Local parking algorithms on $\mathbb{Z}$

Philippe Nadeau (CNRS & Univ Lyon 1)

GT Combinatoire et Interactions, LaBRi, 2 mai 2022

What this talk is about

We define and study a large class of parking procedures that extend the classical parking algorithm in a new way.

What this talk is about

We define and study a large class of parking procedures that extend the classical parking algorithm in a new way.

Goal: Show that certain local conditions will ensure that the classical enumerative sequence $(r+1)^{r-1}$ holds "universally".

More generally, these local parking procedures exhibit nice combinatorics.

What this talk is about

We define and study a large class of parking procedures that extend the classical parking algorithm in a new way.

Goal: Show that certain local conditions will ensure that the classical enumerative sequence $(r+1)^{r-1}$ holds "universally".

More generally, these local parking procedures exhibit nice combinatorics.

(In particular, the goal is not to define any actual, reasonable parking procedure for actual, reasonable drivers.)

What this talk is about

We define and study a large class of parking procedures that extend the classical parking algorithm in a new way.

Goal: Show that certain local conditions will ensure that the classical enumerative sequence $(r+1)^{r-1}$ holds "universally".

More generally, these local parking procedures exhibit nice combinatorics.

(In particular, the goal is not to define any actual, reasonable parking procedure for actual, reasonable drivers.)

A word of warning

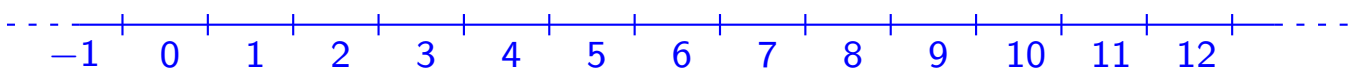
This is a work in progress. Bibliographical research was done, but some constructions and results may already be in the literature somewhere...
If you notice such an occurrence, please forgive my ignorance and let me know!

Classical Parking Functions

Classical parking procedure ([Konheim-Weiss '66]).

- r cars want to park on an empty street where the spots are labeled by $\mathbb{Z}$.
- The cars arrive successively, and the ith car has a preferred spot a_i.
- If the spot is available, it parks there.
- If not, it parks in the *nearest available spot on the right*.

Example. $a_1a_2a_3a_4a_5a_6a_7 = 3525895$

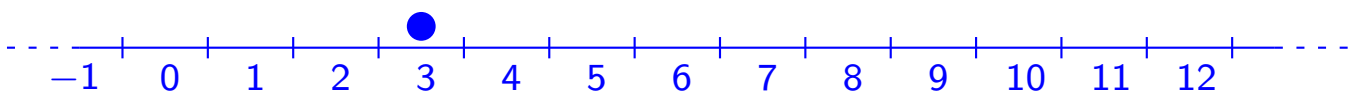


Classical Parking Functions

Classical parking procedure ([Konheim-Weiss '66]).

- r cars want to park on an empty street where the spots are labeled by $\mathbb{Z}$.
- The cars arrive successively, and the ith car has a preferred spot a_i.
- If the spot is available, it parks there.
- If not, it parks in the *nearest available spot on the right*.

Example. $a_1a_2a_3a_4a_5a_6a_7 = 3525895$

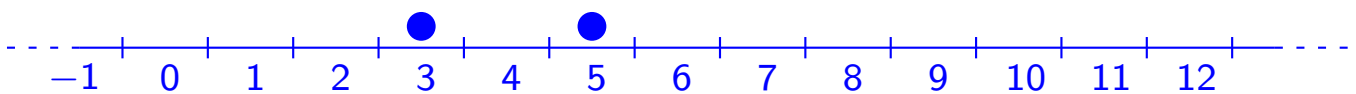


Classical Parking Functions

Classical parking procedure ([Konheim-Weiss '66]).

- r cars want to park on an empty street where the spots are labeled by $\mathbb{Z}$.
- The cars arrive successively, and the ith car has a preferred spot a_i.
- If the spot is available, it parks there.
- If not, it parks in the *nearest available spot on the right*.

Example. $a_1a_2a_3a_4a_5a_6a_7 = 3525895$

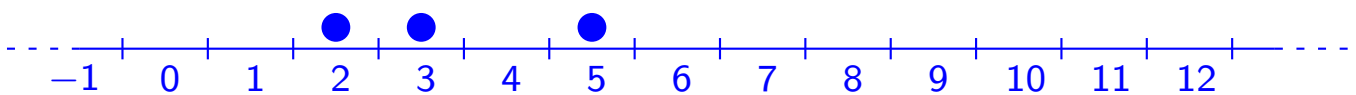


Classical Parking Functions

Classical parking procedure ([Konheim-Weiss '66]).

- r cars want to park on an empty street where the spots are labeled by $\mathbb{Z}$.
- The cars arrive successively, and the ith car has a preferred spot a_i.
- If the spot is available, it parks there.
- If not, it parks in the *nearest available spot on the right*.

Example. $a_1a_2a_3a_4a_5a_6a_7 = 3525895$



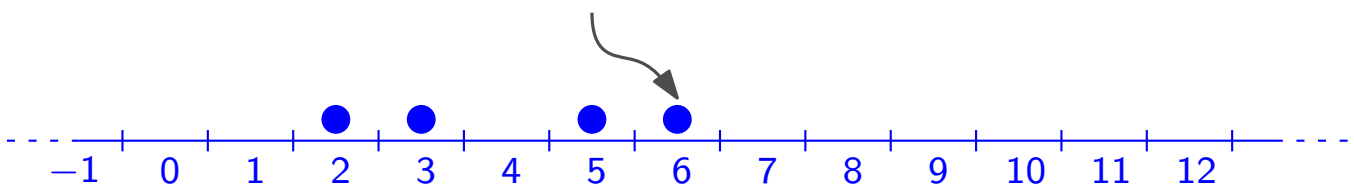
Classical Parking Functions

Classical parking procedure ([Konheim-Weiss '66]).

- r cars want to park on an empty street where the spots are labeled by $\mathbb{Z}$.
- The cars arrive successively, and the ith car has a preferred spot a_i.
- If the spot is available, it parks there.
- If not, it parks in the *nearest available spot on the right*.

Example.

$$a_1a_2a_3a_4a_5a_6a_7 = 3525895$$



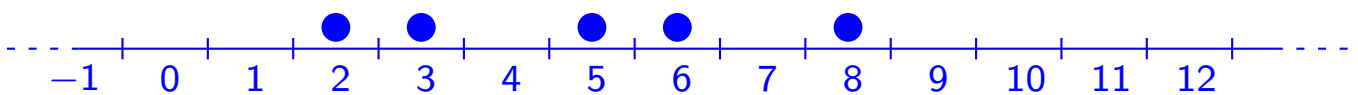
Classical Parking Functions

Classical parking procedure ([Konheim-Weiss '66]).

- r cars want to park on an empty street where the spots are labeled by $\mathbb{Z}$.
- The cars arrive successively, and the ith car has a preferred spot a_i.
- If the spot is available, it parks there.
- If not, it parks in the *nearest available spot on the right*.

Example.

$$a_1a_2a_3a_4a_5a_6a_7 = 3525895$$



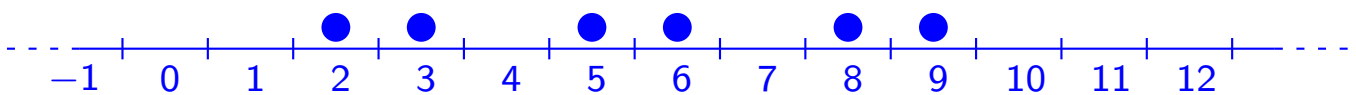
Classical Parking Functions

Classical parking procedure ([Konheim-Weiss '66]).

- r cars want to park on an empty street where the spots are labeled by $\mathbb{Z}$.
- The cars arrive successively, and the ith car has a preferred spot a_i.
- If the spot is available, it parks there.
- If not, it parks in the *nearest available spot on the right*.

Example.

$$a_1a_2a_3a_4a_5a_6a_7 = 3525895$$



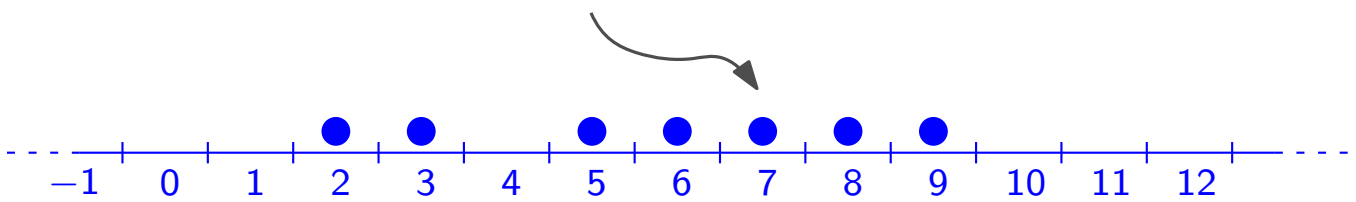
Classical Parking Functions

Classical parking procedure ([Konheim-Weiss '66]).

- r cars want to park on an empty street where the spots are labeled by $\mathbb{Z}$.
- The cars arrive successively, and the ith car has a preferred spot a_i.
- If the spot is available, it parks there.
- If not, it parks in the *nearest available spot on the right*.

Example.

$$a_1a_2a_3a_4a_5a_6a_7 = 3525895$$

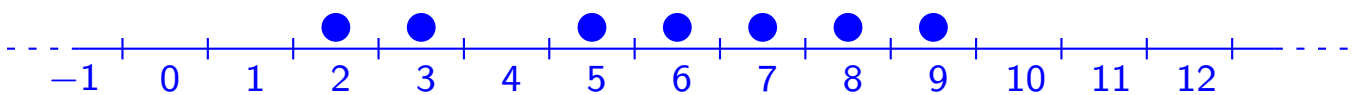


Classical Parking Functions

Classical parking procedure ([Konheim-Weiss '66]).

- r cars want to park on an empty street where the spots are labeled by $\mathbb{Z}$.
- The cars arrive successively, and the ith car has a preferred spot a_i.
- If the spot is available, it parks there.
- If not, it parks in the *nearest available spot on the right*.

Example. $a_1a_2a_3a_4a_5a_6a_7 = 3525895$



Definition. A word $a_1a_2 \dots a_r$ is called a parking word (or function) if the parking procedure ends up with all spots $1, \dots, r$ occupied.

Example.

$(r = 1)$ 1
$(r = 2)$ $11, 12, 21$
$(r = 3)$ $111, 112, 113, 121, 122, 123, 131, 132, 211, 212, 213, 221, 231, 311, 312, 321$

Classical Parking Functions

Proposition. The number of parking functions of length r is $(r+1)^{r-1}$

Proof. Elegant proof by Pollak (c. '74), which we generalize later. □

Proposition. $a_1 a_2 \cdots a_r$ is a parking function
$\Leftrightarrow$ for $k = 1, \ldots, r$, there are at least k indices i such that $1 \leq a_i \leq k$.

Corollary (Abelian property). As a consequence, if $a_1 a_2 \cdots a_r$ is a parking function then so is $a_{\sigma(1)} \cdots a_{\sigma(r)}$ for any permutation σ of $\{1, \ldots, r\}$.

$(r = 1)$ 1
$(r = 2)$ 11, 12, 21
$(r = 3)$ 111, 112, 113, 121, 122, 123, 131, 132, 211, 212, 213, 221, 231, 311, 312, 321

Classical Parking Functions

Proposition. The number of parking functions of length r is $(r+1)^{r-1}$

Proof. Elegant proof by Pollak (c. '74), which we generalize later. □

Proposition. $a_1 a_2 \cdots a_r$ is a parking function
$\Leftrightarrow$ for $k = 1, \ldots, r$, there are at least k indices i such that $1 \leq a_i \leq k$.

Corollary (Abelian property). As a consequence, if $a_1 a_2 \cdots a_r$ is a parking function then so is $a_{\sigma(1)} \cdots a_{\sigma(r)}$ for any permutation σ of $\{1, \ldots, r\}$.

Occurrences of parking functions (see various talks by R. Stanley, survey by C. Yan).

- Minimal factorizations of the long cycle $(1, 2, \ldots, r+1)$ in transpositions.
- Maximal chains in the lattice of noncrossing partitions
- Diagonal harmonics for the symmetric group.
- Regions of the Shi hyperplane arrangement $\{x_i - x_j = 0, 1\}$
- Volume of the Pitman-Stanley polytope.

Classical Parking Functions

Proposition. The number of parking functions of length r is $(r+1)^{r-1}$

Proof. Elegant proof by Pollak (c. '74), which we generalize later. □

Proposition. $a_1 a_2 \cdots a_r$ is a parking function
$\Leftrightarrow$ for $k = 1, \ldots, r$, there are at least k indices i such that $1 \leq a_i \leq k$.

Corollary (Abelian property). As a consequence, if $a_1 a_2 \cdots a_r$ is a parking function then so is $a_{\sigma(1)} \cdots a_{\sigma(r)}$ for any permutation σ of $\{1, \ldots, r\}$.

Occurrences of parking functions (see various talks by R. Stanley, survey by C. Yan).

- Minimal factorizations of the long cycle $(1, 2, \ldots, r+1)$ in transpositions.
- Maximal chains in the lattice of noncrossing partitions
- Diagonal harmonics for the symmetric group.
- Regions of the Shi hyperplane arrangement $\{x_i - x_j = 0, 1\}$
- Volume of the Pitman-Stanley polytope.

Generalizations: **u**-parking functions, G-parking functions...

Generalized Parking procedures

We first relax the parking condition as follows :

"If your desired spot is occupied, you must park in the next spot available either to the left or to the right."

Generalized Parking procedures

We first relax the parking condition as follows :

"If your desired spot is occupied, you must park in the next spot available either to the left or to the right."

In more mathematical terms, a parking procedure will be modeled by a function

$$\mathcal{P} : \mathbb{Z}^* = \{\text{Words over } \mathbb{Z}\} \to \text{Finset}(\mathbb{Z}) = \{\text{Finite subsets of } \mathbb{Z}\}.$$

"List of wanted spots" "Set of occupied spots"

satisfying three natural conditions:

Generalized Parking procedures

We first relax the parking condition as follows :

"If your desired spot is occupied, you must park in the next spot available either to the left or to the right."

In more mathematical terms, a parking procedure will be modeled by a function

$$\mathcal{P} : \mathbb{Z}^* = \{\text{Words over } \mathbb{Z}\} \to \text{Finset}(\mathbb{Z}) = \{\text{Finite subsets of } \mathbb{Z}\}.$$

"List of wanted spots" "Set of occupied spots"

satisfying three natural conditions:

1. (Accessible parking) For any $a_1, a_2, \ldots \in \mathbb{Z}$, $\#\mathcal{P}(a_1 \cdots a_i) = i \; \forall i$, and $\emptyset = \mathcal{P}(\epsilon) \subset \mathcal{P}(a_1) \subset \cdots \subset \mathcal{P}(a_1 \cdots a_i) \subset \mathcal{P}(a_1 \cdots a_{i+1}) \subset \cdots$

Generalized Parking procedures

We first relax the parking condition as follows :

"If your desired spot is occupied, you must park in the next spot available either to the left or to the right."

In more mathematical terms, a parking procedure will be modeled by a function

$$\mathcal{P} : \mathbb{Z}^* = \{\text{Words over } \mathbb{Z}\} \to \text{Finset}(\mathbb{Z}) = \{\text{Finite subsets of } \mathbb{Z}\}.$$

"List of wanted spots" "Set of occupied spots"

satisfying three natural conditions:

1. (Accessible parking) For any $a_1, a_2, \ldots \in \mathbb{Z}$, $\#\mathcal{P}(a_1 \cdots a_i) = i \; \forall i$, and $\emptyset = \mathcal{P}(\epsilon) \subset \mathcal{P}(a_1) \subset \cdots \subset \mathcal{P}(a_1 \cdots a_i) \subset \mathcal{P}(a_1 \cdots a_{i+1}) \subset \cdots$

- $\{\text{lastSpot}_{\mathcal{P}}(Wa)\} = \mathcal{P}(Wa) \setminus \mathcal{P}(W)$. (= the spot where the car eventually parks.)
- $\pi_{\mathcal{P}}^{a_1 \cdots a_r}$: $\mathcal{P}(a_1 \cdots a_r) \to \{1, \ldots, r\}$ is a bijection.

  $\text{lastSpot}_{\mathcal{P}}(a_1 a_2 \cdots a_i) \mapsto i$

Generalized Parking procedures

We first relax the parking condition as follows :

"If your desired spot is occupied, you must park in the next spot available either to the left or to the right."

In more mathematical terms, a parking procedure will be modeled by a function

$$\mathcal{P} : \mathbb{Z}^* = \{\text{Words over } \mathbb{Z}\} \to \text{Finset}(\mathbb{Z}) = \{\text{Finite subsets of } \mathbb{Z}\}.$$

"List of wanted spots" "Set of occupied spots"

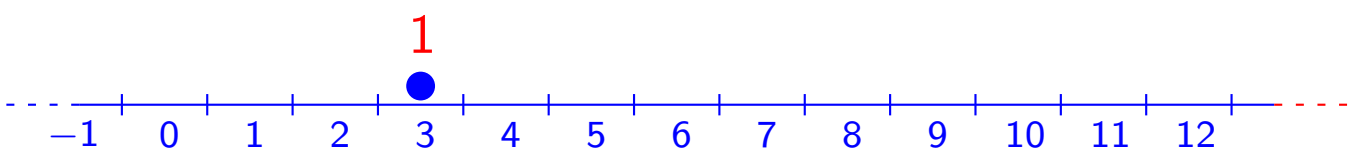
satisfying three natural conditions:

1. (Accessible parking) For any $a_1, a_2, \ldots \in \mathbb{Z}$, $\#\mathcal{P}(a_1 \cdots a_i) = i \; \forall i$, and $\emptyset = \mathcal{P}(\epsilon) \subset \mathcal{P}(a_1) \subset \cdots \subset \mathcal{P}(a_1 \cdots a_i) \subset \mathcal{P}(a_1 \cdots a_{i+1}) \subset \cdots$

- $\{\text{lastSpot}_{\mathcal{P}}(Wa)\} = \mathcal{P}(Wa) \setminus \mathcal{P}(W)$. (= the spot where the car eventually parks.)
- $\pi_{\mathcal{P}}^{a_1 \cdots a_r}$: $\mathcal{P}(a_1 \cdots a_r) \to \{1, \ldots, r\}$ is a bijection.

  $\text{lastSpot}_{\mathcal{P}}(a_1 a_2 \cdots a_i) \mapsto i$

$W = 3525895$
1234567



Generalized Parking procedures

We first relax the parking condition as follows :

"If your desired spot is occupied, you must park in the next spot available either to the left or to the right."

In more mathematical terms, a parking procedure will be modeled by a function

$$\mathcal{P} : \mathbb{Z}^* = \{\text{Words over } \mathbb{Z}\} \to \text{Finset}(\mathbb{Z}) = \{\text{Finite subsets of } \mathbb{Z}\}.$$

"List of wanted spots" "Set of occupied spots"

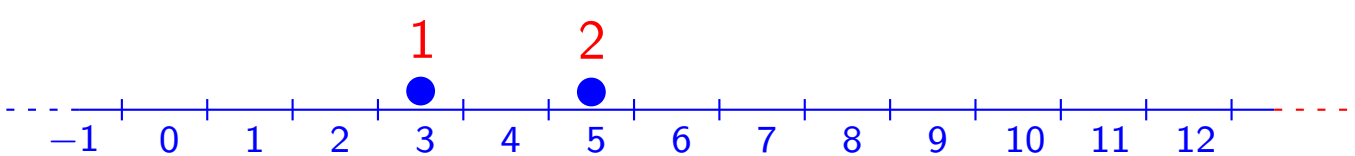
satisfying three natural conditions:

1. (Accessible parking) For any $a_1, a_2, \ldots \in \mathbb{Z}$, $\#\mathcal{P}(a_1 \cdots a_i) = i \; \forall i$, and $\emptyset = \mathcal{P}(\epsilon) \subset \mathcal{P}(a_1) \subset \cdots \subset \mathcal{P}(a_1 \cdots a_i) \subset \mathcal{P}(a_1 \cdots a_{i+1}) \subset \cdots$

- $\{\text{lastSpot}_{\mathcal{P}}(Wa)\} = \mathcal{P}(Wa) \setminus \mathcal{P}(W)$. (= the spot where the car eventually parks.)
- $\pi_{\mathcal{P}}^{a_1 \cdots a_r}$: $\mathcal{P}(a_1 \cdots a_r) \to \{1, \ldots, r\}$ is a bijection.

  $\text{lastSpot}_{\mathcal{P}}(a_1 a_2 \cdots a_i) \mapsto i$

$W = 3525895$
1234567



Generalized Parking procedures

We first relax the parking condition as follows :

"If your desired spot is occupied, you must park in the next spot available either to the left or to the right."

In more mathematical terms, a parking procedure will be modeled by a function

$$\mathcal{P} : \mathbb{Z}^* = \{\text{Words over } \mathbb{Z}\} \to \text{Finset}(\mathbb{Z}) = \{\text{Finite subsets of } \mathbb{Z}\}.$$

"List of wanted spots" "Set of occupied spots"

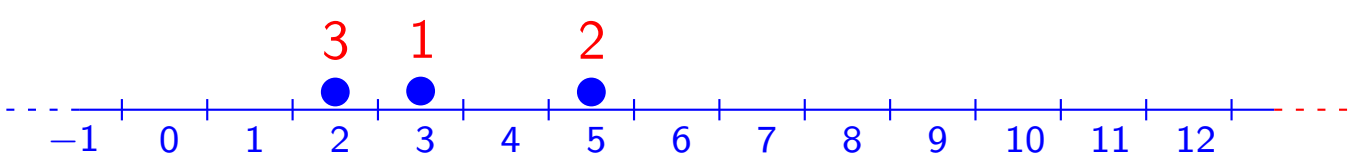
satisfying three natural conditions:

1. (Accessible parking) For any $a_1, a_2, \ldots \in \mathbb{Z}$, $\#\mathcal{P}(a_1 \cdots a_i) = i \; \forall i$, and $\emptyset = \mathcal{P}(\epsilon) \subset \mathcal{P}(a_1) \subset \cdots \subset \mathcal{P}(a_1 \cdots a_i) \subset \mathcal{P}(a_1 \cdots a_{i+1}) \subset \cdots$

- $\{\text{lastSpot}_{\mathcal{P}}(Wa)\} = \mathcal{P}(Wa) \setminus \mathcal{P}(W)$. (= the spot where the car eventually parks.)
- $\pi_{\mathcal{P}}^{a_1 \cdots a_r}$: $\mathcal{P}(a_1 \cdots a_r) \to \{1, \ldots, r\}$ is a bijection.

  $\text{lastSpot}_{\mathcal{P}}(a_1 a_2 \cdots a_i) \mapsto i$

$W = 3525895$
1234567



Generalized Parking procedures

We first relax the parking condition as follows :

"If your desired spot is occupied, you must park in the next spot available either to the left or to the right."

In more mathematical terms, a parking procedure will be modeled by a function

$$\mathcal{P} : \mathbb{Z}^* = \{\text{Words over } \mathbb{Z}\} \to \text{Finset}(\mathbb{Z}) = \{\text{Finite subsets of } \mathbb{Z}\}.$$

"List of wanted spots" "Set of occupied spots"

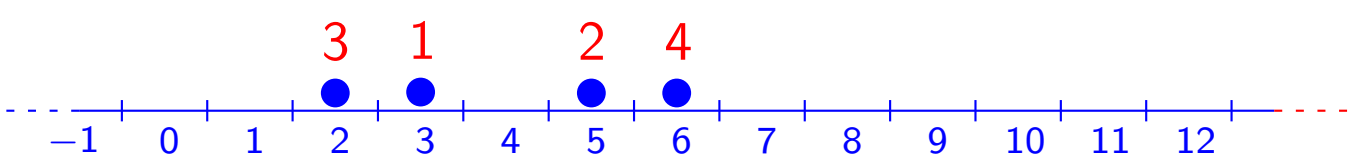
satisfying three natural conditions:

1. (Accessible parking) For any $a_1, a_2, \ldots \in \mathbb{Z}$, $\#\mathcal{P}(a_1 \cdots a_i) = i \; \forall i$, and $\emptyset = \mathcal{P}(\epsilon) \subset \mathcal{P}(a_1) \subset \cdots \subset \mathcal{P}(a_1 \cdots a_i) \subset \mathcal{P}(a_1 \cdots a_{i+1}) \subset \cdots$

- $\{\text{lastSpot}_{\mathcal{P}}(Wa)\} = \mathcal{P}(Wa) \setminus \mathcal{P}(W)$. (= the spot where the car eventually parks.)
- $\pi_{\mathcal{P}}^{a_1 \cdots a_r}$: $\mathcal{P}(a_1 \cdots a_r) \to \{1, \ldots, r\}$ is a bijection.

  $\text{lastSpot}_{\mathcal{P}}(a_1 a_2 \cdots a_i) \mapsto i$

$W = 3525895$
1234567



Generalized Parking procedures

We first relax the parking condition as follows :

"If your desired spot is occupied, you must park in the next spot available either to the left or to the right."

In more mathematical terms, a parking procedure will be modeled by a function

$$\mathcal{P} : \mathbb{Z}^* = \{\text{Words over } \mathbb{Z}\} \to \text{Finset}(\mathbb{Z}) = \{\text{Finite subsets of } \mathbb{Z}\}.$$

"List of wanted spots" "Set of occupied spots"

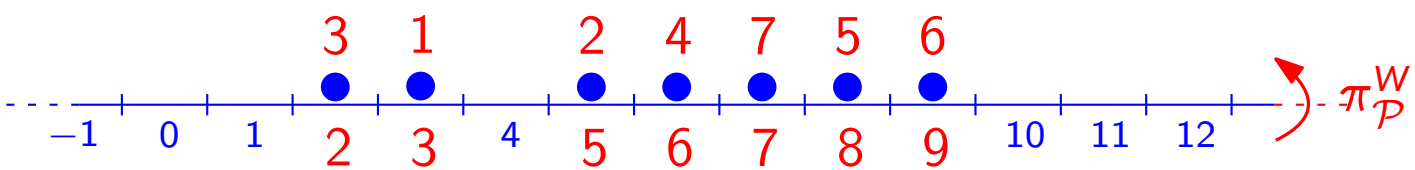
satisfying three natural conditions:

1. (Accessible parking) For any $a_1, a_2, \ldots \in \mathbb{Z}$, $\#\mathcal{P}(a_1 \cdots a_i) = i \; \forall i$, and $\emptyset = \mathcal{P}(\epsilon) \subset \mathcal{P}(a_1) \subset \cdots \subset \mathcal{P}(a_1 \cdots a_i) \subset \mathcal{P}(a_1 \cdots a_{i+1}) \subset \cdots$

- $\{\text{lastSpot}_{\mathcal{P}}(Wa)\} = \mathcal{P}(Wa) \setminus \mathcal{P}(W)$. (= the spot where the car eventually parks.)
- $\pi_{\mathcal{P}}^{a_1 \cdots a_r}$: $\mathcal{P}(a_1 \cdots a_r) \to \{1, \ldots, r\}$ is a bijection.

  $\text{lastSpot}_{\mathcal{P}}(a_1 a_2 \cdots a_i) \mapsto i$

$W = 3525895$
1234567



Generalized Parking procedures

- If your desired spot is free, park there:

2. (Lucky parking) If $a \notin \mathcal{P}(W)$, then $\mathrm{lastSpot}_{\mathcal{P}}(Wa) = a$.

Generalized Parking procedures

- If your desired spot is free, park there:

  2. (Lucky parking) If $a \notin \mathcal{P}(W)$, then $\mathrm{lastSpot}_{\mathcal{P}}(Wa) = a$.

- We call interval I of $F \in \mathrm{Finset}(\mathbb{Z})$ a maximal subset of consecutive integers. The last condition encodes "next available spot left or right".

Generalized Parking procedures

- If your desired spot is free, park there:

2. (Lucky parking) If $a \notin \mathcal{P}(W)$, then $\mathrm{lastSpot}_{\mathcal{P}}(Wa) = a$.

- We call interval I of $F \in \mathrm{Finset}(\mathbb{Z})$ a maximal subset of consecutive integers.
  The last condition encodes "next available spot left or right".

3. (Close parking) If $a \in \mathcal{P}(W)$, let $[t, u]$ be the interval of $\mathcal{P}(W)$ such that $a \in [t, u]$.
Then $\mathrm{lastSpot}_{\mathcal{P}}(Wa) \in \{t-1, u+1\}$. $(= \{\mathit{Left}, \mathit{Right}\})$

$\mathcal{P}_{\mathit{classical}}$ consists in picking $u+1$ ($=$ "Right") each time.

Generalized Parking procedures

- If your desired spot is free, park there:

2. (Lucky parking) If $a \notin \mathcal{P}(W)$, then $\text{lastSpot}_{\mathcal{P}}(Wa) = a$.

- We call interval I of $F \in \text{Finset}(\mathbb{Z})$ a maximal subset of consecutive integers.
  The last condition encodes "next available spot left or right".

3. (Close parking) If $a \in \mathcal{P}(W)$, let $[t, u]$ be the interval of $\mathcal{P}(W)$ such that $a \in [t, u]$.
Then $\text{lastSpot}_{\mathcal{P}}(Wa) \in \{t-1, u+1\}$. $(= \{\textit{Left}, \textit{Right}\})$

$\mathcal{P}_{classical}$ consists in picking $u+1$ ($=$ "Right") each time.

Definition. A parking procedure is a function $\mathcal{P} : \mathbb{Z}^* \to \text{Finset}(\mathbb{Z})$ satisfying the three conditions (Accessible parking),(Lucky parking),(Close parking).

Definition. A $\mathcal{P}$-parking word is a word $a_1 \cdots a_r$ such that $\mathcal{P}(a_1 \cdots a_r) = \{1, \dots, r\}$.

Generalized Parking procedures

- If your desired spot is free, park there:

2. (Lucky parking) If $a \notin \mathcal{P}(W)$, then $\text{lastSpot}_{\mathcal{P}}(Wa) = a$.

- We call interval I of $F \in \text{Finset}(\mathbb{Z})$ a maximal subset of consecutive integers.
  The last condition encodes "next available spot left or right".

3. (Close parking) If $a \in \mathcal{P}(W)$, let $[t, u]$ be the interval of $\mathcal{P}(W)$ such that $a \in [t, u]$.
Then $\text{lastSpot}_{\mathcal{P}}(Wa) \in \{t-1, u+1\}$. $(= \{\textit{Left}, \textit{Right}\})$

$\mathcal{P}_{classical}$ consists in picking $u+1$ ($=$ "Right") each time.

Definition. A parking procedure is a function $\mathcal{P} : \mathbb{Z}^* \to \text{Finset}(\mathbb{Z})$ satisfying the three conditions (Accessible parking),(Lucky parking),(Close parking).

Definition. A $\mathcal{P}$-parking word is a word $a_1 \cdots a_r$ such that $\mathcal{P}(a_1 \cdots a_r) = \{1, \dots, r\}$.

Remark (1). A $\mathcal{P}$-parking word of length r necessarily has all letters $a_i \in \{1, \dots, r\}$.

Remark (2). Permutations of $\{1, \dots, r\}$ are $\mathcal{P}$-parking for any procedure $\mathcal{P}$.

The procedure $\mathcal{P}_{CS}$

To characterize a parking procedure $\mathcal{P}$, we simply need a rule to pick Left or Right when our desired spot is already occupied.

The procedure $\mathcal{P}_{CS}$

To characterize a parking procedure $\mathcal{P}$, we simply need a rule to pick Left or Right when our desired spot is already occupied.

Definition (Procedure $\mathcal{P}_{CS}$). Let $a \in I$ interval of $\mathcal{P}(a_1 \cdots a_r)$. Let $j \in \{1, \dots, r\}$ be the maximal index such that $a_j \in I$.
Rule : If $a < a_j$ park left of I, and if $a \geq a_j$ park right.

[N.-Tewari '22+]

The procedure $\mathcal{P}_{CS}$

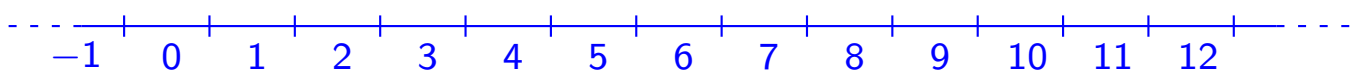
To characterize a parking procedure $\mathcal{P}$, we simply need a rule to pick Left or Right when our desired spot is already occupied.

Definition (Procedure $\mathcal{P}_{CS}$). Let $a \in I$ interval of $\mathcal{P}(a_1 \cdots a_r)$. Let $j \in \{1, \dots, r\}$ be the maximal index such that $a_j \in I$.
Rule : If $a < a_j$ park left of I, and if $a \geq a_j$ park right.

[N.-Tewari '22+]

Example.

$$a_1 a_2 a_3 a_4 a_5 a_6 a_7 = 5.11.8.3.8.4.3$$



The procedure $\mathcal{P}_{CS}$

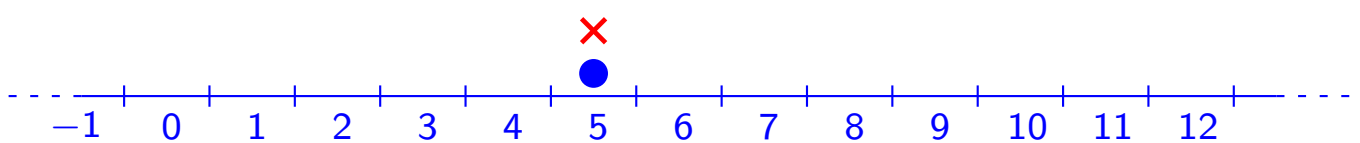
To characterize a parking procedure $\mathcal{P}$, we simply need a rule to pick Left or Right when our desired spot is already occupied.

Definition (Procedure $\mathcal{P}_{CS}$). Let $a \in I$ interval of $\mathcal{P}(a_1 \cdots a_r)$. Let $j \in \{1, \dots, r\}$ be the maximal index such that $a_j \in I$.
Rule : If $a < a_j$ park left of I, and if $a \geq a_j$ park right.

[N.-Tewari '22+]

Example.

$$a_1 a_2 a_3 a_4 a_5 a_6 a_7 = 5.11.8.3.8.4.3$$



The crosses record the values 'a_j' in each interval.

The procedure $\mathcal{P}_{CS}$

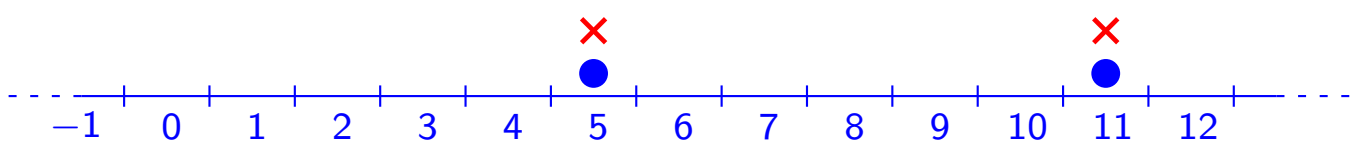
To characterize a parking procedure $\mathcal{P}$, we simply need a rule to pick Left or Right when our desired spot is already occupied.

Definition (Procedure $\mathcal{P}_{CS}$). Let $a \in I$ interval of $\mathcal{P}(a_1 \cdots a_r)$. Let $j \in \{1, \dots, r\}$ be the maximal index such that $a_j \in I$.
Rule : If $a < a_j$ park left of I, and if $a \geq a_j$ park right.

[N.-Tewari '22+]

Example.

$$a_1 a_2 a_3 a_4 a_5 a_6 a_7 = 5.11.8.3.8.4.3$$



The crosses record the values 'a_j' in each interval.

The procedure $\mathcal{P}_{CS}$

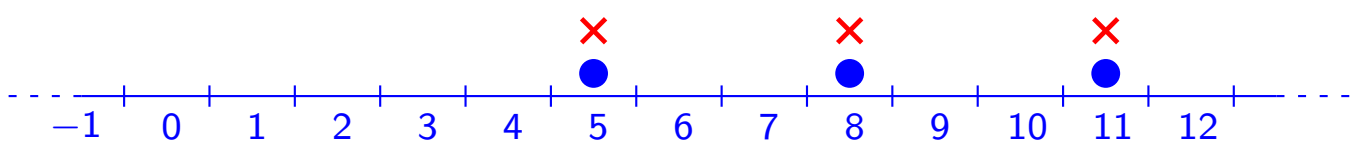
To characterize a parking procedure $\mathcal{P}$, we simply need a rule to pick Left or Right when our desired spot is already occupied.

Definition (Procedure $\mathcal{P}_{CS}$). Let $a \in I$ interval of $\mathcal{P}(a_1 \cdots a_r)$. Let $j \in \{1, \dots, r\}$ be the maximal index such that $a_j \in I$.
Rule : If $a < a_j$ park left of I, and if $a \geq a_j$ park right.

[N.-Tewari '22+]

Example.

$$a_1 a_2 a_3 a_4 a_5 a_6 a_7 = 5.11.8.3.8.4.3$$



The crosses record the values 'a_j' in each interval.

The procedure $\mathcal{P}_{CS}$

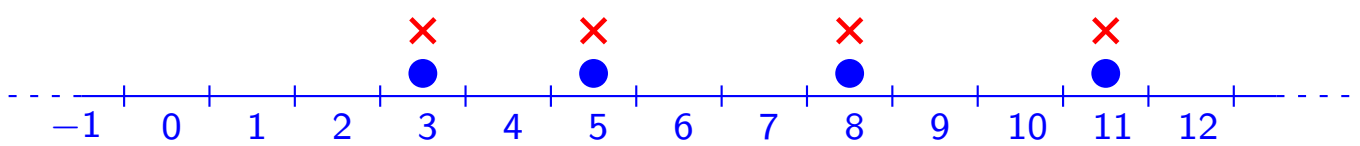
To characterize a parking procedure $\mathcal{P}$, we simply need a rule to pick Left or Right when our desired spot is already occupied.

Definition (Procedure $\mathcal{P}_{CS}$). Let $a \in I$ interval of $\mathcal{P}(a_1 \cdots a_r)$. Let $j \in \{1, \dots, r\}$ be the maximal index such that $a_j \in I$.
Rule : If $a < a_j$ park left of I, and if $a \geq a_j$ park right.

[N.-Tewari '22+]

Example.

$$a_1 a_2 a_3 a_4 a_5 a_6 a_7 = 5.11.8.3.8.4.3$$



The crosses record the values 'a_j' in each interval.

The procedure $\mathcal{P}_{CS}$

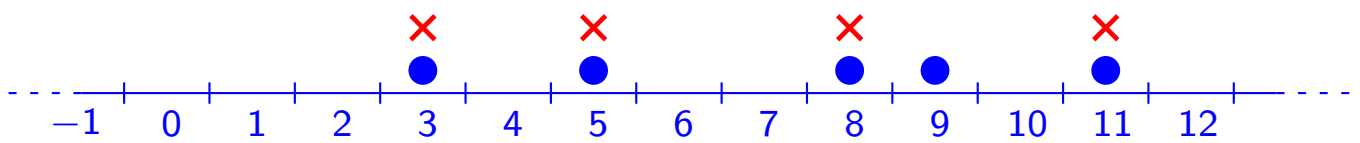
To characterize a parking procedure $\mathcal{P}$, we simply need a rule to pick Left or Right when our desired spot is already occupied.

Definition (Procedure $\mathcal{P}_{CS}$). Let $a \in I$ interval of $\mathcal{P}(a_1 \cdots a_r)$. Let $j \in \{1, \dots, r\}$ be the maximal index such that $a_j \in I$.
Rule : If $a < a_j$ park left of I, and if $a \geq a_j$ park right.

[N.-Tewari '22+]

Example.

$$a_1 a_2 a_3 a_4 a_5 a_6 a_7 = 5.11.8.3.8.4.3$$



The crosses record the values 'a_j' in each interval.

The procedure $\mathcal{P}_{CS}$

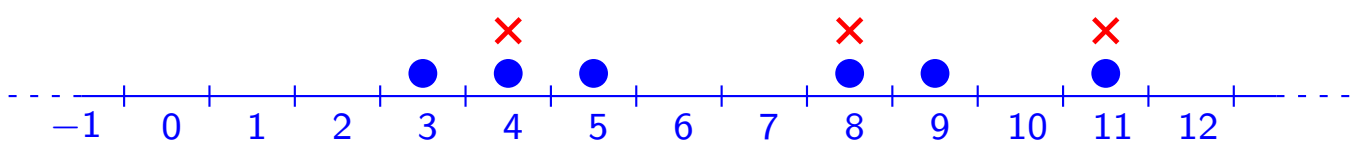
To characterize a parking procedure $\mathcal{P}$, we simply need a rule to pick Left or Right when our desired spot is already occupied.

Definition (Procedure $\mathcal{P}_{CS}$). Let $a \in I$ interval of $\mathcal{P}(a_1 \cdots a_r)$. Let $j \in \{1, \dots, r\}$ be the maximal index such that $a_j \in I$.
Rule : If $a < a_j$ park left of I, and if $a \geq a_j$ park right.

[N.-Tewari '22+]

Example.

$$a_1 a_2 a_3 a_4 a_5 a_6 a_7 = 5.11.8.3.8.4.3$$



The crosses record the values 'a_j' in each interval.

The procedure $\mathcal{P}_{CS}$

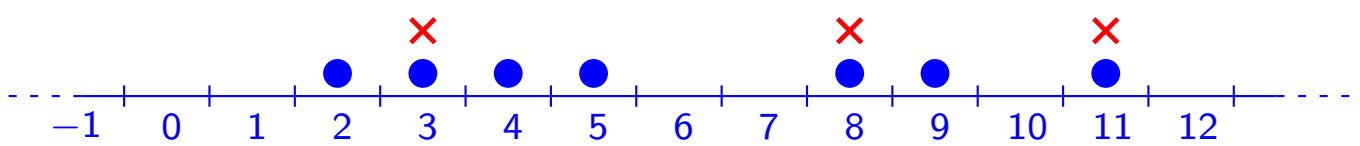
To characterize a parking procedure $\mathcal{P}$, we simply need a rule to pick Left or Right when our desired spot is already occupied.

Definition (Procedure $\mathcal{P}_{CS}$). Let $a \in I$ interval of $\mathcal{P}(a_1 \cdots a_r)$. Let $j \in \{1, \dots, r\}$ be the maximal index such that $a_j \in I$.
Rule : If $a < a_j$ park left of I, and if $a \geq a_j$ park right.

[N.-Tewari '22+]

Example.

$$a_1 a_2 a_3 a_4 a_5 a_6 a_7 = 5.11.8.3.8.4.3$$



The crosses record the values 'a_j' in each interval.

The procedure $\mathcal{P}_{CS}$

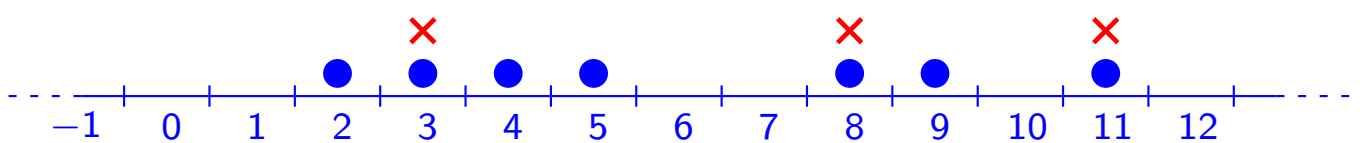
To characterize a parking procedure $\mathcal{P}$, we simply need a rule to pick Left or Right when our desired spot is already occupied.

Definition (Procedure $\mathcal{P}_{CS}$). Let $a \in I$ interval of $\mathcal{P}(a_1 \cdots a_r)$. Let $j \in \{1, \dots, r\}$ be the maximal index such that $a_j \in I$.
Rule : If $a < a_j$ park left of I, and if $a \geq a_j$ park right.

[N.-Tewari '22+]

Example.

$$a_1a_2a_3a_4a_5a_6a_7 = 5.11.8.3.8.4.3$$



The crosses record the values 'a_j' in each interval.

Proposition. The number of $\mathcal{P}_{CS}$-parking words of length r is $(r+1)^{r-1}$

For $r = 3$, they coincide with classical parking words except 232 is a $\mathcal{P}_{CS}$-parking word while 121 is not. In particular $\mathcal{P}_{CS}$ is not abelian.

The procedure $\mathcal{P}_{CS}$

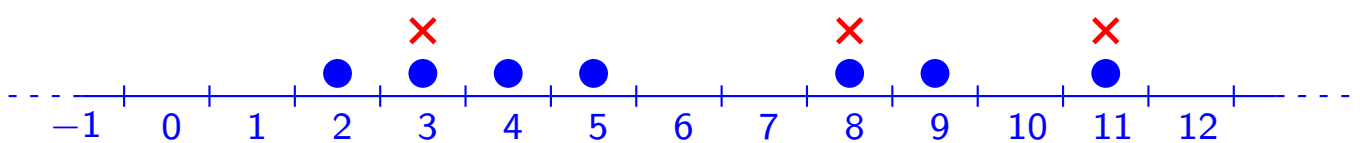
To characterize a parking procedure $\mathcal{P}$, we simply need a rule to pick Left or Right when our desired spot is already occupied.

Definition (Procedure $\mathcal{P}_{CS}$). Let $a \in I$ interval of $\mathcal{P}(a_1 \cdots a_r)$. Let $j \in \{1, \dots, r\}$ be the maximal index such that $a_j \in I$.
Rule : If $a < a_j$ park left of I, and if $a \geq a_j$ park right.

[N.-Tewari '22+]

Example.

$$a_1a_2a_3a_4a_5a_6a_7 = 5.11.8.3.8.4.3$$



The crosses record the values 'a_j' in each interval.

Proposition. The number of $\mathcal{P}_{CS}$-parking words of length r is $(r+1)^{r-1}$

For $r = 3$, they coincide with classical parking words except 232 is a $\mathcal{P}_{CS}$-parking word while 121 is not. In particular $\mathcal{P}_{CS}$ is not abelian.

Definition (Markov property). A procedure $\mathcal{P}$ is called *Markovian* if there is a function $M : \text{Finset}(\mathbb{Z}) \times \mathbb{Z} \to \text{Fin}(\mathbb{Z})$ such that $\mathcal{P}(a_1 \cdots a_r a) = M(\mathcal{P}(a_1 \cdots a_r), a)$

$\mathcal{P}_{CS}$ is not Markovian, while $\mathcal{P}_{classical}$ clearly is.

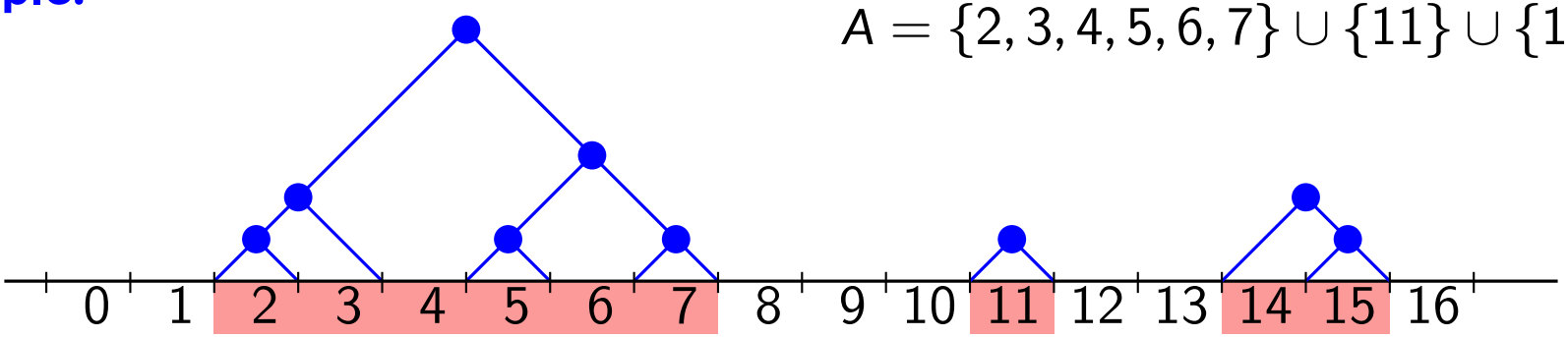
Encoding as binary forests

(**Goal**: Lift a parking procedure $\mathcal{P}$ to an injective encoding $\widehat{\mathcal{P}}$.)

Definition. A binary forest F with support $A \in \mathrm{Finset}(\mathbb{Z})$ is the data of a binary tree with $|I|$ nodes for each interval I of A.

Example.

$$A = \{2,3,4,5,6,7\} \cup \{11\} \cup \{14,15\}$$



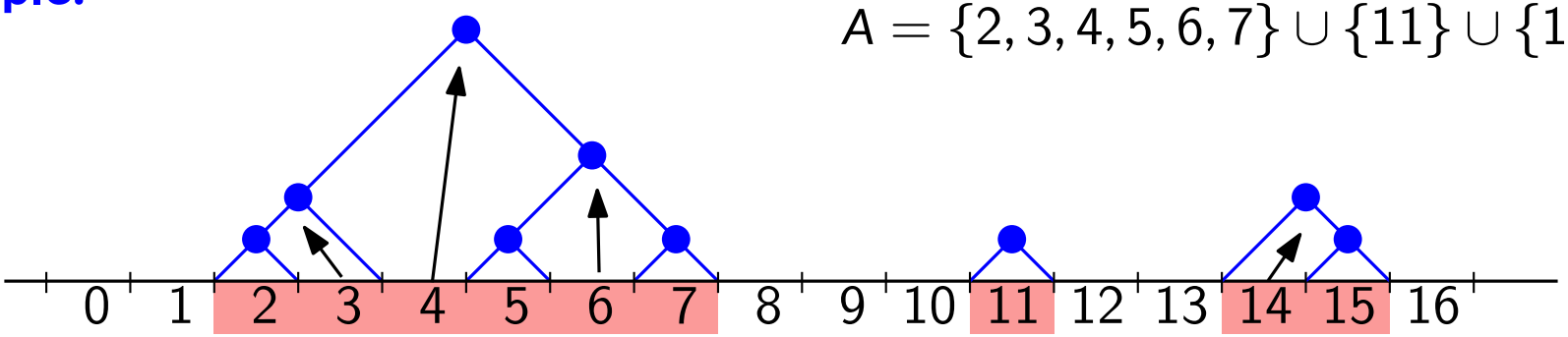
Encoding as binary forests

(**Goal**: Lift a parking procedure $\mathcal{P}$ to an injective encoding $\widehat{\mathcal{P}}$.)

Definition. A binary forest F with support $A \in \mathrm{Finset}(\mathbb{Z})$ is the data of a binary tree with $|I|$ nodes for each interval I of A.

Example.

$$A = \{2,3,4,5,6,7\} \cup \{11\} \cup \{14,15\}$$



Remark. The nodes are canonically labeled by an infix transversal.

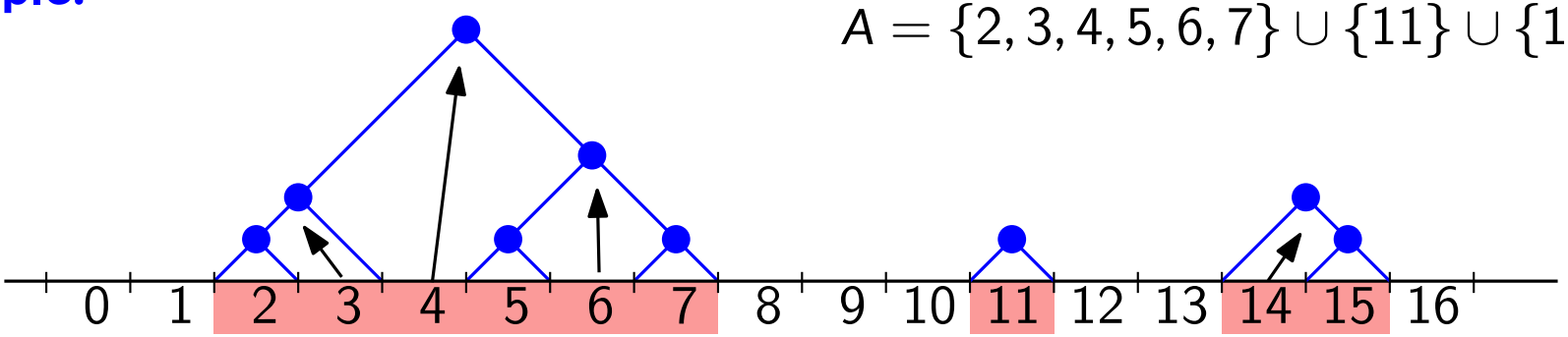
Encoding as binary forests

(**Goal**: Lift a parking procedure $\mathcal{P}$ to an injective encoding $\widehat{\mathcal{P}}$.)

Definition. A binary forest F with support $A \in \mathrm{Finset}(\mathbb{Z})$ is the data of a binary tree with $|I|$ nodes for each interval I of A.

Example.

$$A = \{2,3,4,5,6,7\} \cup \{11\} \cup \{14,15\}$$



Remark. The nodes are canonically labeled by an infix transversal.

Given any parking procedure $\mathcal{P}$, we now define

$$\widehat{\mathcal{P}} : W \in \mathbb{Z}^* \mapsto (P,Q) \in \bigcup_{F \text{ forest}} \mathcal{F}(F) \times \mathrm{Dec}(F).$$

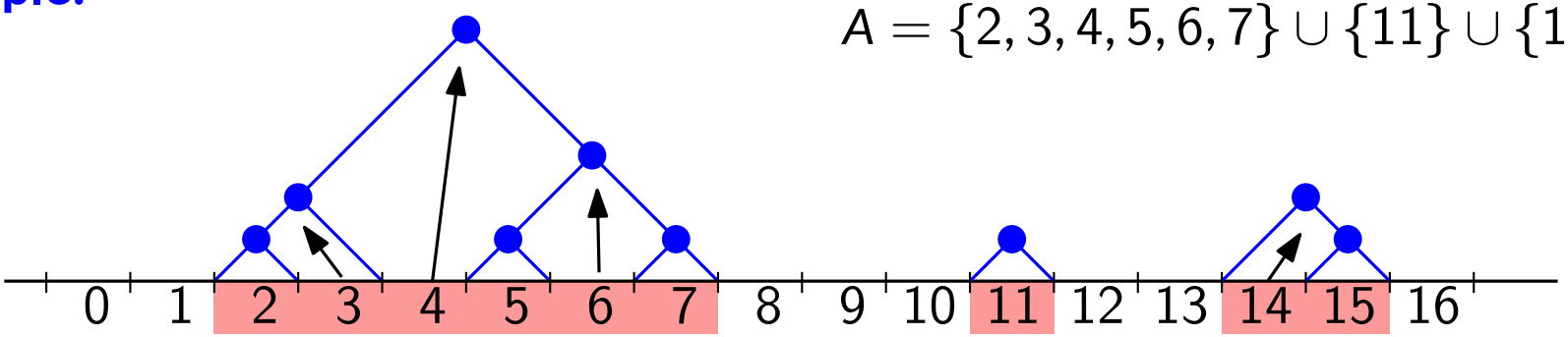
Encoding as binary forests

(**Goal**: Lift a parking procedure $\mathcal{P}$ to an injective encoding $\widehat{\mathcal{P}}$.)

Definition. A binary forest F with support $A \in \mathrm{Finset}(\mathbb{Z})$ is the data of a binary tree with $|I|$ nodes for each interval I of A.

Example.

$$A = \{2,3,4,5,6,7\} \cup \{11\} \cup \{14,15\}$$



Remark. The nodes are canonically labeled by an infix transversal.

Given any parking procedure $\mathcal{P}$, we now define

$$\widehat{\mathcal{P}} : W \in \mathbb{Z}^* \mapsto (P, Q) \in \bigcup_{F \text{ forest}} \mathcal{F}(F) \times \mathrm{Dec}(F).$$

Labelings of the forest F

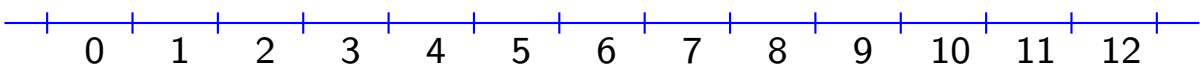
Information about the original desired spots

Decreasing forest, encodes exactly the bijection $\pi_{\mathcal{P}}^{W}$

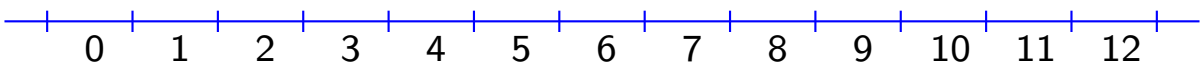
Encoding as binary forests

Definition (by example). $W = a_1a_2a_3a_4a_5a_6a_7 = 5.11.8.3.8.4.3$

$P(W)$



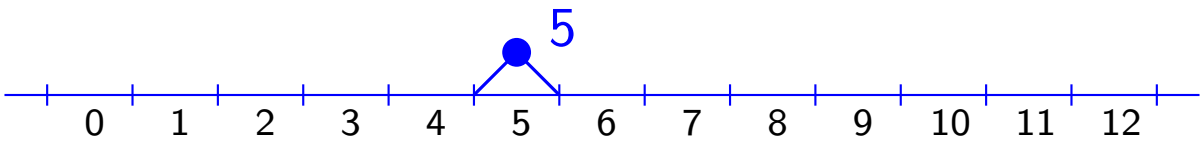
$Q(W)$



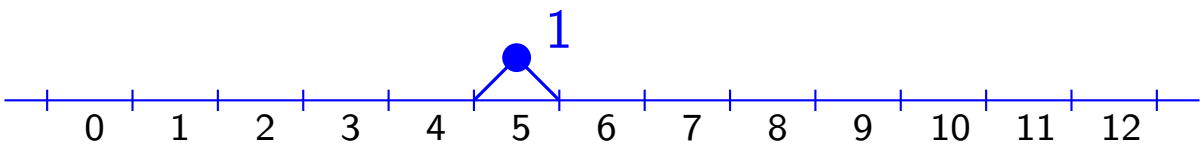
Encoding as binary forests

Definition (by example). $W = a_1a_2a_3a_4a_5a_6a_7 = 5.11.8.3.8.4.3$

$P(W)$



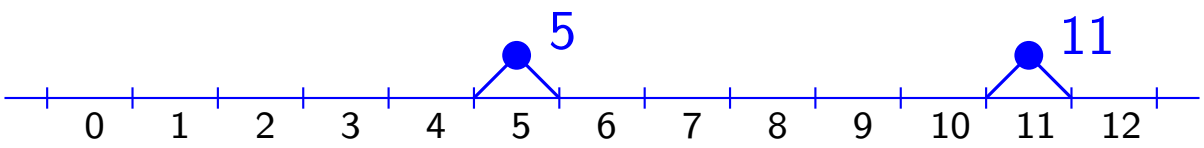
$Q(W)$



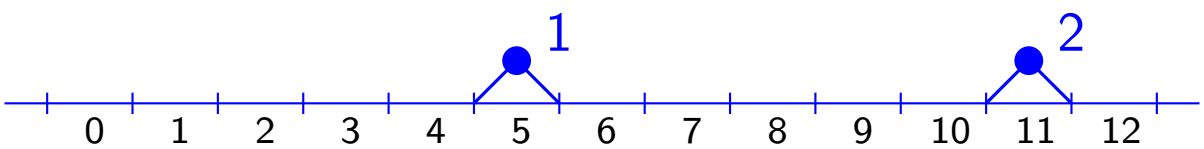
Encoding as binary forests

Definition (by example). $W = a_1a_2a_3a_4a_5a_6a_7 = 5.11.8.3.8.4.3$

$P(W)$



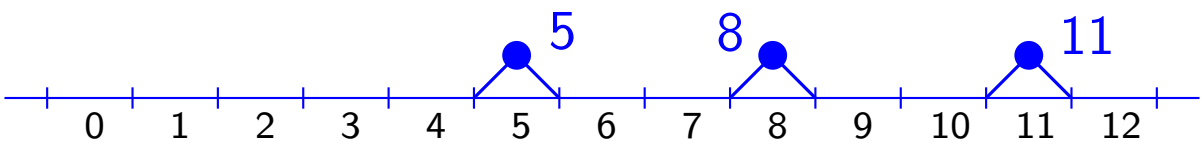
$Q(W)$



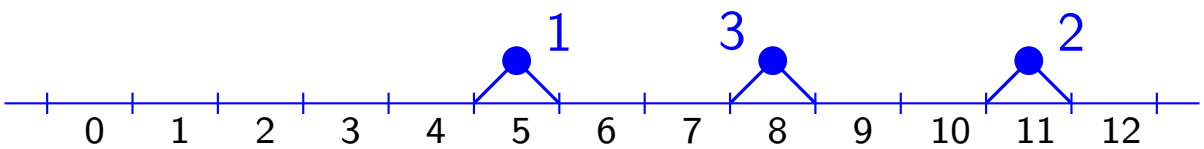
Encoding as binary forests

Definition (by example). $W = a_1a_2a_3a_4a_5a_6a_7 = 5.11.8.3.8.4.3$

$P(W)$



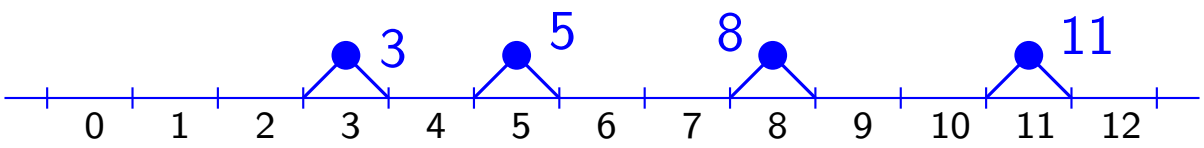
$Q(W)$



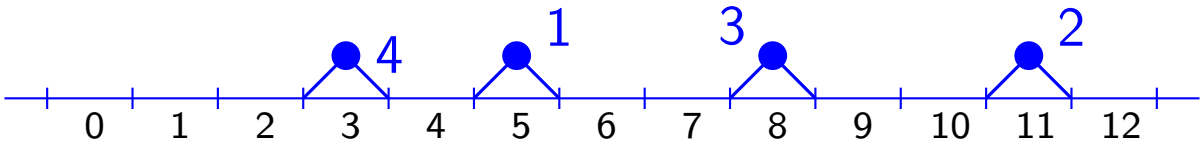
Encoding as binary forests

Definition (by example). $W = a_1a_2a_3a_4a_5a_6a_7 = 5.11.8.3.8.4.3$

$P(W)$



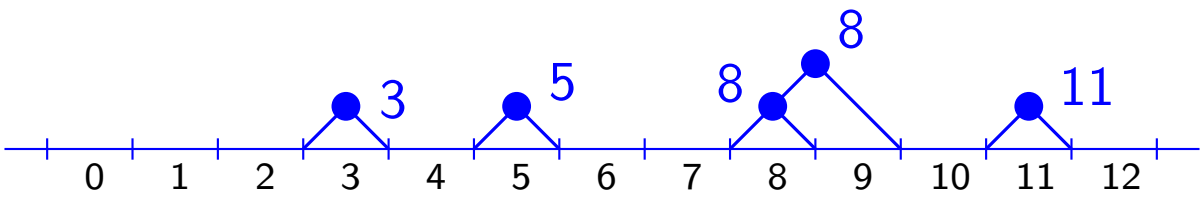
$Q(W)$



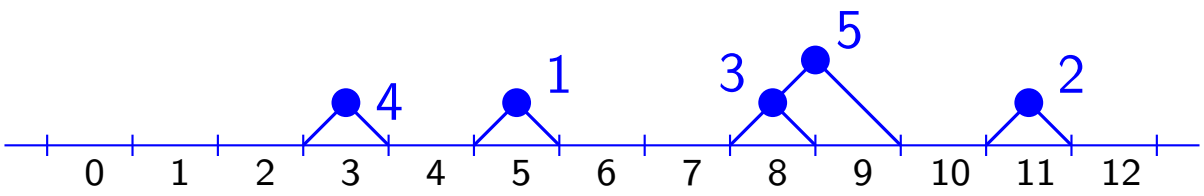
Encoding as binary forests

Definition (by example). $W = a_1a_2a_3a_4a_5a_6a_7 = 5.11.8.3.8.4.3$

$P(W)$



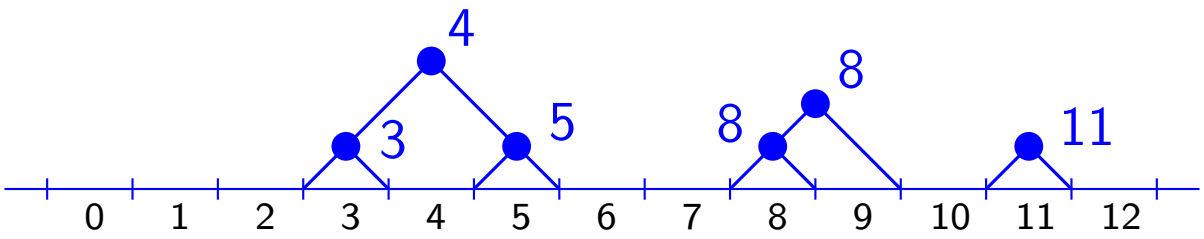
$Q(W)$



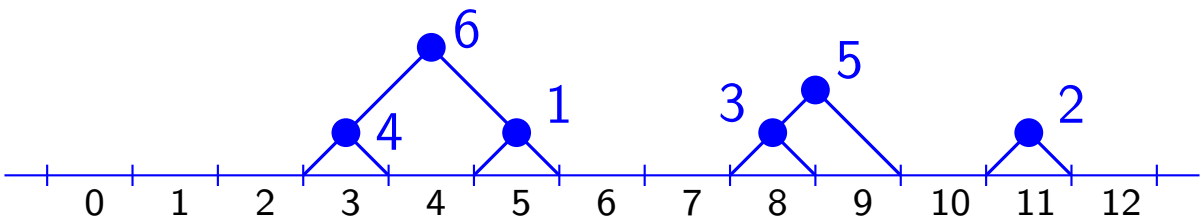
Encoding as binary forests

Definition (by example). $W = a_1a_2a_3a_4a_5a_6a_7 = 5.11.8.3.8.4.3$

$P(W)$



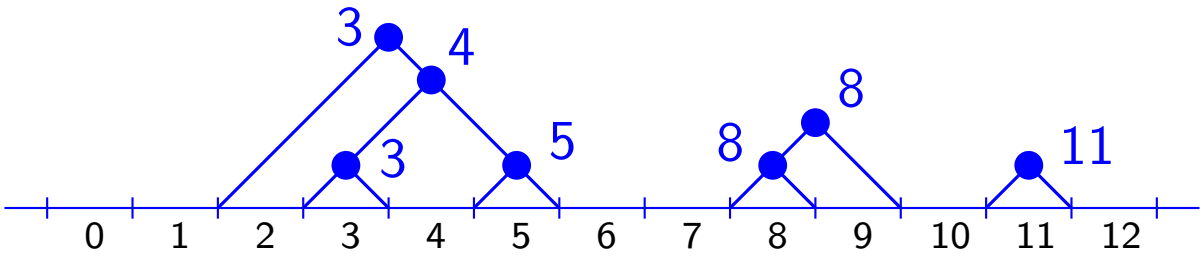
$Q(W)$



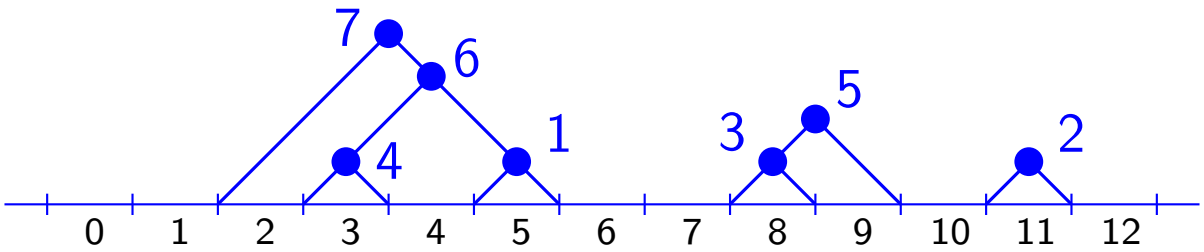
Encoding as binary forests

Definition (by example). $W = a_1a_2a_3a_4a_5a_6a_7 = 5.11.8.3.8.4.3$

$P(W)$



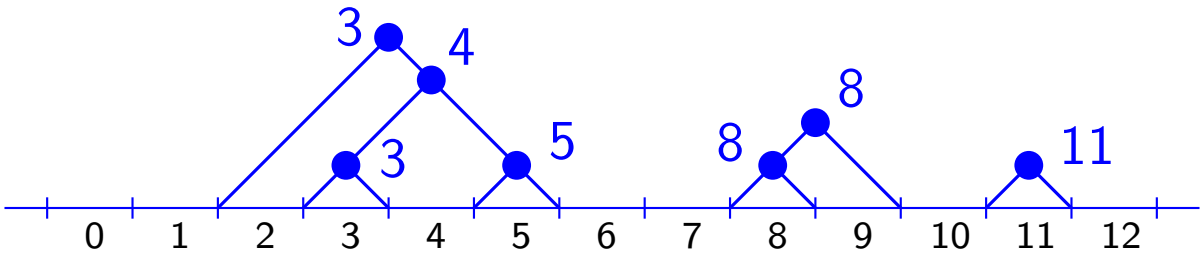
$Q(W)$



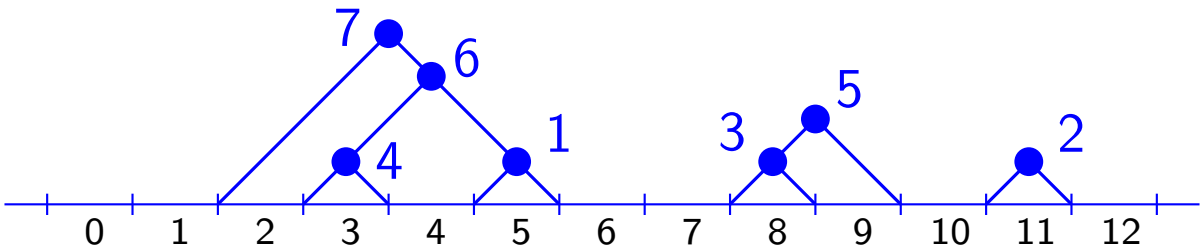
Encoding as binary forests

Definition (by example). $W = a_1a_2a_3a_4a_5a_6a_7 = 5.11.8.3.8.4.3$

$P(W)$



$Q(W)$



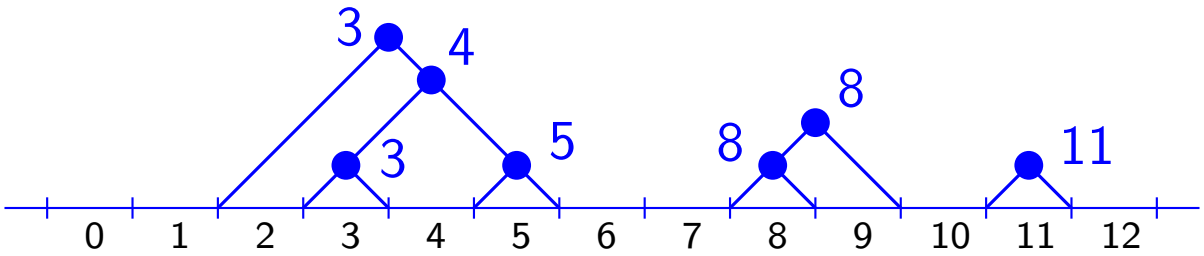
Definition of $\widehat{\mathcal{P}}$. The forest F is constructed inductively: if the actual parking spot is j, create a root node with canonical label j, and attach subtrees if necessary.
If the desired spot was a_i, the label in P is a_i and the label in Q is i.

$$\widehat{\mathcal{P}}(W) = (P(W), Q(W)) \in \bigcup_{F \text{ forest}} \mathcal{F}(F) \times \mathrm{Dec}(F).$$

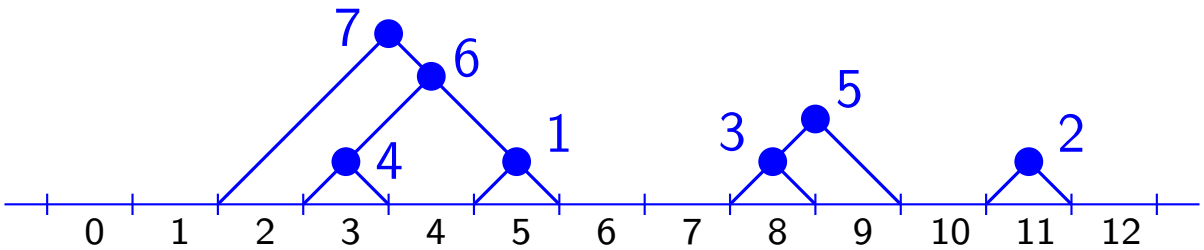
Encoding as binary forests

Definition (by example). $W = a_1a_2a_3a_4a_5a_6a_7 = 5.11.8.3.8.4.3$

$P(W)$



$Q(W)$



Definition of $\widehat{\mathcal{P}}$. The forest F is constructed inductively: if the actual parking spot is j, create a root node with canonical label j, and attach subtrees if necessary.
If the desired spot was a_i, the label in P is a_i and the label in Q is i.

$$\widehat{\mathcal{P}}(W) = (P(W), Q(W)) \in \bigcup_{F \text{ forest}} \mathcal{F}(F) \times \mathrm{Dec}(F).$$

Some facts.

- $\widehat{\mathcal{P}}$ is injective, and $\mathcal{P}(W)$ is the support of the forest F in $\widehat{\mathcal{P}}(W)$.
- If the letters of W are distinct, one has essentially the *Sylvester correspondence*.
- But the image cannot be "nice" for general $\mathcal{P}$...

Local parking procedures

We add two more “axioms” for a parking procedure $\mathcal{P} : \mathbb{Z}^* \to \text{Finset}(\mathbb{Z})$.

Local parking procedures

We add two more "axioms" for a parking procedure $\mathcal{P} : \mathbb{Z}^* \to \text{Finset}(\mathbb{Z})$.

- Let τ be the shift $i \mapsto i + 1$ on $\mathbb{Z}$, extended to $\mathbb{Z}^*$ and $\text{Finset}(\mathbb{Z})$.

Local parking procedures

We add two more "axioms" for a parking procedure $\mathcal{P} : \mathbb{Z}^* \to \text{Finset}(\mathbb{Z})$.

- Let τ be the shift $i \mapsto i + 1$ on $\mathbb{Z}$, extended to $\mathbb{Z}^*$ and $\text{Finset}(\mathbb{Z})$.

(Shift invariance) For any W, $\mathcal{P}(\tau(W)) = \tau(\mathcal{P}(W))$.

Local parking procedures

We add two more "axioms" for a parking procedure $\mathcal{P} : \mathbb{Z}^* \to \text{Finset}(\mathbb{Z})$.

- Let τ be the shift $i \mapsto i + 1$ on $\mathbb{Z}$, extended to $\mathbb{Z}^*$ and $\text{Finset}(\mathbb{Z})$.

(Shift invariance) For any W, $\mathcal{P}(\tau(W)) = \tau(\mathcal{P}(W))$.

- For any word W and any subset $I \subset \mathcal{P}(W)$, define $W_{|I}$ as the subword corresponding to the cars that parked in I.

Local parking procedures

We add two more "axioms" for a parking procedure $\mathcal{P} : \mathbb{Z}^* \to \text{Finset}(\mathbb{Z})$.

- Let τ be the shift $i \mapsto i + 1$ on $\mathbb{Z}$, extended to $\mathbb{Z}^*$ and $\text{Finset}(\mathbb{Z})$.

(Shift invariance) For any W, $\mathcal{P}(\tau(W)) = \tau(\mathcal{P}(W))$.

- For any word W and any subset $I \subset \mathcal{P}(W)$, define $W_{|I}$ as the subword corresponding to the cars that parked in I.

(Local decision) Let W, a, I such that I is an interval of $\mathcal{P}(W)$ and $a \in I$. Then $\text{lastSpot}_{\mathcal{P}}(Wa) = \text{lastSpot}_{\mathcal{P}}(W_{|I}a)$.

Local parking procedures

We add two more "axioms" for a parking procedure $\mathcal{P} : \mathbb{Z}^* \to \text{Finset}(\mathbb{Z})$.

- Let τ be the shift $i \mapsto i+1$ on $\mathbb{Z}$, extended to $\mathbb{Z}^*$ and $\text{Finset}(\mathbb{Z})$.

(Shift invariance) For any W, $\mathcal{P}(\tau(W)) = \tau(\mathcal{P}(W))$.

- For any word W and any subset $I \subset \mathcal{P}(W)$, define $W_{|I}$ as the subword corresponding to the cars that parked in I.

(Local decision) Let W, a, I such that I is an interval of $\mathcal{P}(W)$ and $a \in I$. Then $\text{lastSpot}_{\mathcal{P}}(Wa) = \text{lastSpot}_{\mathcal{P}}(W_{|I}a)$.

Definition. A parking procedure $\mathcal{P}$ is called local if it satisfies the extra axioms (Shift invariance) and (Local decision).

Remark. When $\mathcal{P}$ is local, it is entirely characterized by the data of $\mathcal{P}(Wa)$ when $\mathcal{P}(W) = \{1, 2, \dots, r\}$, i.e. W is $\mathcal{P}$-parking.

Local parking procedures

We add two more "axioms" for a parking procedure $\mathcal{P} : \mathbb{Z}^* \to \text{Finset}(\mathbb{Z})$.

- Let τ be the shift $i \mapsto i+1$ on $\mathbb{Z}$, extended to $\mathbb{Z}^*$ and $\text{Finset}(\mathbb{Z})$.

(Shift invariance) For any W, $\mathcal{P}(\tau(W)) = \tau(\mathcal{P}(W))$.

- For any word W and any subset $I \subset \mathcal{P}(W)$, define $W_{|I}$ as the subword corresponding to the cars that parked in I.

(Local decision) Let W, a, I such that I is an interval of $\mathcal{P}(W)$ and $a \in I$. Then $\text{lastSpot}_{\mathcal{P}}(Wa) = \text{lastSpot}_{\mathcal{P}}(W_{|I}a)$.

Definition. A parking procedure $\mathcal{P}$ is called local if it satisfies the extra axioms (Shift invariance) and (Local decision).

Remark. When $\mathcal{P}$ is local, it is entirely characterized by the data of $\mathcal{P}(Wa)$ when $\mathcal{P}(W) = \{1, 2, \dots, r\}$, i.e. W is $\mathcal{P}$-parking.

Proposition. If $\mathcal{P}$ is local, there is a class of labeled forests $\mathcal{F}_{\mathcal{P}}(F) \subset \mathcal{F}(F)$ such that $\widehat{\mathcal{P}} : \mathbb{Z}^* \to \bigcup_{F \text{ forest}} \mathcal{F}_{\mathcal{P}}(F) \times \text{Dec}(F)$ is a bijection.

Enumeration

Theorem ([N. '22+]). For any local parking procedure $\mathcal{P}$, the number of $\mathcal{P}$-parking words of size r is $(r+1)^{r-1}$.

Enumeration

Theorem ([N. '22+]). For any local parking procedure $\mathcal{P}$, the number of $\mathcal{P}$-parking words of size r is $(r+1)^{r-1}$.

Proof sketch: We extend Pollak's argument to the local case.

Idea: project $\mathcal{P}$ to a cyclic parking procedure $\mathcal{P}^{r+1}$ on $\mathbb{Z}_{r+1} = \mathbb{Z}/(r+1)\mathbb{Z}$.

Enumeration

Theorem ([N. '22+]). For any local parking procedure $\mathcal{P}$, the number of $\mathcal{P}$-parking words of size r is $(r+1)^{r-1}$.

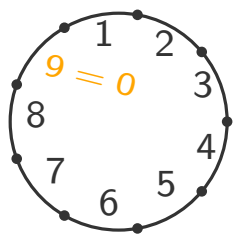
Proof sketch: We extend Pollak's argument to the local case.

Idea: project $\mathcal{P}$ to a cyclic parking procedure $\mathcal{P}^{r+1}$ on $\mathbb{Z}_{r+1} = \mathbb{Z}/(r+1)\mathbb{Z}$.

Definition ($\mathcal{P}^{r+1} : \mathbb{Z}_{r+1}^{\leq r} \to$ Subsets of $\mathbb{Z}_{r+1}$).

Let J be a *cyclic interval* of $\mathcal{P}^{r+1}(w)$, and $\bar{a} \in J$.

- "Lift" $J, \bar{a}, w$ to $\{1, \dots, |J|\}, a, W$ in $\mathbb{Z}$.
- Pick left or right for $\mathcal{P}^{r+1}(wa)$ as in $\mathcal{P}(Wa)$.



$r = 8$

Enumeration

Theorem ([N. '22+]). For any local parking procedure $\mathcal{P}$, the number of $\mathcal{P}$-parking words of size r is $(r+1)^{r-1}$.

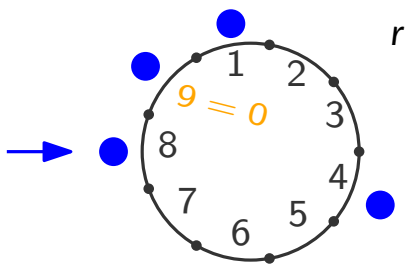
Proof sketch: We extend Pollak's argument to the local case.

Idea: project $\mathcal{P}$ to a cyclic parking procedure $\mathcal{P}^{r+1}$ on $\mathbb{Z}_{r+1} = \mathbb{Z}/(r+1)\mathbb{Z}$.

Definition ($\mathcal{P}^{r+1} : \mathbb{Z}_{r+1}^{\leq r} \to$ Subsets of $\mathbb{Z}_{r+1}$).

Let J be a *cyclic interval* of $\mathcal{P}^{r+1}(w)$, and $\bar{a} \in J$.

- "Lift" $J, \bar{a}, w$ to $\{1, \dots, |J|\}, a, W$ in $\mathbb{Z}$.
- Pick left or right for $\mathcal{P}^{r+1}(wa)$ as in $\mathcal{P}(Wa)$.



$r = 8$

Example. Suppose $w = \overline{1941}$ has $\mathcal{P}^9(w) = \{\bar{8}, \bar{9} = \bar{0}, \bar{1}\} \sqcup \{\bar{4}\} \subset \mathbb{Z}_9$

Pick $\bar{a} = \bar{8}$. We lift $\{\bar{8}, \bar{9} = \bar{0}, \bar{1}\}$ to $\{1, 2, 3\}$, $\bar{8}$ to 1, and $\overline{1941}$ to $\overline{3136}$.

Then $\mathcal{P}^9(\overline{1941}.\bar{8})$ is determined by $\mathcal{P}(3136.1)$.

Enumeration

Theorem ([N. '22+]). For any local parking procedure $\mathcal{P}$, the number of $\mathcal{P}$-parking words of size r is $(r+1)^{r-1}$.

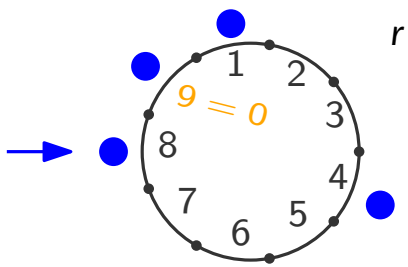
Proof sketch: We extend Pollak's argument to the local case.

Idea: project $\mathcal{P}$ to a cyclic parking procedure $\mathcal{P}^{r+1}$ on $\mathbb{Z}_{r+1} = \mathbb{Z}/(r+1)\mathbb{Z}$.

Definition ($\mathcal{P}^{r+1} : \mathbb{Z}_{r+1}^{\leq r} \to$ Subsets of $\mathbb{Z}_{r+1}$).

Let J be a *cyclic interval* of $\mathcal{P}^{r+1}(w)$, and $\bar{a} \in J$.

- "Lift" $J, \bar{a}, w$ to $\{1, \dots, |J|\}, a, W$ in $\mathbb{Z}$.
- Pick left or right for $\mathcal{P}^{r+1}(wa)$ as in $\mathcal{P}(Wa)$.



Example. Suppose $w = \overline{1941}$ has $\mathcal{P}^9(w) = \{\bar{8}, \bar{9} = \bar{0}, \bar{1}\} \sqcup \{\bar{4}\} \subset \mathbb{Z}_9$

Pick $\bar{a} = \bar{8}$. We lift $\{\bar{8}, \bar{9} = \bar{0}, \bar{1}\}$ to $\{1, 2, 3\}$, $\bar{8}$ to 1, and $\overline{1941}$ to $\overline{3136}$.

Then $\mathcal{P}^9(\overline{1941}.\bar{8})$ is determined by $\mathcal{P}(3136.1)$.

The theorem then follows from the following lemmas:

Lemma (1). Let $W \in \{1, \dots, r+1\}^r$. Then
W is a $\mathcal{P}$-parking function $\Leftrightarrow \mathcal{P}^{r+1}(W \mod (r+1))$ has empty spot $\bar{0}$.

Lemma (2). Let $w \in \mathbb{Z}_{r+1}^r$ have empty spot k.
Then $\bar{\tau}^i(w) \in \mathbb{Z}_{r+1}^r$ has empty spot $k+i$ for $i = 0, 1, \dots, r$.

□

Extension 1: Probabilization

The classical parking procedure and functions naturally lead to probabilistic questions: see Konheim–Weiss, Flajolet–Poblete–Viola, Diaconis–Yan, …

Extension 1: Probabilization

The classical parking procedure and functions naturally lead to probabilistic questions: see Konheim–Weiss, Flajolet–Poblete–Viola, Diaconis–Yan, ...

We can probabilize the parking procedures themselves:

- Constructed by a sequence of "Bernoulli" laws on *{Left, Right}*.
- $\mathcal{P}(W)$ is then a finite probability measure on $\mathsf{Finset}(\mathbb{Z})$.

Extension 1: Probabilization

The classical parking procedure and functions naturally lead to probabilistic questions: see Konheim–Weiss, Flajolet–Poblete–Viola, Diaconis–Yan, ...

We can probabilize the parking procedures themselves:

- Constructed by a sequence of "Bernoulli" laws on *{Left, Right}*.
- $\mathcal{P}(W)$ is then a finite probability measure on Finset($\mathbb{Z}$).

Example (1). Fix a parameter $p \in [0, 1]$, and decide to go Right with probability p and Left with probability $1 - p$.

Extension 1: Probabilization

The classical parking procedure and functions naturally lead to probabilistic questions: see Konheim–Weiss, Flajolet–Poblete–Viola, Diaconis–Yan, ...

We can probabilize the parking procedures themselves:

- Constructed by a sequence of "Bernoulli" laws on $\{Left, Right\}$.
- $\mathcal{P}(W)$ is then a finite probability measure on $\mathrm{Finset}(\mathbb{Z})$.

Example (1). Fix a parameter $p \in [0, 1]$, and decide to go Right with probability p and Left with probability $1 - p$.

The results for the deterministic case have natural extensions to this setting:

1. Encoding by binary forests.
2. Local procedures.
3. Enumeration via Pollak's argument.

Extension 1: Probabilization

The classical parking procedure and functions naturally lead to probabilistic questions: see Konheim–Weiss, Flajolet–Poblete–Viola, Diaconis–Yan, ...

We can probabilize the parking procedures themselves:

- Constructed by a sequence of "Bernoulli" laws on $\{Left, Right\}$.
- $\mathcal{P}(W)$ is then a finite probability measure on $\mathrm{Finset}(\mathbb{Z})$.

Example (1). Fix a parameter $p \in [0,1]$, and decide to go Right with probability p and Left with probability $1-p$.

The results for the deterministic case have natural extensions to this setting:

1. Encoding by binary forests.
2. Local procedures.
3. Enumeration via Pollak's argument.

Example (2). Fix a parameter $q \in [0,\infty]$. Suppose one wants to park at $i \in [1,n]$. Then go Right with probability $\frac{1+q+\cdots+q^{i-1}}{1+q+\cdots+q^{n-1}}$ (and thus Left with probability $\frac{q^i+\cdots+q^{n-1}}{1+q+\cdots+q^{n-1}}$). This determines a local procedure that is abelian.

This last procedure can be seen a series of random walks, and has many nice properties. (cf. [Diaconis-Fulton, Tewari-N.])

Extension 2: Extra information

Let S be any set, and consider the alphabet $A = \mathbb{Z} \times S$. The set S represents some extra information: in terms of cars, one might consider its brand or color.

Extension 2: Extra information

Let S be any set, and consider the alphabet $A = \mathbb{Z} \times S$. The set S represents some extra information: in terms of cars, one might consider its brand or color.

All the constructions generalize easily to words in A^*:

- parking procedures.
- binary forest encoding (label the first tree with A).
- local procedures.
- Pollak's argument.

Extension 2: Extra information

Let S be any set, and consider the alphabet $A = \mathbb{Z} \times S$. The set S represents some extra information: in terms of cars, one might consider its brand or color.

All the constructions generalize easily to words in A^*:

- parking procedures.
- binary forest encoding (label the first tree with A).
- local procedures.
- Pollak's argument.

One can also restrict the set of words to $L \subset A^*$: that is, the procedure need not be defined everywhere.

Extension 2: Extra information

Let S be any set, and consider the alphabet $A = \mathbb{Z} \times S$. The set S represents some extra information: in terms of cars, one might consider its brand or color.

All the constructions generalize easily to words in A^*:

- parking procedures.
- binary forest encoding (label the first tree with A).
- local procedures.
- Pollak's argument.

One can also restrict the set of words to $L \subset A^*$: that is, the procedure need not be defined everywhere.

Example. The procedure $\mathcal{P}_{CS}$ is extended to words with $S = \{1, 2, ...\}$ and distinct letters. The comparison $a < a_j$ uses then the lexicographic order on $A = \mathbb{Z} \times S$.

This last example came up naturally in joint work with Vasu Tewari, and was the starting point of the current project.

Extension 2: Extra information

Let S be any set, and consider the alphabet $A = \mathbb{Z} \times S$. The set S represents some extra information: in terms of cars, one might consider its brand or color.

All the constructions generalize easily to words in A^*:

- parking procedures.
- binary forest encoding (label the first tree with A).
- local procedures.
- Pollak's argument.

One can also restrict the set of words to $L \subset A^*$: that is, the procedure need not be defined everywhere.

Example. The procedure $\mathcal{P}_{CS}$ is extended to words with $S = \{1, 2, ...\}$ and distinct letters. The comparison $a < a_j$ uses then the lexicographic order on $A = \mathbb{Z} \times S$.

This last example came up naturally in joint work with Vasu Tewari, and was the starting point of the current project.

MERCI DE VOTRE ATTENTION