, 17, 2, 20, 14, 12, 15, 10, 16,

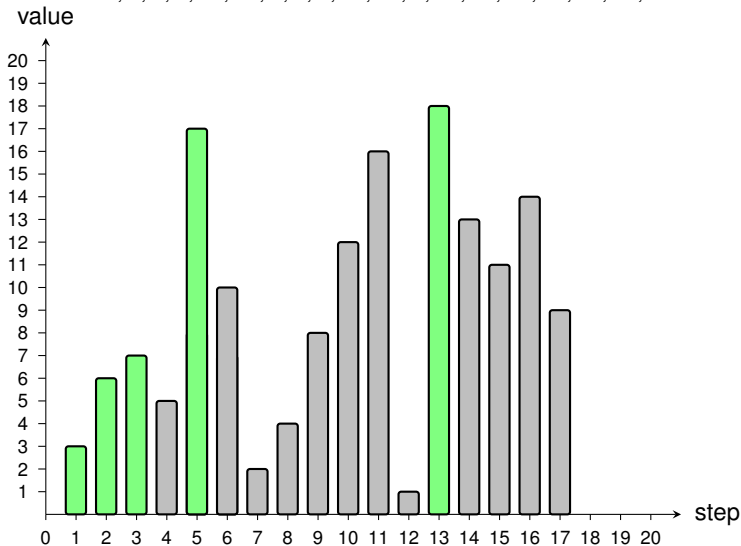


Illustration ($n = 20$)

unknown permutation:

4, 7, 8, 6, 18, 11, 3, 5, 9, 13, 17, 2, 20, 14, 12, 15, 10, 16,

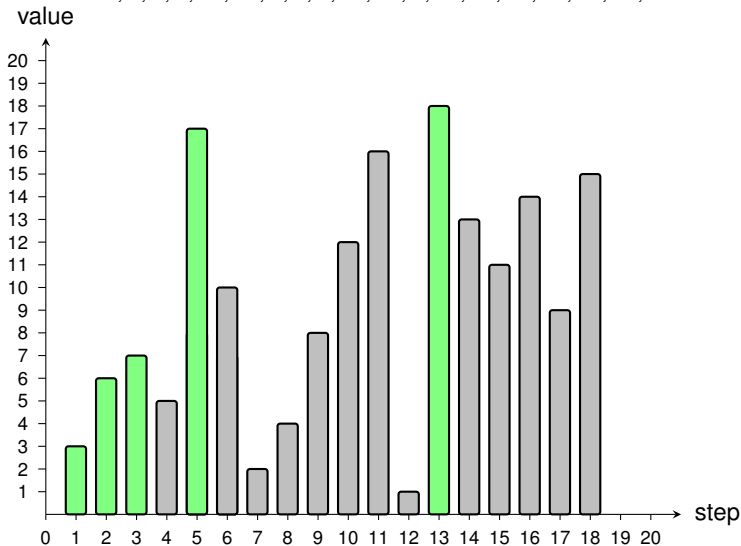


Illustration ($n = 20$)

unknown permutation:

4, 7, 8, 6, 18, 11, 3, 5, 9, 13, 17, 2, 20, 14, 12, 15, 10, 16, 19,

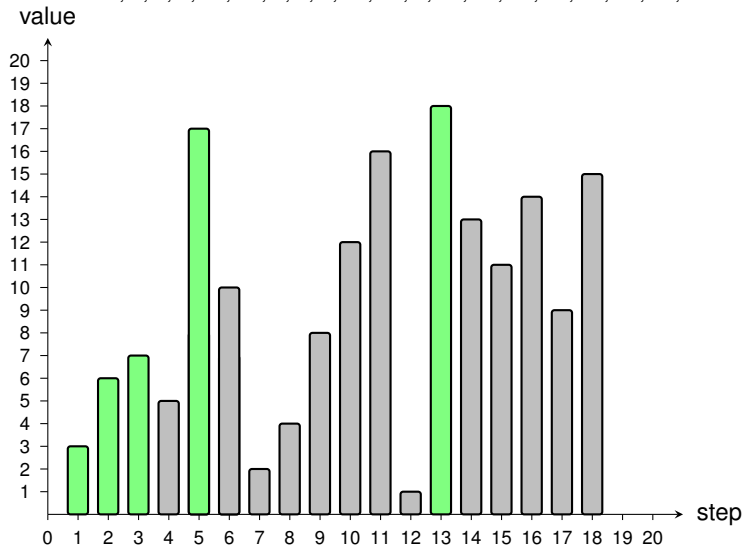


Illustration ($n = 20$)

unknown permutation:

4, 7, 8, 6, 18, 11, 3, 5, 9, 13, 17, 2, 20, 14, 12, 15, 10, 16, 19,

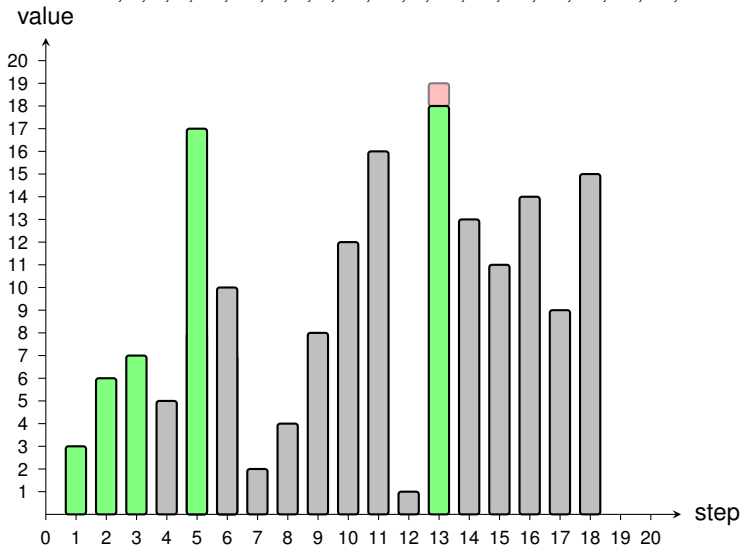


Illustration ($n = 20$)

unknown permutation:

4, 7, 8, 6, 18, 11, 3, 5, 9, 13, 17, 2, 20, 14, 12, 15, 10, 16, 19,

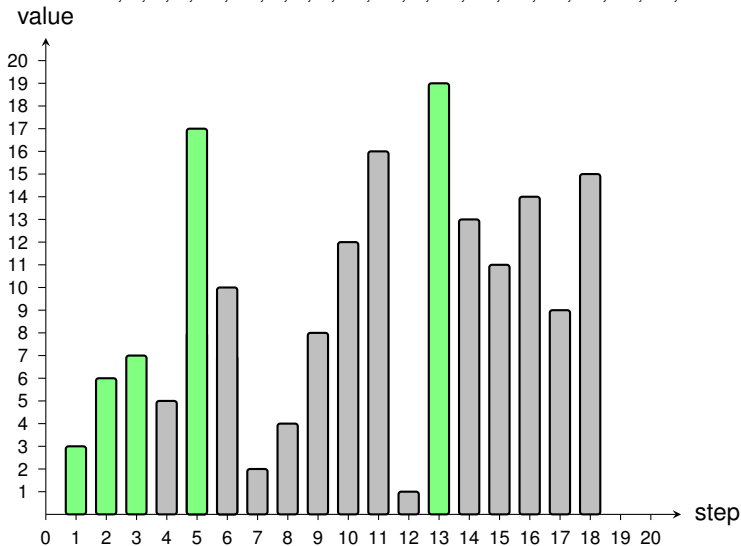


Illustration ($n = 20$)

unknown permutation:

4, 7, 8, 6, 18, 11, 3, 5, 9, 13, 17, 2, 20, 14, 12, 15, 10, 16, 19,

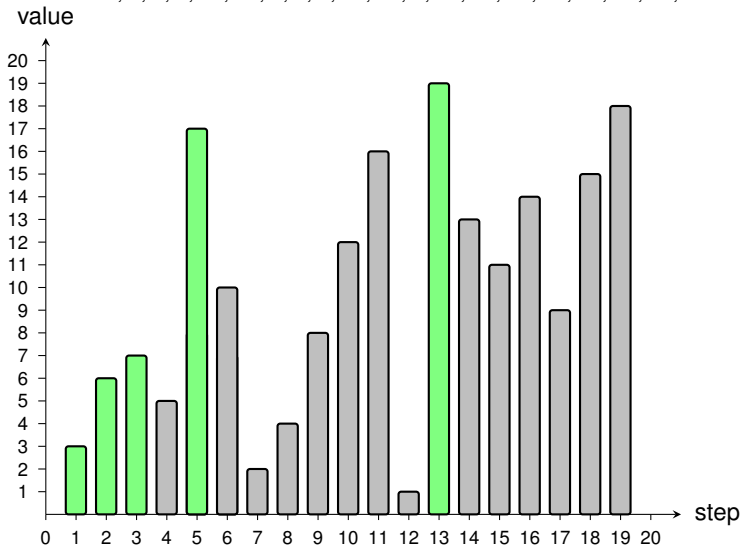


Illustration ($n = 20$)

unknown permutation:

4, 7, 8, 6, 18, 11, 3, 5, 9, 13, 17, 2, 20, 14, 12, 15, 10, 16, 19, 1.

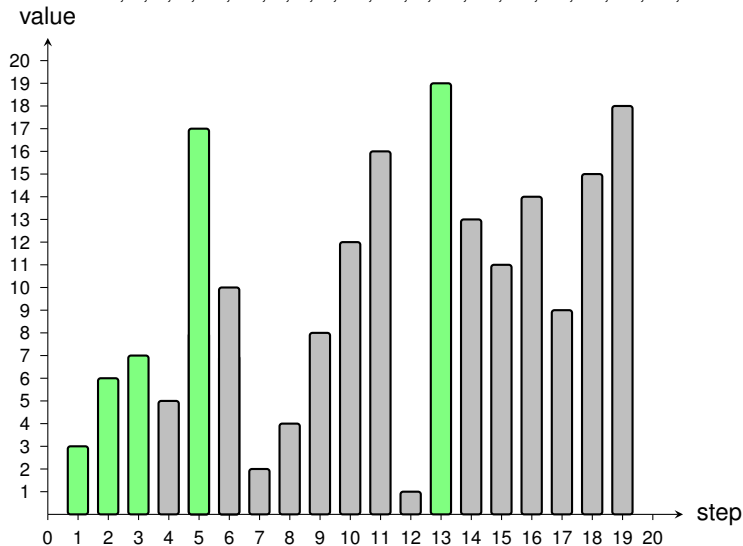


Illustration ($n = 20$)

unknown permutation:

4, 7, 8, 6, 18, 11, 3, 5, 9, 13, 17, 2, 20, 14, 12, 15, 10, 16, 19, 1.

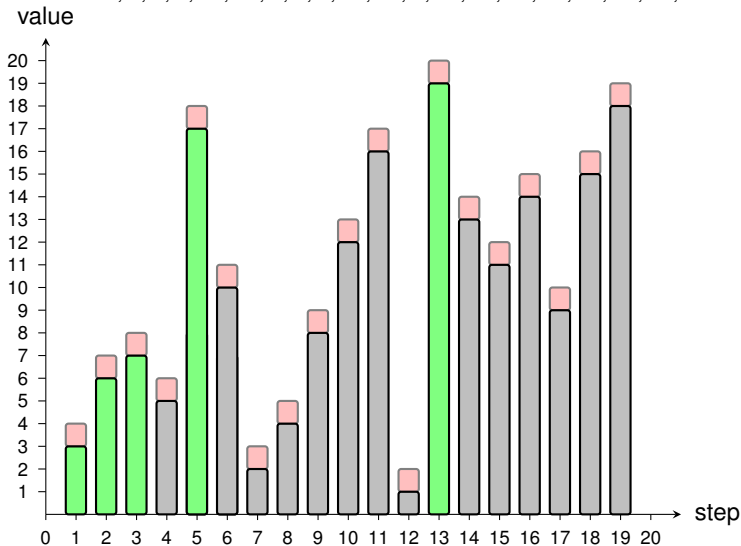


Illustration ($n = 20$)

unknown permutation:

4, 7, 8, 6, 18, 11, 3, 5, 9, 13, 17, 2, 20, 14, 12, 15, 10, 16, 19, 1.

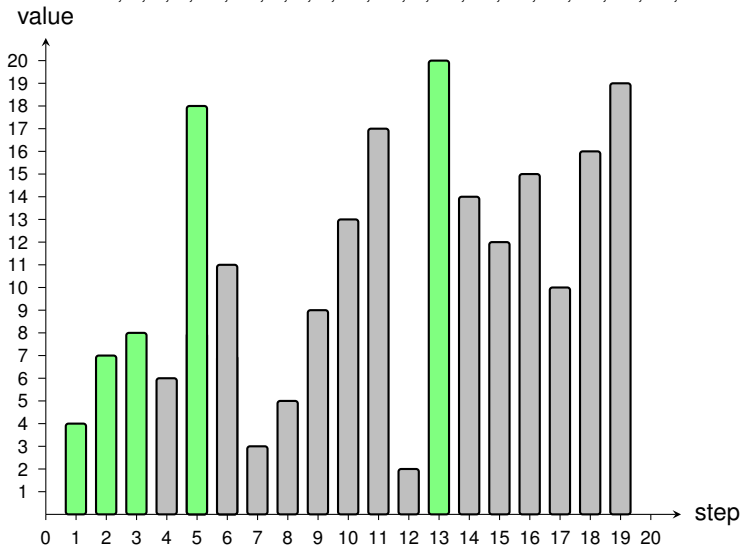


Illustration ($n = 20$)

unknown permutation:

4, 7, 8, 6, 18, 11, 3, 5, 9, 13, 17, 2, 20, 14, 12, 15, 10, 16, 19, 1.

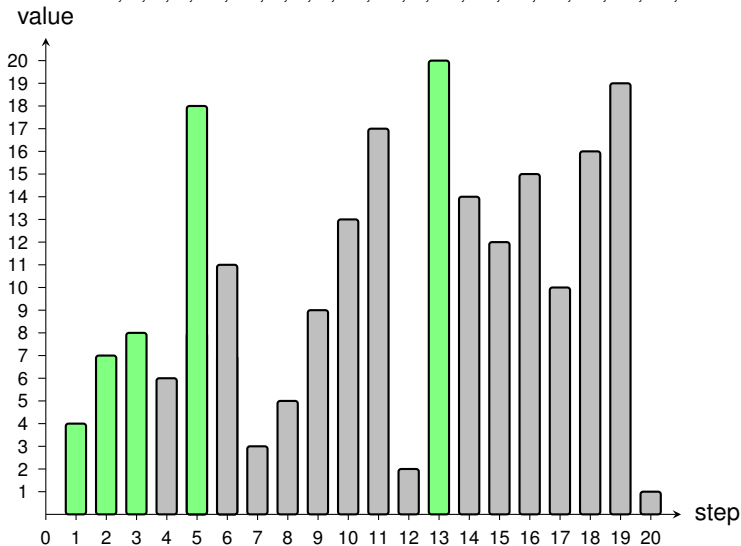
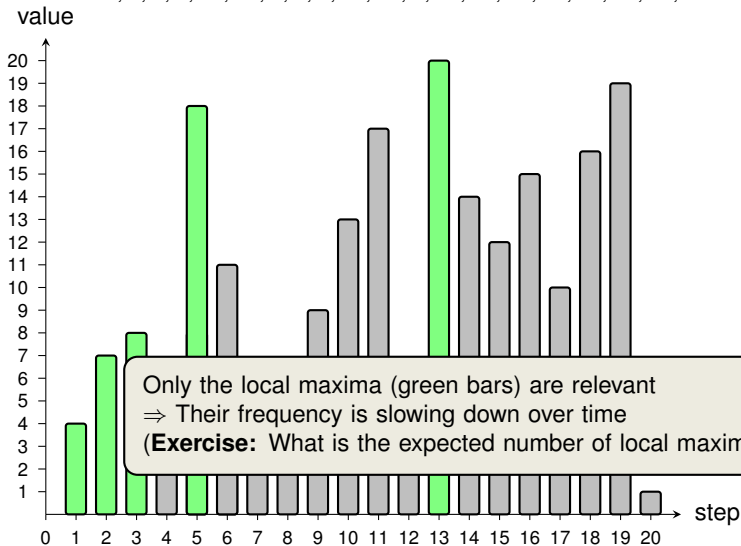


Illustration ($n = 20$)

unknown permutation:

4, 7, 8, 6, 18, 11, 3, 5, 9, 13, 17, 2, 20, 14, 12, 15, 10, 16, 19, 1.



Two Basic Strategies

Naive Approach



Two Basic Strategies

Naive Approach

- Always pick the **first** (or any other) candidate

Two Basic Strategies

Naive Approach

- Always pick the **first** (or any other) candidate
- Probability for success is:

$$\mathbf{P} [\text{hire best candidate}]$$

Two Basic Strategies

Naive Approach

- Always pick the **first** (or any other) candidate
- Probability for success is:

$$\mathbf{P}[\text{hire best candidate}] = \frac{1}{n}.$$

Two Basic Strategies

Naive Approach

- Always pick the **first** (or any other) candidate
- Probability for success is:

$$\mathbf{P}[\text{hire best candidate}] = \frac{1}{n}.$$

Smarter Approach

Two Basic Strategies

Naive Approach

- Always pick the **first** (or any other) candidate
- Probability for success is:

$$\mathbf{P}[\text{hire best candidate}] = \frac{1}{n}.$$

Smarter Approach

- Reject the first $n/2$ candidates, then take the first candidate that is better than the first $n/2$

Two Basic Strategies

Naive Approach

- Always pick the **first** (or any other) candidate
- Probability for success is:

$$\mathbf{P}[\text{hire best candidate}] = \frac{1}{n}.$$

Smarter Approach

- Reject the first $n/2$ candidates, then take the first candidate that is better than the first $n/2$ (if none is taken before, take last candidate)

Two Basic Strategies

Naive Approach

- Always pick the **first** (or any other) candidate
- Probability for success is:

$$\mathbf{P}[\text{hire best candidate}] = \frac{1}{n}.$$

A typical **exploration-exploitation** based strategy.

Smarter Approach

- Reject the first $n/2$ candidates, then take the first candidate that is better than the first $n/2$ (if none is taken before, take last candidate)

Two Basic Strategies

Naive Approach

- Always pick the **first** (or any other) candidate
- Probability for success is:

$$\mathbf{P}[\text{hire best candidate}] = \frac{1}{n}.$$

A typical **exploration-exploitation** based strategy.

Smarter Approach

- Reject the first $n/2$ candidates, then take the first candidate that is better than the first $n/2$ (if none is taken before, take last candidate)

How good is this approach?

Analysis of the Refined Approach

Example 1

Find a lower bound on the success probability of the refined approach (picking the first candidate better than the first $n/2$).

_____ Answer _____

Example 1

Find a lower bound on the success probability of the refined approach (picking the first candidate better than the first $n/2$).

Answer

- Probability for success is:

\mathbf{P} [hire best candidate]

\geq

Example 1

Find a lower bound on the success probability of the refined approach (picking the first candidate better than the first $n/2$).

Answer

- Probability for success is:

\mathbf{P} [hire best candidate]

$\geq \mathbf{P}$ [best in 2nd half \cap second best in 1st half]

Example 1

Find a lower bound on the success probability of the refined approach (picking the first candidate better than the first $n/2$).

Answer

- Probability for success is:

$$\mathbf{P}[\text{hire best candidate}]$$

$$\geq \mathbf{P}[\text{best in 2nd half} \cap \text{second best in 1st half}]$$

$$= \mathbf{P}[\text{best in 2nd half}] \cdot \mathbf{P}[\text{second best in 1st half} \mid \text{best in 2nd half}]$$

Example 1

Find a lower bound on the success probability of the refined approach (picking the first candidate better than the first $n/2$).

Answer

- Probability for success is:

$$\begin{aligned} & \mathbf{P}[\text{hire best candidate}] \\ & \geq \mathbf{P}[\text{best in 2nd half} \cap \text{second best in 1st half}] \\ & = \mathbf{P}[\text{best in 2nd half}] \cdot \mathbf{P}[\text{second best in 1st half} \mid \text{best in 2nd half}] \\ & = \frac{n/2}{n} \cdot \frac{n/2}{n-1} \end{aligned}$$

Example 1

Find a lower bound on the success probability of the refined approach (picking the first candidate better than the first $n/2$).

Answer

- Probability for success is:

$$\mathbf{P}[\text{hire best candidate}]$$

$$\geq \mathbf{P}[\text{best in 2nd half} \cap \text{second best in 1st half}]$$

$$= \mathbf{P}[\text{best in 2nd half}] \cdot \mathbf{P}[\text{second best in 1st half} \mid \text{best in 2nd half}]$$

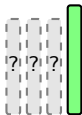
$$= \frac{n/2}{n} \cdot \frac{n/2}{n-1} > \frac{1}{4}.$$

Finding the Optimal Strategy (1/2)

- **Observation 1:** At interview i , it only matters if current candidate is best so far (i.e., no benefit in counting how many “best-so-far” candidates we had).

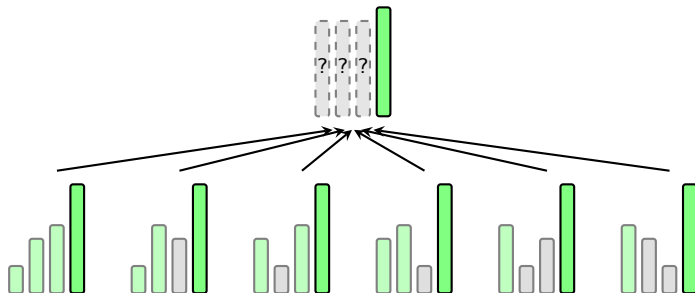
Finding the Optimal Strategy (1/2)

- **Observation 1:** At interview i , it only matters if current candidate is best so far (i.e., no benefit in counting how many “best-so-far” candidates we had).



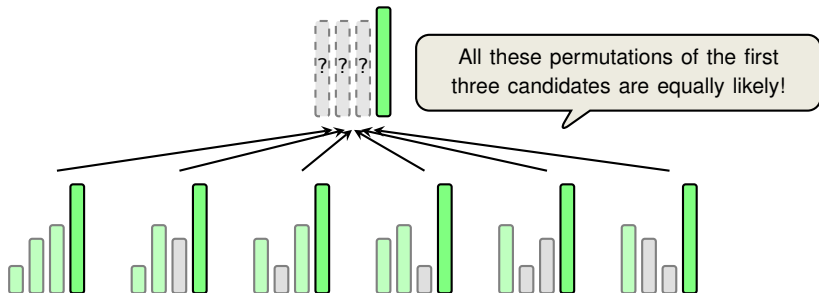
Finding the Optimal Strategy (1/2)

- **Observation 1:** At interview i , it only matters if current candidate is best so far (i.e., no benefit in counting how many “best-so-far” candidates we had).



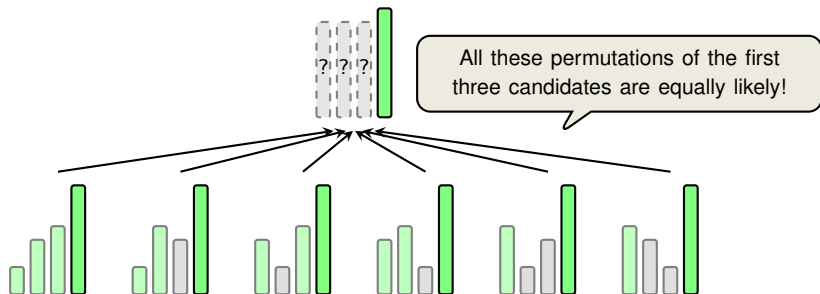
Finding the Optimal Strategy (1/2)

- **Observation 1:** At interview i , it only matters if current candidate is best so far (i.e., no benefit in counting how many “best-so-far” candidates we had).



Finding the Optimal Strategy (1/2)

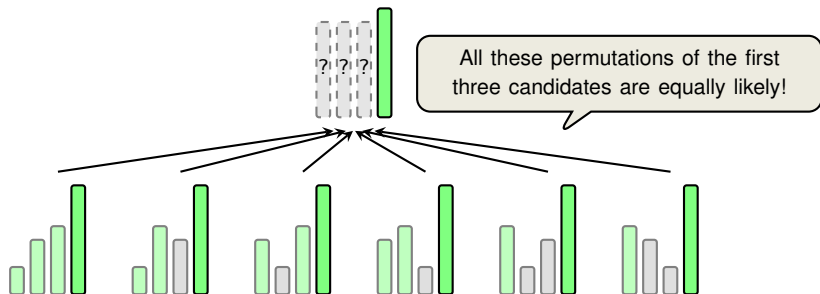
- **Observation 1:** At interview i , it only matters if current candidate is best so far (i.e., no benefit in counting how many “best-so-far” candidates we had).



- **Observation 2:** If at interview i , the best strategy is to accept the candidate (if it is “best-so-far”), then the same holds for interview $i + 1$

Finding the Optimal Strategy (1/2)

- **Observation 1:** At interview i , it only matters if current candidate is best so far (i.e., no benefit in counting how many “best-so-far” candidates we had).



- **Observation 2:** If at interview i , the best strategy is to accept the candidate (if it is “best-so-far”), then the same holds for interview $i + 1$

Optimal Strategy

- **Explore** but reject the first $x - 1$ candidates
- **Accept** first candidate $i \geq x$ which is better than **all candidates before**

Example 2

Find x which maximises the probability of hiring the best candidate.

Answer

- First compute **success probability** for any $x \in \{1, \dots, n\}$, and then **optimise**:

Example 2

Find x which maximises the probability of hiring the best candidate.

Answer

- First compute **success probability** for any $x \in \{1, \dots, n\}$, and then **optimise**:

P [hire best candidate]

Example 2

Find x which maximises the probability of hiring the best candidate.

Answer

- First compute **success probability** for any $x \in \{1, \dots, n\}$, and then **optimise**:

\mathbf{P} [hire best candidate]

$$= \sum_{i=1}^n \mathbf{P} [\text{hire candidate } i \cap \text{candidate } i \text{ is best}]$$

Example 2

Find x which maximises the probability of hiring the best candidate.

Answer

- First compute **success probability** for any $x \in \{1, \dots, n\}$, and then **optimise**:

$\mathbf{P}[\text{hire best candidate}]$

$$= \sum_{i=1}^n \mathbf{P}[\text{hire candidate } i \cap \text{candidate } i \text{ is best}]$$

$$= \sum_{i=x}^n \mathbf{P}[\text{hire candidate } i \cap \text{candidate } i \text{ is best}]$$

Example 2

Find x which maximises the probability of hiring the best candidate.

Answer

- First compute **success probability** for any $x \in \{1, \dots, n\}$, and then **optimise**:

$\mathbf{P}[\text{hire best candidate}]$

$$= \sum_{i=1}^n \mathbf{P}[\text{hire candidate } i \cap \text{candidate } i \text{ is best}]$$

$$= \sum_{i=x}^n \mathbf{P}[\text{hire candidate } i \cap \text{candidate } i \text{ is best}]$$

$$= \sum_{i=x}^n \mathbf{P}[\text{hire candidate } i \mid \text{candidate } i \text{ is best}] \cdot \mathbf{P}[\text{candidate } i \text{ is best}]$$

Example 2

Find x which maximises the probability of hiring the best candidate.

Answer

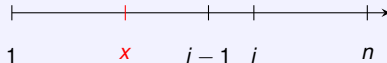
- First compute **success probability** for any $x \in \{1, \dots, n\}$, and then **optimise**:

$\mathbf{P}[\text{hire best candidate}]$

$$= \sum_{i=1}^n \mathbf{P}[\text{hire candidate } i \cap \text{candidate } i \text{ is best}]$$

$$= \sum_{i=x}^n \mathbf{P}[\text{hire candidate } i \cap \text{candidate } i \text{ is best}]$$

$$= \sum_{i=x}^n \mathbf{P}[\text{hire candidate } i \mid \text{candidate } i \text{ is best}] \cdot \mathbf{P}[\text{candidate } i \text{ is best}]$$



Example 2

Find x which maximises the probability of hiring the best candidate.

Answer

- First compute **success probability** for any $x \in \{1, \dots, n\}$, and then **optimise**:

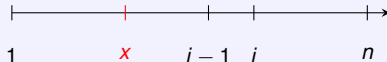
$\mathbf{P}[\text{hire best candidate}]$

$$= \sum_{i=1}^n \mathbf{P}[\text{hire candidate } i \cap \text{candidate } i \text{ is best}]$$

$$= \sum_{i=x}^n \mathbf{P}[\text{hire candidate } i \cap \text{candidate } i \text{ is best}]$$

$$= \sum_{i=x}^n \mathbf{P}[\text{hire candidate } i \mid \text{candidate } i \text{ is best}] \cdot \mathbf{P}[\text{candidate } i \text{ is best}]$$

$$= \frac{1}{n} \cdot \sum_{i=x}^n \mathbf{P}[\text{second best of first } i \text{ candidates is in the first } x-1 \mid \text{candidate } i \text{ is best}]$$



Example 2

Find x which maximises the probability of hiring the best candidate.

Answer

- First compute **success probability** for any $x \in \{1, \dots, n\}$, and then **optimise**:

\mathbf{P} [hire best candidate]

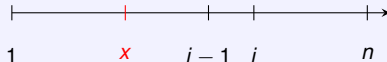
$$= \sum_{i=1}^n \mathbf{P} [\text{hire candidate } i \cap \text{candidate } i \text{ is best}]$$

$$= \sum_{i=x}^n \mathbf{P} [\text{hire candidate } i \cap \text{candidate } i \text{ is best}]$$

$$= \sum_{i=x}^n \mathbf{P} [\text{hire candidate } i \mid \text{candidate } i \text{ is best}] \cdot \mathbf{P} [\text{candidate } i \text{ is best}]$$

$$= \frac{1}{n} \cdot \sum_{i=x}^n \mathbf{P} [\text{second best of first } i \text{ candidates is in the first } x-1 \mid \text{candidate } i \text{ is best}]$$

$$= \frac{1}{n} \cdot \sum_{i=x}^n \frac{x-1}{i-1}$$



Example 2

Find x which maximises the probability of hiring the best candidate.

Answer

- First compute **success probability** for any $x \in \{1, \dots, n\}$, and then **optimise**:

\mathbf{P} [hire best candidate]

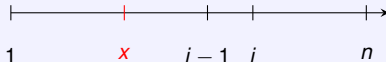
$$= \sum_{i=1}^n \mathbf{P} [\text{hire candidate } i \cap \text{candidate } i \text{ is best}]$$

$$= \sum_{i=x}^n \mathbf{P} [\text{hire candidate } i \cap \text{candidate } i \text{ is best}]$$

$$= \sum_{i=x}^n \mathbf{P} [\text{hire candidate } i \mid \text{candidate } i \text{ is best}] \cdot \mathbf{P} [\text{candidate } i \text{ is best}]$$

$$= \frac{1}{n} \cdot \sum_{i=x}^n \mathbf{P} [\text{second best of first } i \text{ candidates is in the first } x-1 \mid \text{candidate } i \text{ is best}]$$

$$= \frac{1}{n} \cdot \sum_{i=x}^n \frac{x-1}{i-1} = \frac{x-1}{n} \cdot \sum_{i=x}^n \frac{1}{i-1}$$



Example 2

Find x which maximises the probability of hiring the best candidate.

Answer

- First compute **success probability** for any $x \in \{1, \dots, n\}$, and then **optimise**:

\mathbf{P} [hire best candidate]

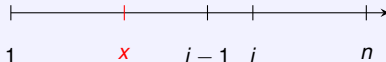
$$= \sum_{i=1}^n \mathbf{P} [\text{hire candidate } i \cap \text{candidate } i \text{ is best}]$$

$$= \sum_{i=x}^n \mathbf{P} [\text{hire candidate } i \cap \text{candidate } i \text{ is best}]$$

$$= \sum_{i=x}^n \mathbf{P} [\text{hire candidate } i \mid \text{candidate } i \text{ is best}] \cdot \mathbf{P} [\text{candidate } i \text{ is best}]$$

$$= \frac{1}{n} \cdot \sum_{i=x}^n \mathbf{P} [\text{second best of first } i \text{ candidates is in the first } x-1 \mid \text{candidate } i \text{ is best}]$$

$$= \frac{1}{n} \cdot \sum_{i=x}^n \frac{x-1}{i-1} = \frac{x-1}{n} \cdot \sum_{i=x}^n \frac{1}{i-1}$$



$$\Rightarrow \sum_{i=x}^n \frac{1}{i-1} \approx \ln(n/x)$$

Example 2

Find x which maximises the probability of hiring the best candidate.

Answer

- First compute **success probability** for any $x \in \{1, \dots, n\}$, and then **optimise**:

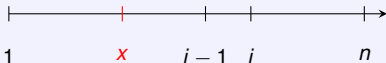
\mathbf{P} [hire best candidate]

$$= \sum_{i=1}^n \mathbf{P} [\text{hire candidate } i \cap \text{candidate } i \text{ is best}]$$

$$= \sum_{i=x}^n \mathbf{P} [\text{hire candidate } i \cap \text{candidate } i \text{ is best}]$$

$$= \sum_{i=x}^n \mathbf{P} [\text{hire candidate } i \mid \text{candidate } i \text{ is best}] \cdot \mathbf{P} [\text{candidate } i \text{ is best}]$$

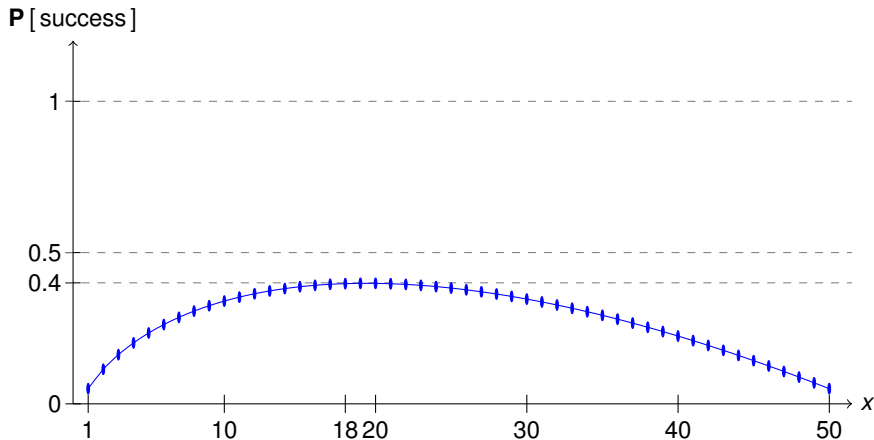
$$= \frac{1}{n} \cdot \sum_{i=x}^n \mathbf{P} [\text{second best of first } i \text{ candidates is in the first } x-1 \mid \text{candidate } i \text{ is best}]$$

$$= \frac{1}{n} \cdot \sum_{i=x}^n \frac{x-1}{i-1} = \frac{x-1}{n} \cdot \sum_{i=x}^n \frac{1}{i-1}$$


$$\Rightarrow \sum_{i=x}^n \frac{1}{i-1} \approx \ln(n/x) \Rightarrow \text{maximum success probability for } x = \frac{1}{e} \cdot n.$$

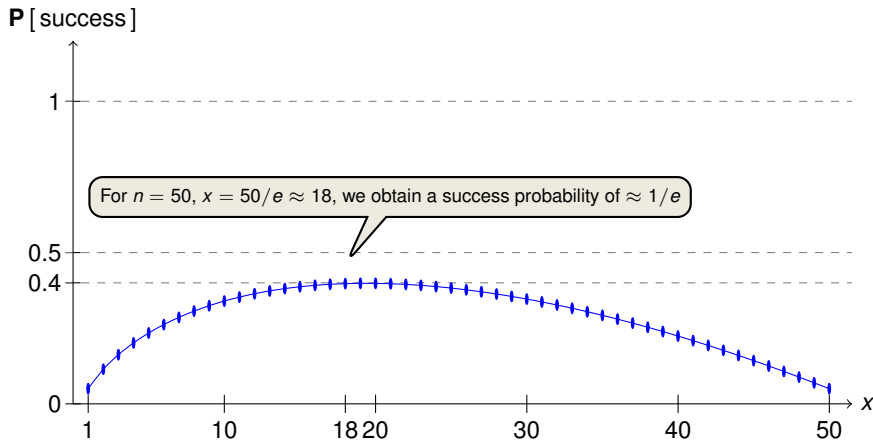
Probability for Success (Illustration)

Suppose $n = 50$:



Probability for Success (Illustration)

Suppose $n = 50$:



Another Variant of the Secretary Problem

— “The Postdoc Variant of the Secretary Problem” (Vanderbei’80) —

- same setup as in the secretary problem before
- **difference**: we want to pick the **second-best** (“the best [postdoc] is going to Harvard”)
- Success probability of the optimal strategy is:

$$\frac{0.25n^2}{n(n-1)} \xrightarrow{n \rightarrow \infty} \frac{1}{4}$$

- Thus it is **easier** to pick the best than the second-best(!)

Outline

Stopping Problem 1: Dice Game

Stopping Problem 2: The Secretary Problem

A Generalisation: The Odds Algorithm (non-examinable)

The End...

Details of the Odds Algorithm

- Let I_1, I_2, \dots, I_n be a sequence of **independent** indicators and let $p_j = \mathbf{E}[I_j]$

Details of the Odds Algorithm

- Let I_1, I_2, \dots, I_n be a sequence of **independent** indicators and let $p_j = \mathbf{E}[I_j]$
- Let $r_j := \frac{p_j}{1-p_j}$ (**the odds**) and $p_j \in (0, 1)$ for all $j = 1, 2, \dots, n$

Details of the Odds Algorithm

- Let I_1, I_2, \dots, I_n be a sequence of **independent** indicators and let $p_j = \mathbf{E}[I_j]$
- Let $r_j := \frac{p_j}{1-p_j}$ (**the odds**) and $p_j \in (0, 1)$ for all $j = 1, 2, \dots, n$

Example 3

What is the probability that after trial k , there is exactly one success?

Answer

Details of the Odds Algorithm

- Let I_1, I_2, \dots, I_n be a sequence of **independent** indicators and let $p_j = \mathbf{E}[I_j]$
- Let $r_j := \frac{p_j}{1-p_j}$ (**the odds**) and $p_j \in (0, 1)$ for all $j = 1, 2, \dots, n$

Example 3

What is the probability that after trial k , there is exactly one success?

Answer

$$\mathbf{P} \left[\sum_{j=k}^n I_j = 1 \right]$$

Details of the Odds Algorithm

- Let l_1, l_2, \dots, l_n be a sequence of **independent** indicators and let $p_j = \mathbf{E}[l_j]$
- Let $r_j := \frac{p_j}{1-p_j}$ (**the odds**) and $p_j \in (0, 1)$ for all $j = 1, 2, \dots, n$

Example 3

What is the probability that after trial k , there is exactly one success?

Answer

$$\mathbf{P} \left[\sum_{j=k}^n l_j = 1 \right] = \sum_{j=k}^n p_j \cdot \prod_{k \leq j \leq n, j \neq i} (1 - p_i)$$

Details of the Odds Algorithm

- Let I_1, I_2, \dots, I_n be a sequence of **independent** indicators and let $p_j = \mathbf{E}[I_j]$
- Let $r_j := \frac{p_j}{1-p_j}$ (**the odds**) and $p_j \in (0, 1)$ for all $j = 1, 2, \dots, n$

Example 3

What is the probability that after trial k , there is exactly one success?

Answer

$$\mathbf{P} \left[\sum_{j=k}^n I_j = 1 \right] = \sum_{j=k}^n p_j \cdot \prod_{k \leq j \leq n, j \neq i} (1 - p_i) = \sum_{j=k}^n r_j \cdot \left(\prod_{i=k}^n (1 - p_i) \right)$$

Details of the Odds Algorithm

- Let l_1, l_2, \dots, l_n be a sequence of **independent** indicators and let $p_j = \mathbf{E}[l_j]$
- Let $r_j := \frac{p_j}{1-p_j}$ (**the odds**) and $p_j \in (0, 1)$ for all $j = 1, 2, \dots, n$

Example 3

What is the probability that after trial k , there is exactly one success?

Answer

$$\mathbf{P} \left[\sum_{j=k}^n l_j = 1 \right] = \sum_{j=k}^n p_j \cdot \prod_{k \leq j \leq n, j \neq i} (1 - p_i) = \sum_{j=k}^n r_j \cdot \left(\prod_{i=k}^n (1 - p_i) \right)$$

- One can prove that $\mathbf{P} \left[\sum_{j=k}^n l_j = 1 \right]$ is **unimodal** in $k \Rightarrow$ there is an **ideal point** from which on we should **STOP at the first success!**

Details of the Odds Algorithm

- Let I_1, I_2, \dots, I_n be a sequence of **independent** indicators and let $p_j = \mathbf{E}[I_j]$
- Let $r_j := \frac{p_j}{1-p_j}$ (**the odds**) and $p_j \in (0, 1)$ for all $j = 1, 2, \dots, n$

Example 3

What is the probability that after trial k , there is exactly one success?

Answer

$$\mathbf{P} \left[\sum_{j=k}^n I_j = 1 \right] = \sum_{j=k}^n p_j \cdot \prod_{k \leq j \leq n, j \neq i} (1 - p_i) = \sum_{j=k}^n r_j \cdot \left(\prod_{i=k}^n (1 - p_i) \right)$$

- One can prove that $\mathbf{P} \left[\sum_{j=k}^n I_j = 1 \right]$ is **unimodal** in $k \Rightarrow$ there is an **ideal point** from which on we should **STOP at the first success!**

Odds Algorithm ("Sum the Odds to One and Stop", F. Thomas Bruss, 2000)

- Let k^* be the largest k such that $\sum_{j=k}^n r_j \geq 1$
- Ignore** everything before the k^* -th trial, then **STOP** at the **first** success.

Details of the Odds Algorithm

- Let I_1, I_2, \dots, I_n be a sequence of **independent** indicators and let $p_j = \mathbf{E}[I_j]$
- Let $r_j := \frac{p_j}{1-p_j}$ (**the odds**) and $p_j \in (0, 1)$ for all $j = 1, 2, \dots, n$

Example 3

What is the probability that after trial k , there is exactly one success?

Answer

$$\mathbf{P} \left[\sum_{j=k}^n I_j = 1 \right] = \sum_{j=k}^n p_j \cdot \prod_{k \leq j \leq n, j \neq i} (1 - p_i) = \sum_{j=k}^n r_j \cdot \left(\prod_{i=k}^n (1 - p_i) \right)$$

- One can prove that $\mathbf{P} \left[\sum_{j=k}^n I_j = 1 \right]$ is **unimodal** in $k \Rightarrow$ there is an **ideal point** from which on we should **STOP at the first success!**

Odds Algorithm ("Sum the Odds to One and Stop", F. Thomas Bruss, 2000)

- Let k^* be the largest k such that $\sum_{j=k}^n r_j \geq 1$
- Ignore** everything before the k^* -th trial, then **STOP** at the **first** success.

- The **success probability** is $\sum_{j=k^*}^n r_j \cdot \left(\prod_{i=k^*}^n (1 - p_i) \right)$.

Details of the Odds Algorithm

- Let I_1, I_2, \dots, I_n be a sequence of **independent** indicators and let $p_j = \mathbf{E}[I_j]$
- Let $r_j := \frac{p_j}{1-p_j}$ (**the odds**) and $p_j \in (0, 1)$ for all $j = 1, 2, \dots, n$

Example 3

What is the probability that after trial k , there is exactly one success?

Answer

$$\mathbf{P} \left[\sum_{j=k}^n I_j = 1 \right] = \sum_{j=k}^n p_j \cdot \prod_{k \leq j \leq n, j \neq i} (1 - p_i) = \sum_{j=k}^n r_j \cdot \left(\prod_{i=k}^n (1 - p_i) \right)$$

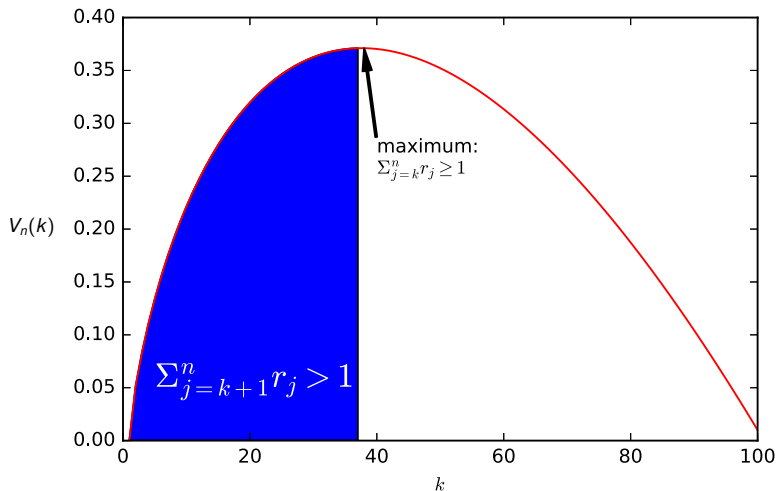
- One can prove that $\mathbf{P} \left[\sum_{j=k}^n I_j = 1 \right]$ is **unimodal** in $k \Rightarrow$ there is an **ideal point** from which on we should **STOP at the first success!**

Odds Algorithm ("Sum the Odds to One and Stop", F. Thomas Bruss, 2000)

- Let k^* be the largest k such that $\sum_{j=k}^n r_j \geq 1$
- Ignore** everything before the k^* -th trial, then **STOP** at the **first** success.

- The **success probability** is $\sum_{j=k^*}^n r_j \cdot \left(\prod_{i=k^*}^n (1 - p_i) \right)$.
- This algorithm always executes the **optimal strategy!**

Illustration of the probability of having the last success ($n = 100$)



Source: Group Fibonado

Example 4

Use the Odds Algorithm to analyse the Secretary Problem.

Answer

- Let $I_j = 1$ if and only if secretary j is the best secretary so far.

Example 4

Use the Odds Algorithm to analyse the Secretary Problem.

Answer

- Let $I_j = 1$ if and only if secretary j is the best secretary so far.
- The I_j 's are independent (this is an question is on the exercise sheet)

Example 4

Use the **Odds Algorithm** to analyse the **Secretary Problem**.

Answer

- Let $I_j = 1$ if and only if secretary j is the best secretary so far.
- The I_j 's are **independent** (this is an question is on the exercise sheet)
- Then:

$$p_j = \mathbf{P} [I_j = 1] = \frac{1}{j}$$
$$r_j = \frac{p_j}{1 - p_j} = \frac{1/j}{(j-1)/j} = \frac{1}{j-1}$$

Example 4

Use the **Odds Algorithm** to analyse the **Secretary Problem**.

Answer

- Let $I_j = 1$ if and only if secretary j is the best secretary so far.
- The I_j 's are **independent** (this is an question is on the exercise sheet)
- Then:

$$p_j = \mathbf{P} [I_j = 1] = \frac{1}{j}$$
$$r_j = \frac{p_j}{1 - p_j} = \frac{1/j}{(j-1)/j} = \frac{1}{j-1}$$

- Largest k for which $\sum_{j=k}^n \frac{1}{j-1} \geq 1$ is $k = 1/e \cdot n$

Example 4

Use the **Odds Algorithm** to analyse the **Secretary Problem**.

Answer

- Let $I_j = 1$ if and only if secretary j is the best secretary so far.
- The I_j 's are **independent** (this is an question is on the exercise sheet)
- Then:

$$p_j = \mathbf{P} [I_j = 1] = \frac{1}{j}$$
$$r_j = \frac{p_j}{1 - p_j} = \frac{1/j}{(j-1)/j} = \frac{1}{j-1}$$

- Largest k for which $\sum_{j=k}^n \frac{1}{j-1} \geq 1$ is $k = 1/e \cdot n$
- Probability for success:

$$\mathbf{P} \left[\sum_{j=k}^n I_j = 1 \right] = \sum_{j=k}^n r_j \cdot \left(\prod_{i=k}^n (1 - p_i) \right)$$

Use the **Odds Algorithm** to analyse the **Secretary Problem**.

Answer

- Let $I_j = 1$ if and only if secretary j is the best secretary so far.
- The I_j 's are **independent** (this is an question is on the exercise sheet)
- Then:

$$p_j = \mathbf{P} [I_j = 1] = \frac{1}{j}$$

$$r_j = \frac{p_j}{1 - p_j} = \frac{1/j}{(j-1)/j} = \frac{1}{j-1}$$

- Largest k for which $\sum_{j=k}^n \frac{1}{j-1} \geq 1$ is $k = 1/e \cdot n$
- Probability for success:

$$\begin{aligned} \mathbf{P} \left[\sum_{j=k}^n I_j = 1 \right] &= \sum_{j=k}^n r_j \cdot \left(\prod_{i=k}^n (1 - p_i) \right) \\ &= \sum_{j=k}^n \frac{1}{j-1} \cdot \left(\prod_{i=k}^n \frac{i-1}{i} \right) \end{aligned}$$

Use the **Odds Algorithm** to analyse the **Secretary Problem**.

Answer

- Let $I_j = 1$ if and only if secretary j is the best secretary so far.
- The I_j 's are **independent** (this is an question is on the exercise sheet)
- Then:

$$p_j = \mathbf{P} [I_j = 1] = \frac{1}{j}$$

$$r_j = \frac{p_j}{1 - p_j} = \frac{1/j}{(j-1)/j} = \frac{1}{j-1}$$

- Largest k for which $\sum_{j=k}^n \frac{1}{j-1} \geq 1$ is $k = 1/e \cdot n$
- Probability for success:

$$\begin{aligned} \mathbf{P} \left[\sum_{j=k}^n I_j = 1 \right] &= \sum_{j=k}^n r_j \cdot \left(\prod_{i=k}^n (1 - p_i) \right) \\ &= \sum_{j=k}^n \frac{1}{j-1} \cdot \left(\prod_{i=k}^n \frac{i-1}{i} \right) \\ &= \sum_{j=k}^n \frac{1}{j-1} \cdot \frac{k-1}{n} \end{aligned}$$

Use the **Odds Algorithm** to analyse the **Secretary Problem**.

Answer

- Let $I_j = 1$ if and only if secretary j is the best secretary so far.
- The I_j 's are **independent** (this is an question is on the exercise sheet)
- Then:

$$p_j = \mathbf{P} [I_j = 1] = \frac{1}{j}$$

$$r_j = \frac{p_j}{1 - p_j} = \frac{1/j}{(j-1)/j} = \frac{1}{j-1}$$

- Largest k for which $\sum_{j=k}^n \frac{1}{j-1} \geq 1$ is $k = 1/e \cdot n$
- Probability for success:

$$\begin{aligned} \mathbf{P} \left[\sum_{j=k}^n I_j = 1 \right] &= \sum_{j=k}^n r_j \cdot \left(\prod_{i=k}^n (1 - p_i) \right) \\ &= \sum_{j=k}^n \frac{1}{j-1} \cdot \left(\prod_{i=k}^n \frac{i-1}{i} \right) \\ &= \sum_{j=k}^n \frac{1}{j-1} \cdot \frac{k-1}{n} \approx \frac{1}{e}. \end{aligned}$$

Use the **Odds Algorithm** to analyse the **Secretary Problem**.

Answer

- Let $I_j = 1$ if and only if secretary j is the best secretary so far.
- The I_j 's are **independent** (this is an question is on the exercise sheet)
- Then:

$$p_j = \mathbf{P} [I_j = 1] = \frac{1}{j}$$

$$r_j = \frac{p_j}{1 - p_j} = \frac{1/j}{(j-1)/j} = \frac{1}{j-1}$$

- Largest k for which $\sum_{j=k}^n \frac{1}{j-1} \geq 1$ is $k = 1/e \cdot n$
- Probability for success:

$$\begin{aligned} \mathbf{P} \left[\sum_{j=k}^n I_j = 1 \right] &= \sum_{j=k}^n r_j \cdot \left(\prod_{i=k}^n (1 - p_i) \right) \\ &= \sum_{j=k}^n \frac{1}{j-1} \cdot \left(\prod_{i=k}^n \frac{i-1}{i} \right) \\ &= \sum_{j=k}^n \frac{1}{j-1} \cdot \frac{k-1}{n} \approx \frac{1}{e}. \end{aligned}$$

We re-derived the solution of the **secretary problem** as a special case!

Outline

Stopping Problem 1: Dice Game

Stopping Problem 2: The Secretary Problem

A Generalisation: The Odds Algorithm (non-examinable)

The End...

- **Part I: Introduction to Probability**

- **Lecture 1:** Conditional probabilities and Bayes' theorem

- **Part II: Random Variables**

- **Lecture 2:** Random variables, probability mass function, expectation
- **Lecture 3:** Expectation properties, variance, discrete distributions
- **Lecture 4:** More discrete distributions: Poisson, Geometric, Negative Binomial
- **Lecture 5:** Continuous random variables
- **Lecture 6:** Marginals and Joint Distributions
- **Lecture 7:** Independence, Covariance and Correlation

- **Part III: Moments and Limit Theorems**

- **Lecture 8:** Basic Inequalities and Law of Large Numbers
- **Lecture 9:** Central Limit Theorem

- **Part IV: Applications and Statistics**

- **Lecture 10:** Estimators (Part I)
- **Lecture 11:** Estimators (Part II)
- **Lecture 12:** Online Algorithms

List of Distributions

Very Important:

- Bernoulli, Binomial, Poisson
- (Continuous) Uniform, Normal, Exponential

(Somewhat Less) Important:

- Geometric, Negative Binomial, Hypergeometric, Discrete Uniform

Not used or not defined in this course (and thus not examinable):

- Cauchy, Gamma, bivariate Normal
- Beta

Thank you and Best Wishes for the Exam!