Lowerbounds for the Online Minimum Matching Problem on the Line

Maximillian C.W. Bender

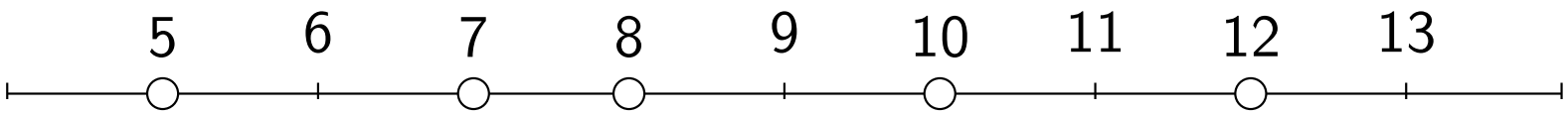
Research Advisor: Christine Chung

CONNECTICUT COLLEGE

An Example

Ski Size	Renter's Size	Difference
5		
7		
8		
10		
12		

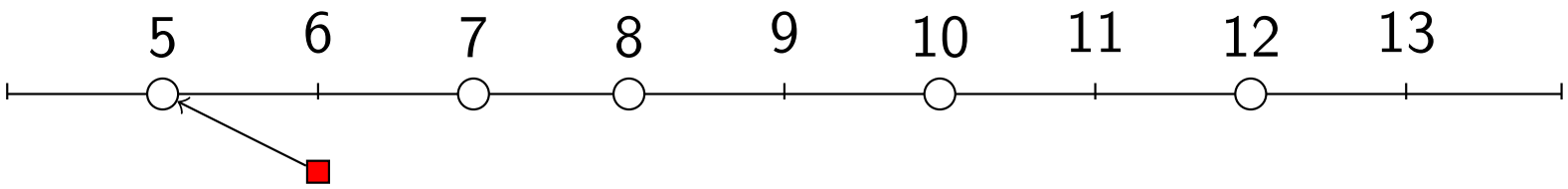
Renter's Size Sequence:



An Example

Ski Size	Renter's Size	Difference
5	6	1
7		
8		
10		
12		
		Sum: 1
		Max: 1

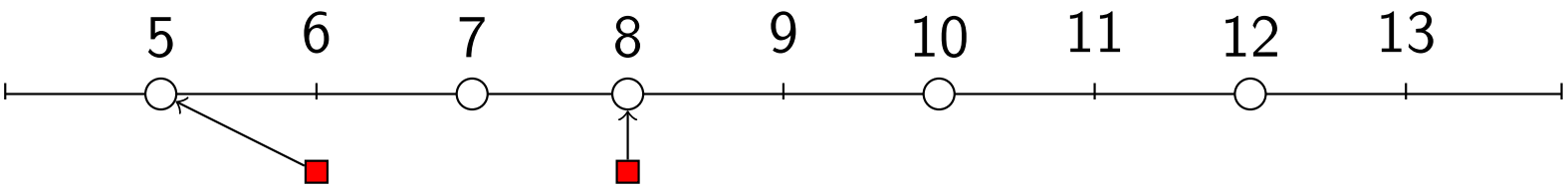
Renter's Size Sequence: (6,



An Example

Ski Size	Renter's Size	Difference
5	6	1
7		
8	8	0
10		
12		
		Sum: 1
		Max: 1

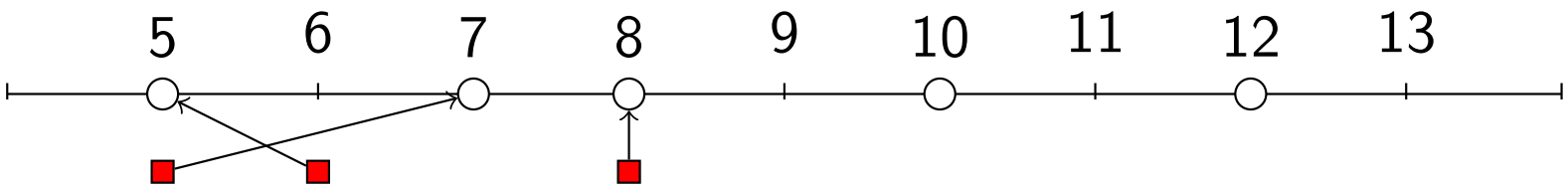
Renter's Size Sequence: $(6, 8,$



An Example

Ski Size	Renter's Size	Difference
5	6	1
7	5	2
8	8	0
10		
12		
		Sum: 3
		Max: 2

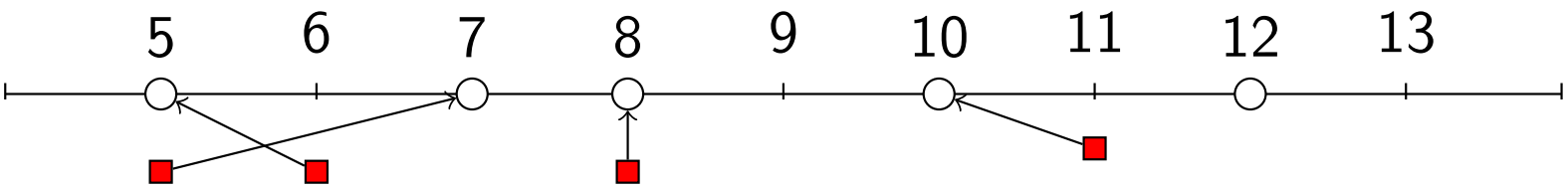
Renter's Size Sequence: $(6, 8, 5,$



An Example

Ski Size	Renter's Size	Difference
5	6	1
7	5	2
8	8	0
10	11	1
12		
		Sum: 4
		Max: 2

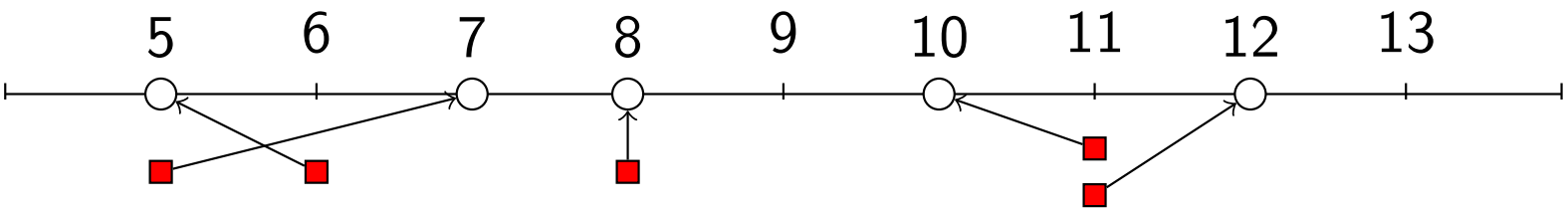
Renter's Size Sequence: $(6, 8, 5, 11,$



An Example

Ski Size	Renter's Size	Difference
5	6	1
7	5	2
8	8	0
10	11	1
12	11	1
		Sum: 5
		Max: 2

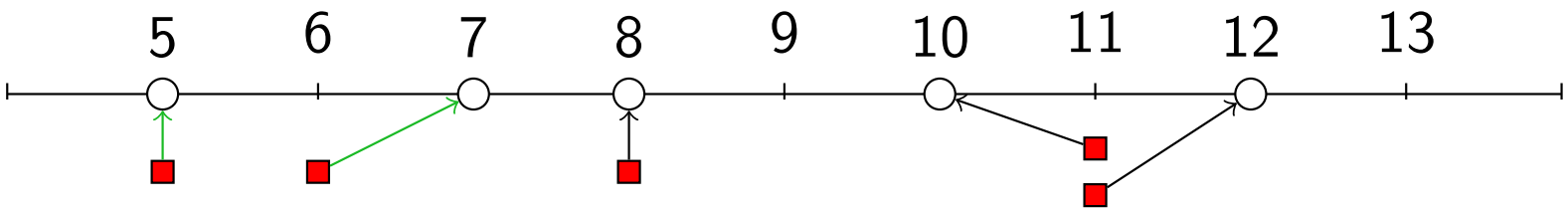
Renter's Size Sequence: $(6, 8, 5, 11, 11)$



An Example

Ski Size	Renter's Size	Difference
5	5	0
7	6	1
8	8	0
10	11	1
12	11	1
		Sum: 3
		Max: 1

Renter's Size Sequence: $(6, 8, 5, 11, 11)$



Questions for the Online Matching Model

1. What is the best general approach to minimizing the total or maximal difference between the sizes? Can we establish an **upperbound** to the approach's cost?

2. What is the best we could hope to achieve with any algorithm to minimize the total or maximal difference? Can we fix a **lowerbound** for any algorithm's performance?

Formalism

Problem Definition

- Input:
  - A set $S \subset \mathbb{R}$ of n servers
  - A sequence of requests $R = (r_1, r_2, ..., r_n) \subset \mathbb{R}$ whose locations are unknown prior to arrival.

- Input:
  - A set $S \subset \mathbb{R}$ of n servers
  - A sequence of requests $R = (r_1, r_2, ..., r_n) \subset \mathbb{R}$ whose locations are unknown prior to arrival.

- Output: A bijective function $f : R \to S$.
  - We denote the "Minimum-Weight" or "Total" cost of any function as

$$c(f) = \sum_{i=1}^{n} |f(r_i) - r_i|.$$

  - The "Bottleneck Cost" is defined as

$$c(f) = \max_{i \in \{1,...,n\}} |f(r_i) - r_i|.$$

Goal

The goal is to construct a bijective function $f : R \to S$ that best approximates the objective function:

$$f = \underset{g:R \rightarrowtail S}{\operatorname{argmin}} \left(c(g)\right)$$

Definition (Deterministic vs. Randomized)

We denote the set of deterministic algorithms for this problem $\mathcal{A}_{\mathcal{D}}$. We denote the set of randomized algorithms $\mathcal{A}$, where $\mathcal{A}_{\mathcal{D}} \subset \mathcal{A}$.

Definition (Competitive Ratio for Deterministic Algorithms)

We define the *competitive ratio* ρ of any *deterministic* algorithm $A \in \mathcal{A}_{\mathcal{D}}$ by

$$\rho = \max_{I \in \mathcal{I}} \left(\frac{c\left(A\left(I\right)\right)}{c\left(\mathrm{OPT}(I)\right)} \right)$$

where $\mathcal{I}$ denotes the set of all possible instances for the problem.

Definition (Competitive Ratio for Randomized Algorithms)

We define the *competitive ratio* ρ of any *randomized* algorithm $A \in \mathcal{A}$ by

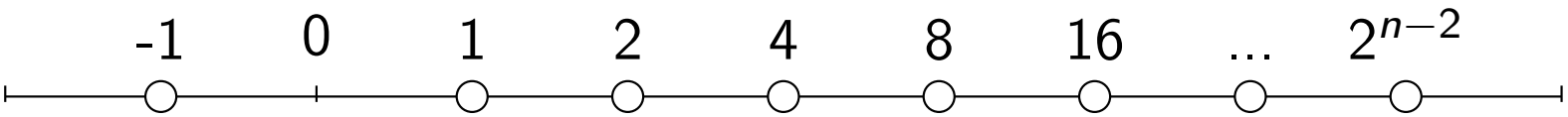
$$\rho = \max_{I \in \mathcal{I}} \left(\mathrm{E}\left[\frac{c\left(A\left(I\right)\right)}{c\left(\mathrm{OPT}(I)\right)} \right] \right)$$

where $\mathcal{I}$ denotes the set of all possible instances for the problem.

Lowerbound of Greedy

We will define the deterministic Greedy algorithm as the algorithm which sends each request to its closest available server at the time of arrival.

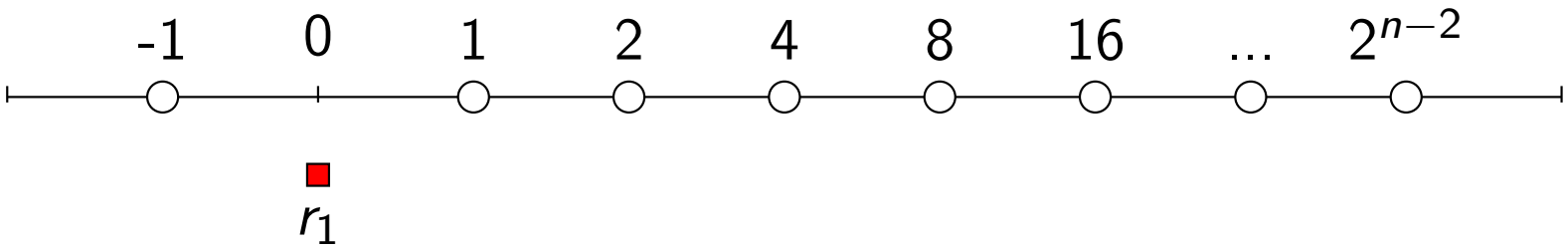
Consider the following instance I with n servers:



Lowerbound of Greedy

We will define the deterministic Greedy algorithm as the algorithm which sends each request to its closest available server at the time of arrival.

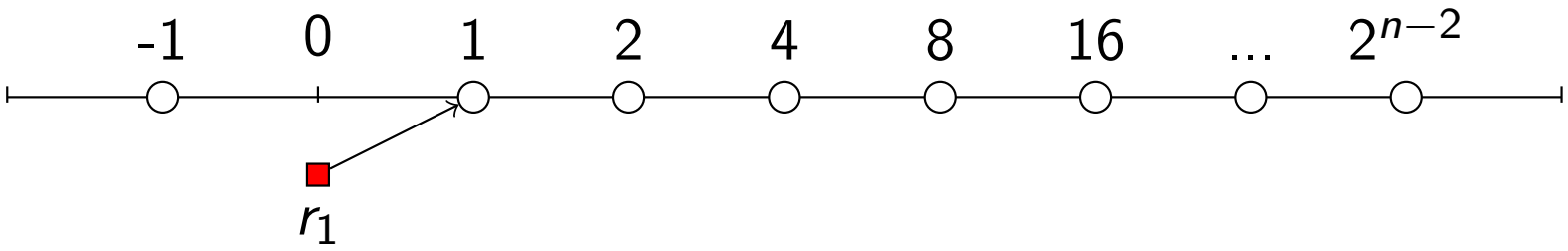
Consider the following instance I with n servers:



Lowerbound of GREEDY

We will define the deterministic GREEDY algorithm as the algorithm which sends each request to its closest available server at the time of arrival.

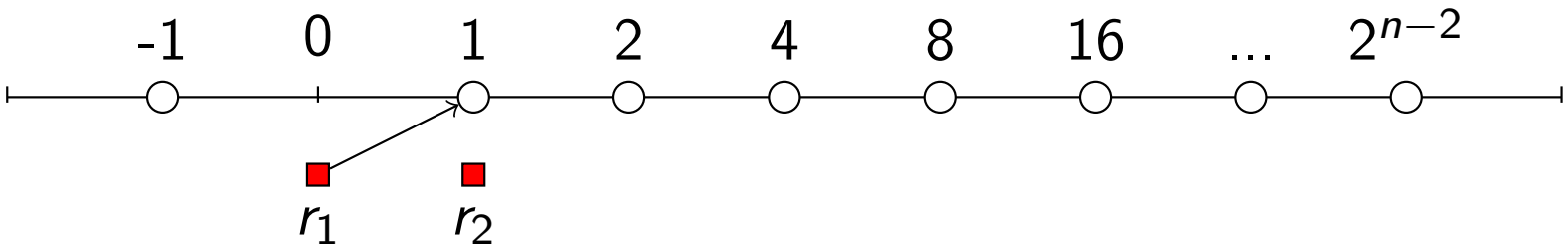
Consider the following instance I with n servers:



Lowerbound of Greedy

We will define the deterministic Greedy algorithm as the algorithm which sends each request to its closest available server at the time of arrival.

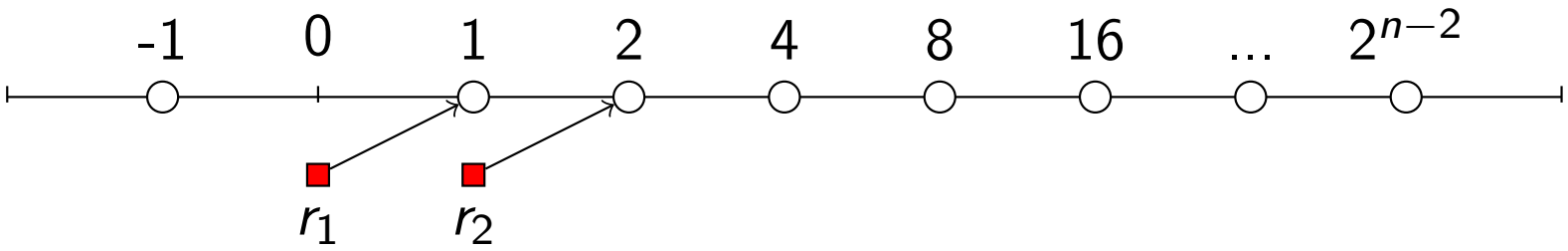
Consider the following instance I with n servers:



Lowerbound of GREEDY

We will define the deterministic GREEDY algorithm as the algorithm which sends each request to its closest available server at the time of arrival.

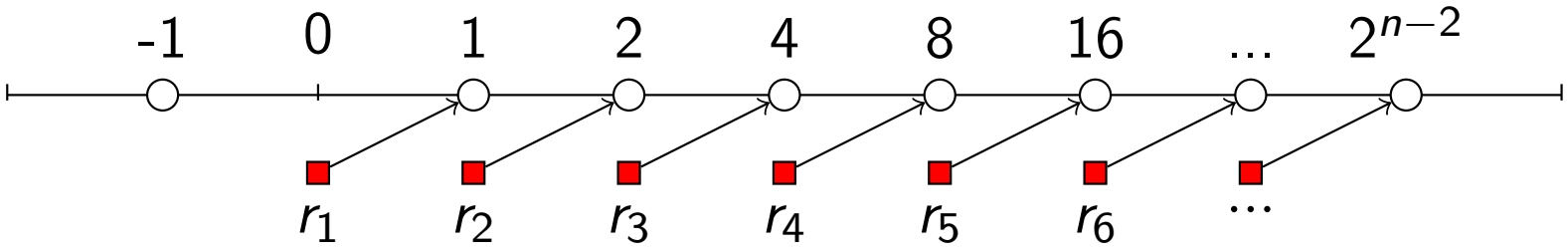
Consider the following instance I with n servers:



Lowerbound of GREEDY

We will define the deterministic GREEDY algorithm as the algorithm which sends each request to its closest available server at the time of arrival.

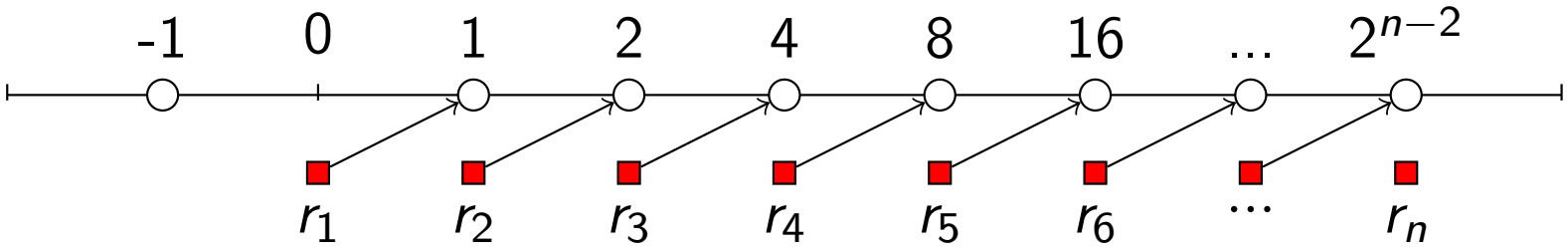
Consider the following instance I with n servers:



Lowerbound of Greedy

We will define the deterministic Greedy algorithm as the algorithm which sends each request to its closest available server at the time of arrival.

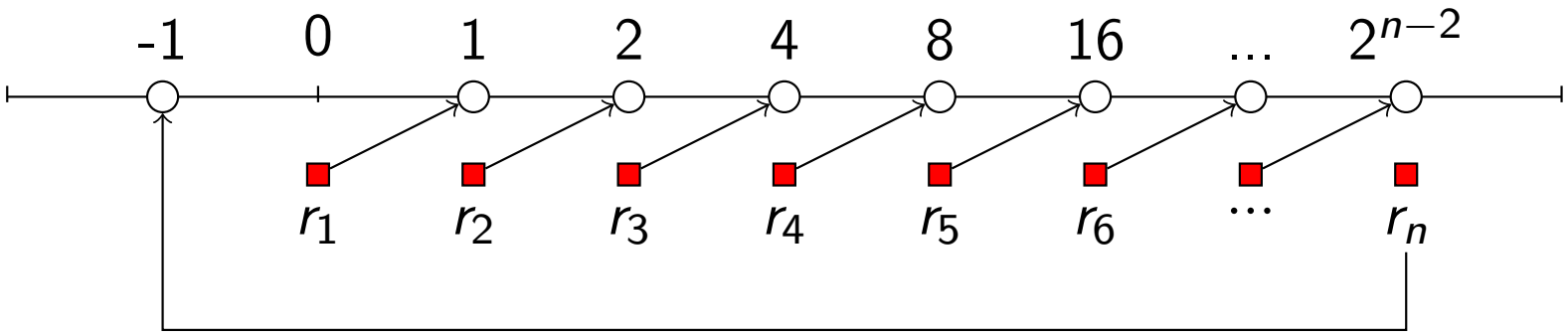
Consider the following instance I with n servers:



Lowerbound of GREEDY

We will define the deterministic GREEDY algorithm as the algorithm which sends each request to its closest available server at the time of arrival.

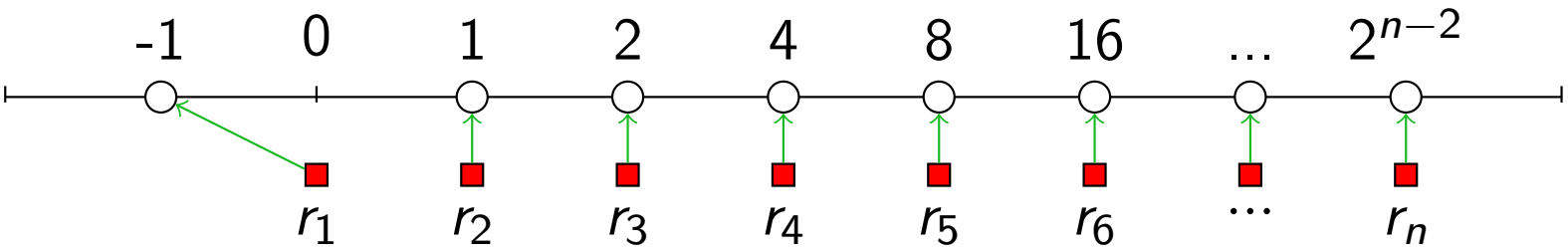
Consider the following instance I with n servers:



Lowerbound of GREEDY

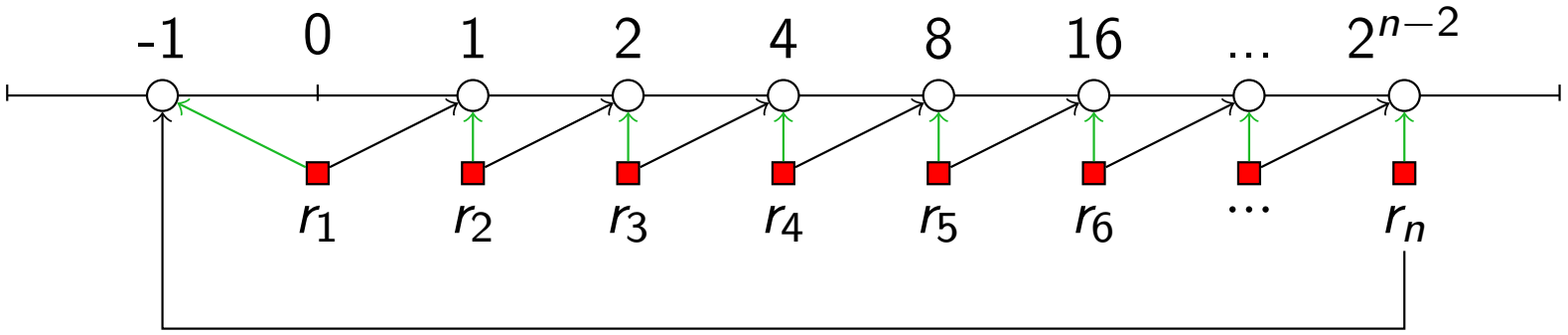
We will define the deterministic GREEDY algorithm as the algorithm which sends each request to its closest available server at the time of arrival.

Consider the following instance I with n servers:



Lowerbound of Greedy

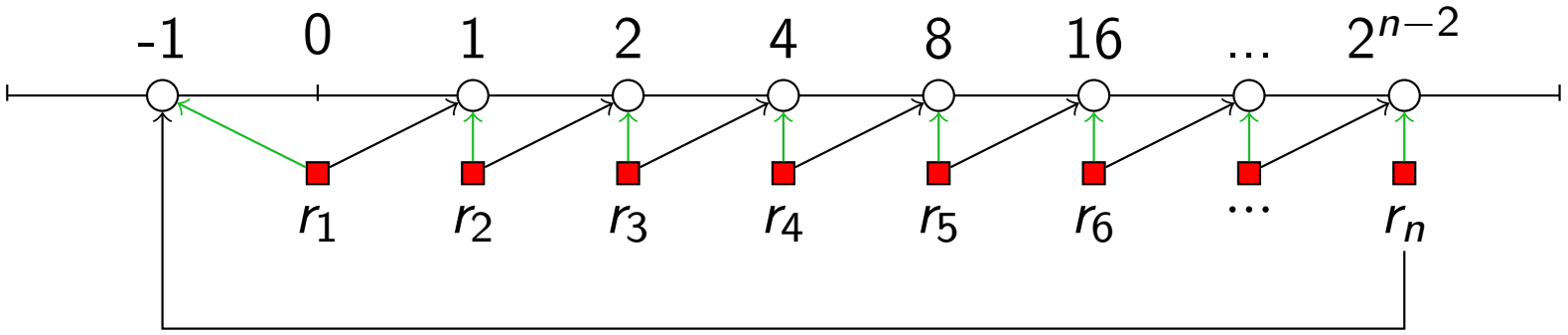
Min-Weight



$$c\left(\textsc{Greedy}\left(I\right)\right) = 1 + \sum_{i=1}^{n-2}\left(2^i - 2^{i-1}\right) + 2^{n-2} + 1$$
$$= 1 + 2 \times 2^{n-2} = 1 + 2^{n-1}$$

Lowerbound of Greedy

Min-Weight



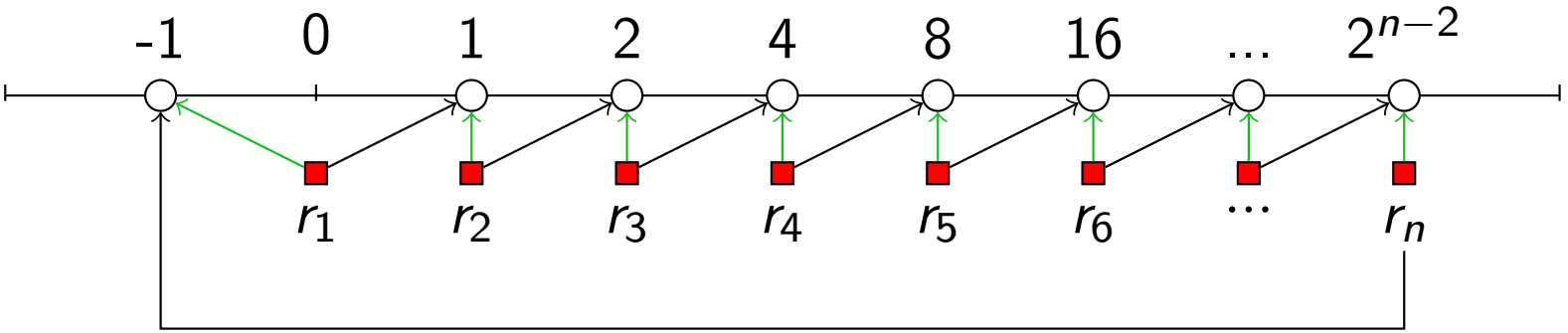
$$c\left(\textsc{Greedy}\left(I\right)\right) = 1 + \sum_{i=1}^{n-2}\left(2^i - 2^{i-1}\right) + 2^{n-2} + 1$$

$$= 1 + 2 \times 2^{n-2} = 1 + 2^{n-1}$$

$$c\left(\textsc{Opt}\left(I\right)\right) = 1$$

Lowerbound of Greedy

Min-Weight



$$c\left(\textsc{Greedy}\left(I\right)\right) = 1 + \sum_{i=1}^{n-2}\left(2^i - 2^{i-1}\right) + 2^{n-2} + 1$$

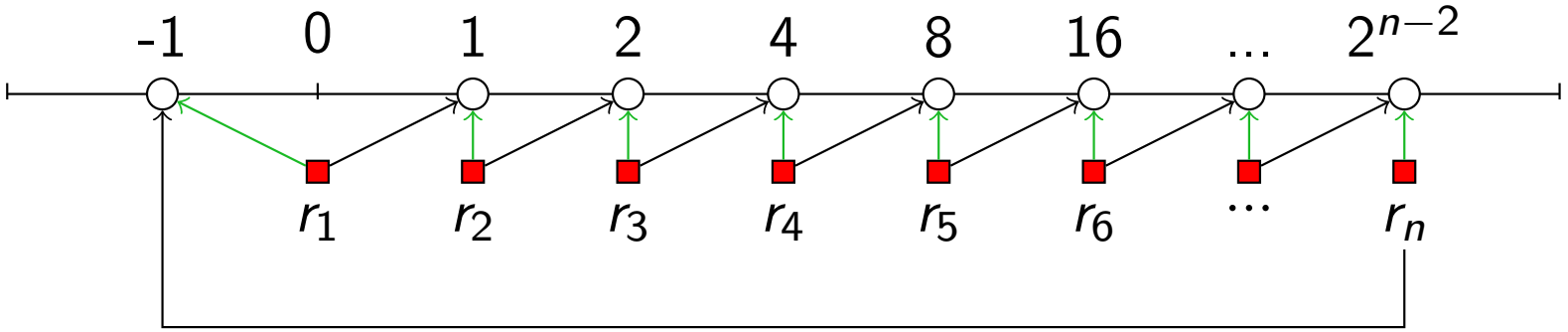
$$= 1 + 2 \times 2^{n-2} = 1 + 2^{n-1}$$

$$c\left(\textsc{Opt}\left(I\right)\right) = 1$$

$$\rho = \max_{I \in \mathcal{I}} \left(\frac{c\left(A\left(I\right)\right)}{c\left(\textsc{Opt}(I)\right)} \right) \geq 2^{n-1}$$

Lowerbound of Greedy

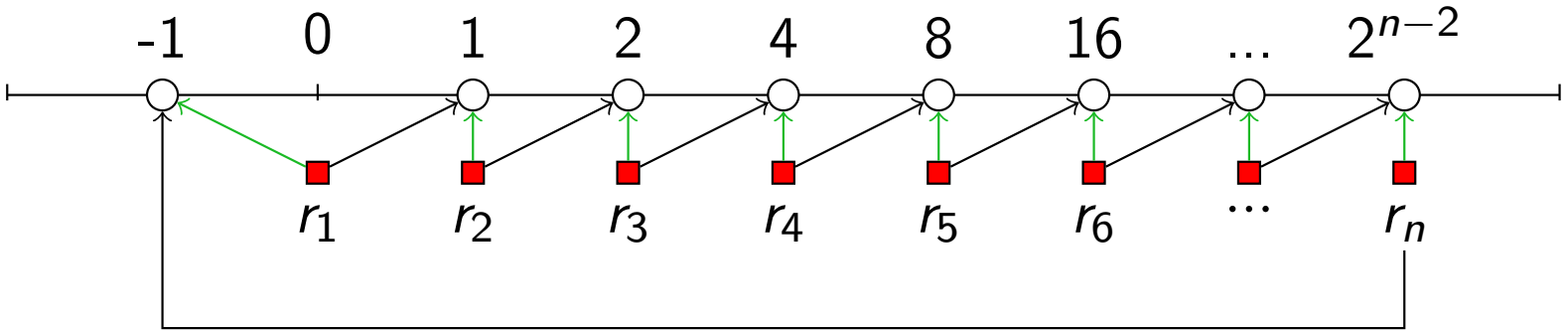
Bottleneck



$$c\left(\textsc{Greedy}\left(I\right)\right) = 2^{n-2} + 1$$

Lowerbound of Greedy

Bottleneck

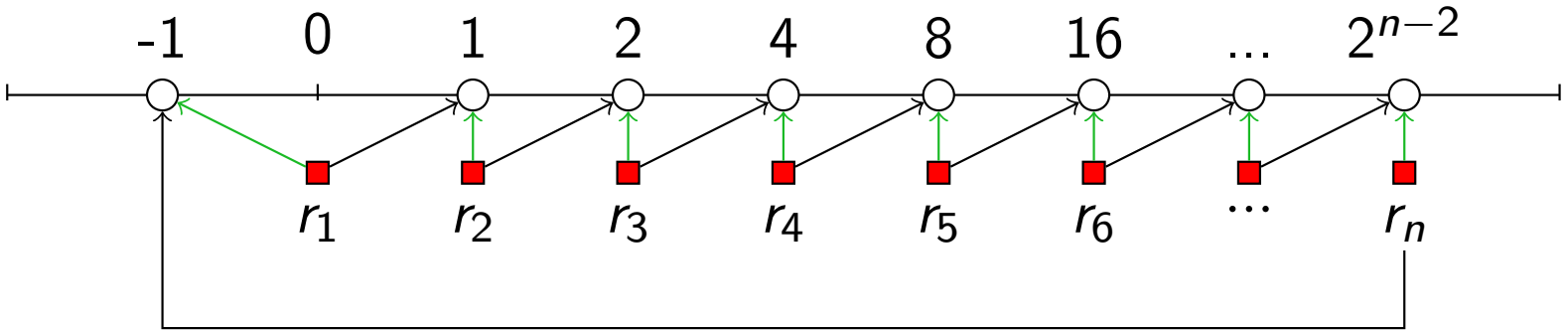


$$c\left(\textsc{Greedy}\left(I\right)\right) = 2^{n-2} + 1$$

$$c\left(\textsc{Opt}\left(I\right)\right) = 1$$

Lowerbound of Greedy

Bottleneck



$$c\left(\textsc{Greedy}\left(I\right)\right) = 2^{n-2} + 1$$

$$c\left(\textsc{Opt}\left(I\right)\right) = 1$$

$$\rho = \max_{I \in \mathcal{I}} \left(\frac{c\left(A\left(I\right)\right)}{c\left(\textsc{Opt}(I)\right)} \right) \geq 2^{n-2}$$

Permutation

Definition (Partial Matching)

A *partial matching* on i requests is an injective function $f : R_i = \{r_1, ..., r_i\} \to S$.

Definition (Optimal Partial Matching)

The *optimal partial matching* on i requests is the function f satisfying

$$f = \underset{g}{\operatorname{argmin}}\, c(g)$$

where g ranges over all partial matchings on i requests. We will define $S_i = \{f(r_1), ..., f(r_i)\}$, the set of servers used by the optimal partial matching on i requests.

Permutation

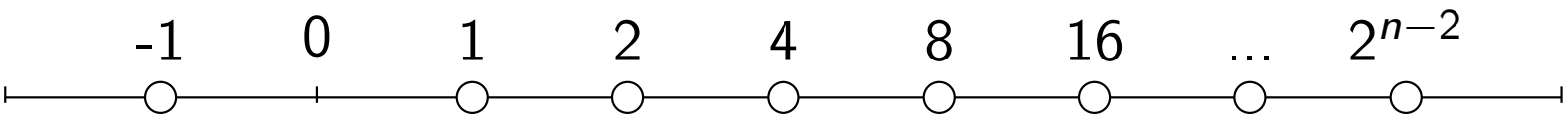
Lemma (From [4])

If S_i is the set of servers used by the optimal partial matching on i requests, then $|S_i \setminus S_{i-1}| = 1$ for all i.

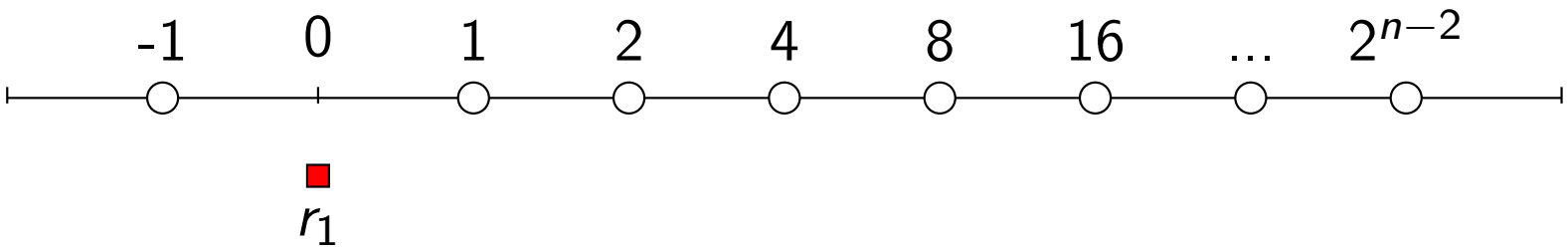
Definition (Permutation)

The Permutation Algorithm is defined as follows: for each r_i compute S_i and let $\{s_i\} = S_i \setminus S_{i-1}$. Then match r_i to s_i.

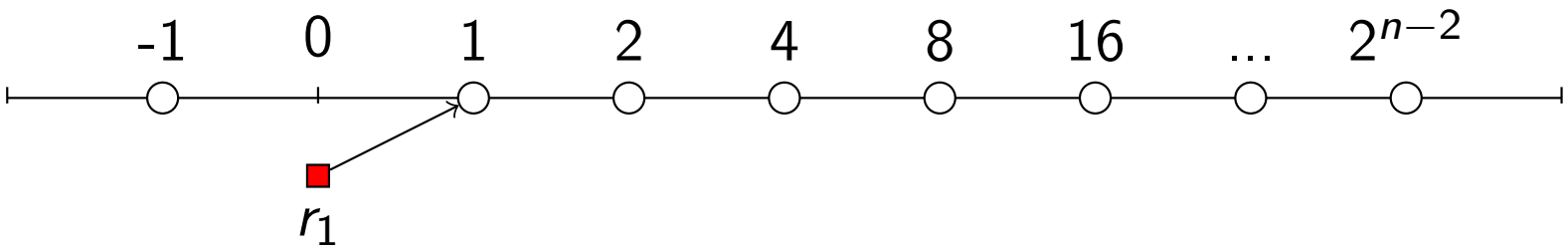
Permutation



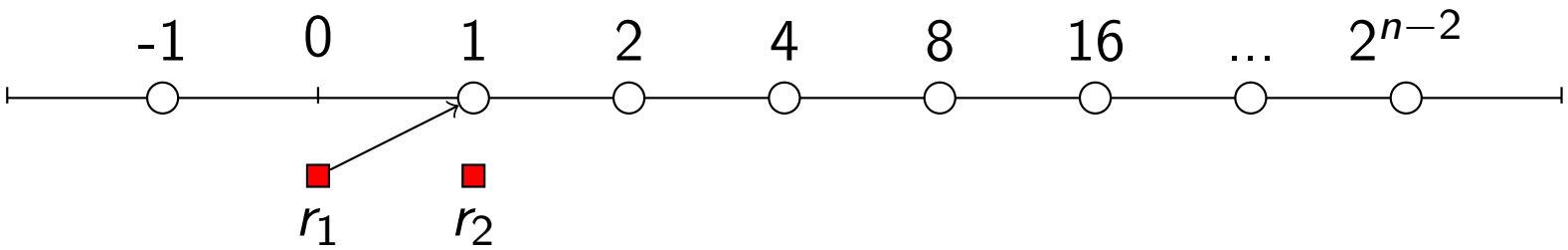
Permutation



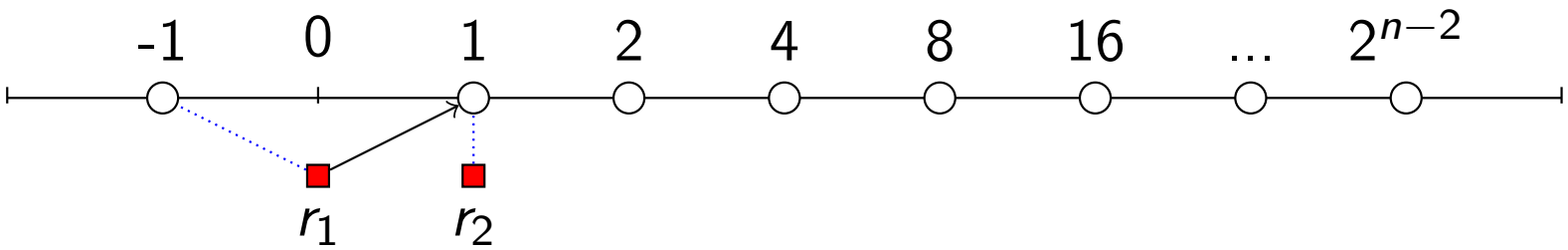
Permutation



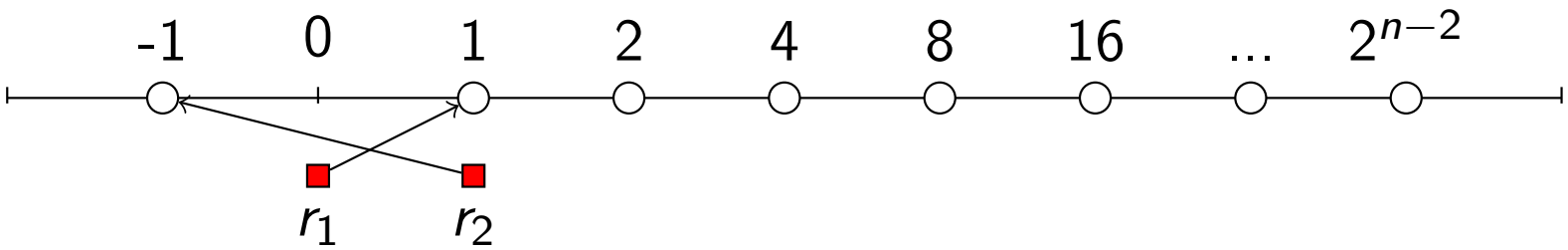
Permutation



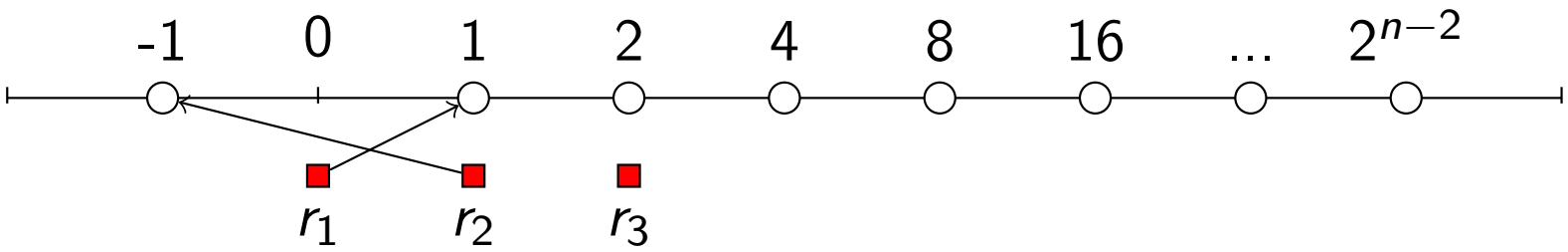
Permutation



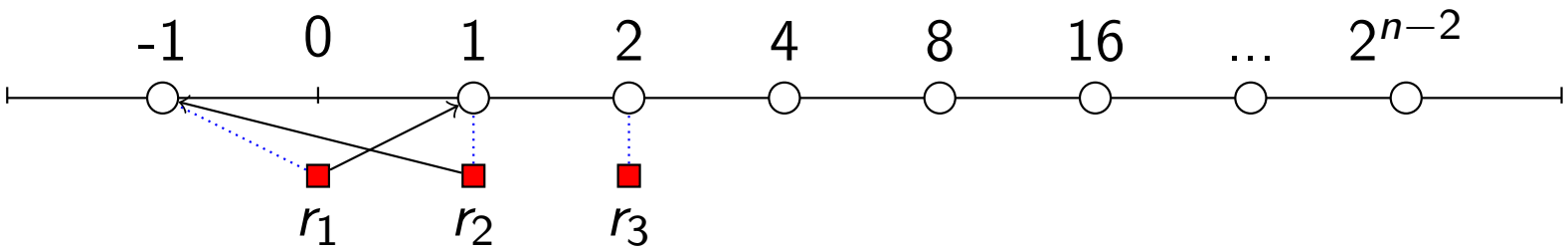
Permutation



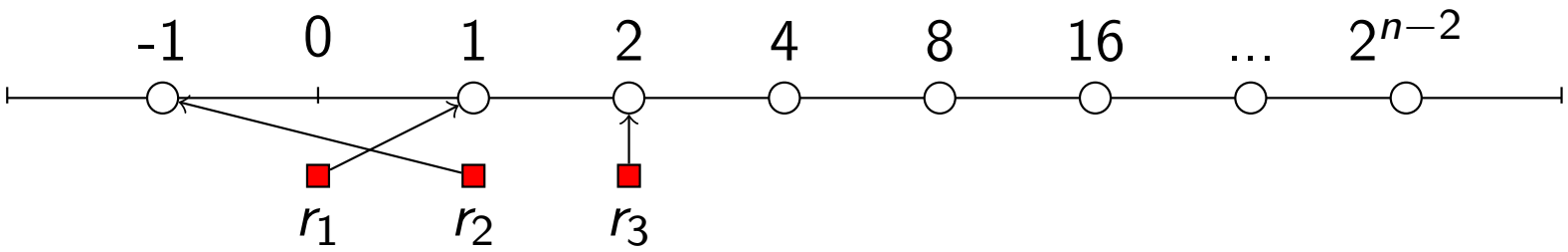
Permutation



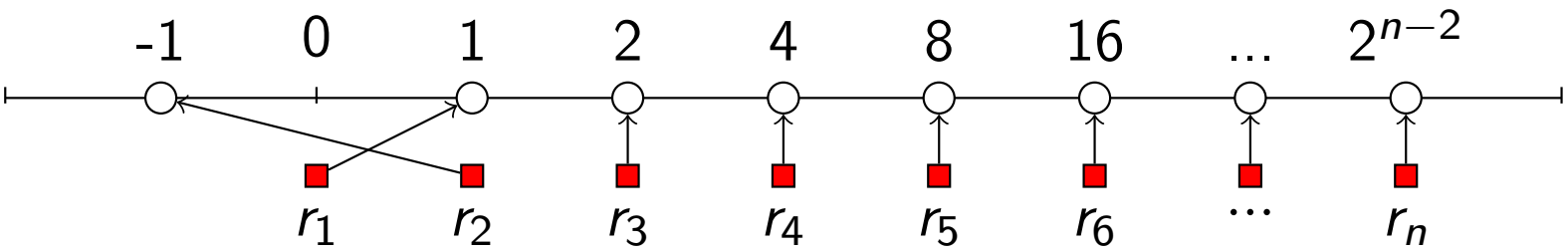
Permutation



Permutation



Permutation



Definition (HARMONIC)

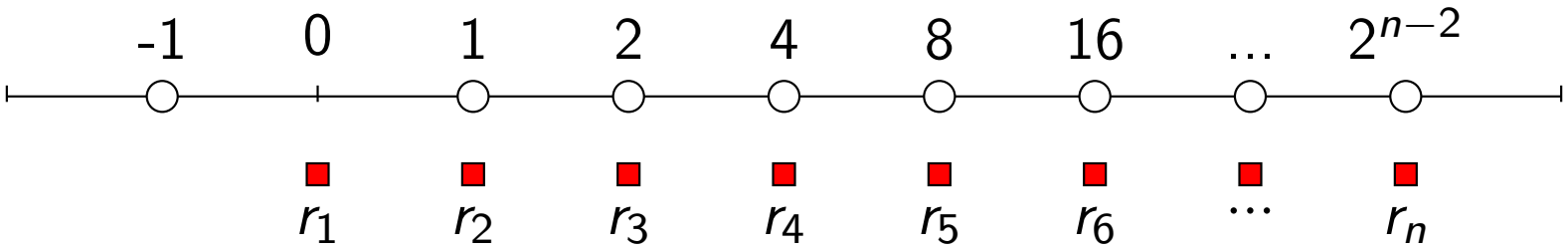
The HARMONIC Algorithm is defined as follows: for each r_i, let d_l be the distance to the closest available server on the left and d_r be the distance to the closest available server on the right. Then assign r_i to the server on the right with probability

$$\frac{d_l}{d_l + d_r}$$

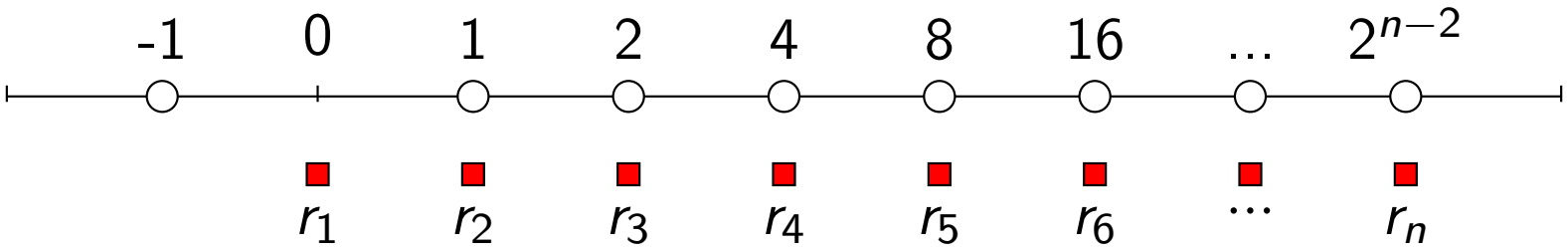
and on the left with probability

$$1 - \frac{d_l}{d_l + d_r} = \frac{d_r}{d_l + d_r}.$$

Harmonic

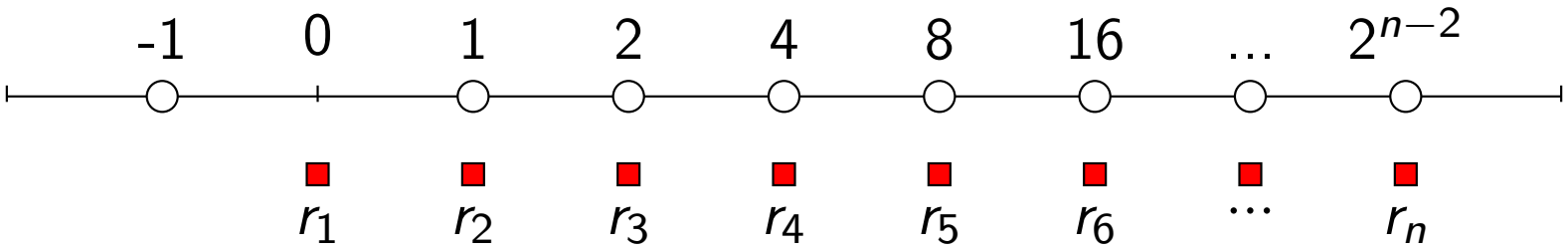


Harmonic



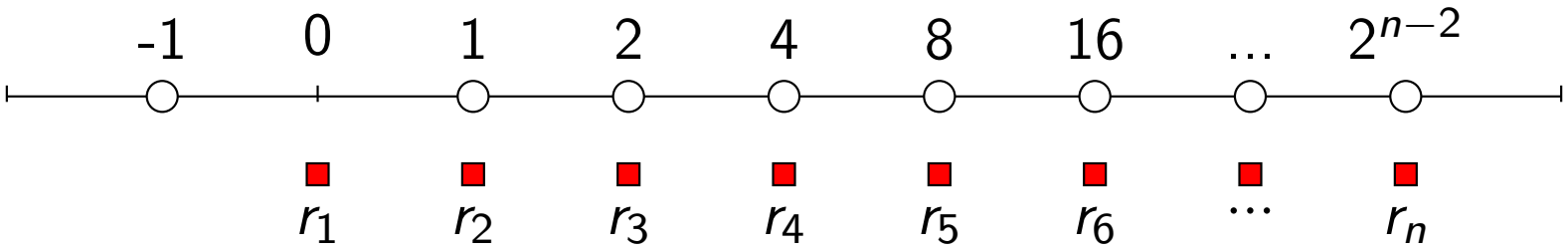
- Let $P(i)$ be the probability that r_i is assigned to the server at -1.

Harmonic



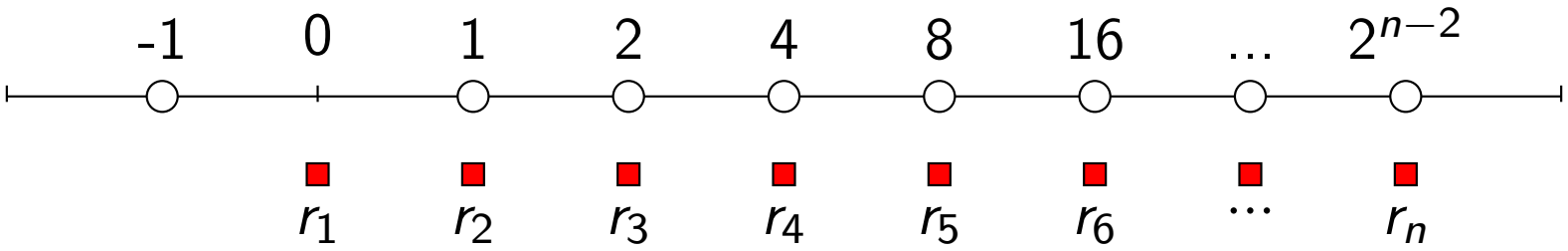
- Let $P(i)$ be the probability that r_i is assigned to the server at -1.
- Then $P(i) = \frac{1}{2^i}$ for $1 \leq i \leq n-1$ and $P(n) = \frac{1}{2^{n-1}}$.

Harmonic



- Let $P(i)$ be the probability that r_i is assigned to the server at -1.
- Then $P(i) = \frac{1}{2^i}$ for $1 \leq i \leq n-1$ and $P(n) = \frac{1}{2^{n-1}}$.
- If r_i is assigned to the left, then the total cost will be $2^{i-1} + 1$ for $2 \leq i \leq n$ and 1 for $i = 1$.

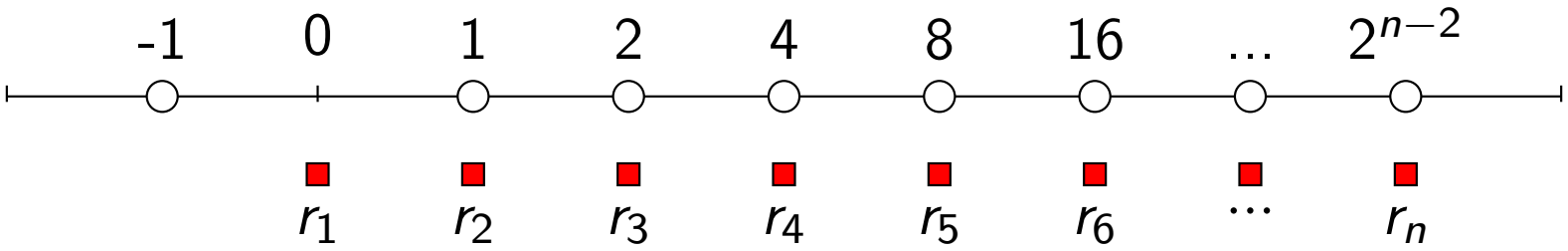
Harmonic



- Let $P(i)$ be the probability that r_i is assigned to the server at -1.
- Then $P(i) = \frac{1}{2^i}$ for $1 \leq i \leq n-1$ and $P(n) = \frac{1}{2^{n-1}}$.
- If r_i is assigned to the left, then the total cost will be $2^{i-1}+1$ for $2 \leq i \leq n$ and 1 for $i = 1$.

$$\mathrm{E}\,(\text{Total Cost}) = \frac{1}{2} + \sum_{i=2}^{n-1} \frac{2^{i-1}+1}{2^i} + \frac{2^{n-1}+1}{2^{n-1}}$$

Harmonic



- Let $P(i)$ be the probability that r_i is assigned to the server at -1.
- Then $P(i) = \frac{1}{2^i}$ for $1 \leq i \leq n-1$ and $P(n) = \frac{1}{2^{n-1}}$.
- If r_i is assigned to the left, then the total cost will be $2^{i-1}+1$ for $2 \leq i \leq n$ and 1 for $i = 1$.

$$\mathrm{E}(\text{Total Cost}) = \frac{1}{2} + \sum_{i=2}^{n-1} \frac{2^{i-1}+1}{2^i} + \frac{2^{n-1}+1}{2^{n-1}}$$

$$= \sum_{i=1}^{n-1} \frac{2^{i-1}+1}{2^i} - \frac{1}{2} + \frac{2^{n-1}+1}{2^{n-1}} = \frac{n}{2} + 1$$

Previous Results

Deterministic:

	Lowerbound	Best known Algorithm	Competitive Ratio
Total:	$9+\epsilon$ [1]	Permutation [4][5]	$2n-1$
Bottleneck:	$1.5n$ [3]	Permutation	$2n-1$

Randomized:

	Lowerbound	Best known Algorithm	Competitive Ratio
Total:	—	Harmonic [2]	$O(\log(n))$
Bottleneck:	—	Permutation	$2n-1$

Our Results

Deterministic:

	Lowerbound	Best known Algorithm	Competitive Ratio
Total:	$9+\epsilon$ [1]	PERMUTATION [4][5]	$2n-1$
Bottleneck:	$1.5n$ [3]	PERMUTATION	$2n-1$

Randomized:

	Lowerbound	Best known Algorithm	Competitive Ratio
Total:	2	HARMONIC [2]	$O(\log(n))$
Bottleneck:	$\frac{n}{2}$	PERMUTATION	$2n-1$

No Skip Necessary

Theorem (No Skip Necessary)

Given any instance of the problem for the minimum weight objective, there exists an optimal min-weight matching function $f : R \to S$ that satisfies $f(r_j) \in N_f(r_j)$, where $N_f(r_j)$ denotes the neighboring unmatched servers of r_j on the line. Formally, by letting

$$S_j = S \setminus \{f(r_1), ..., f(r_{j-1})\},$$
$$L_j = \max\{s | s \in S_j, s \leq r_j\}, \text{ and}$$
$$R_j = \min\{s | s \in S_j, s \geq r_j\},$$

then $N_f(r_j) = \{L_j, R_j\}$.

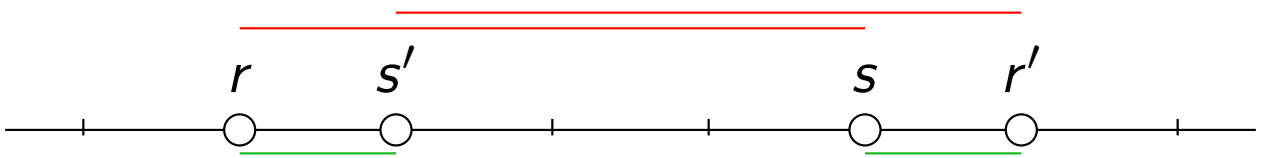
No Skip Necessary

Proof Idea

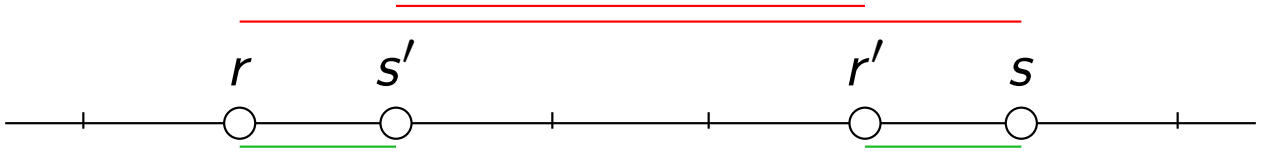
- Proceed by contradiction - let r_j be the first request where the claim fails in some instance.
- Let
$$r = r_j, \quad s = f(r_j), \quad s' = R_j, \quad r' = f^{-1}(s').$$
- If we can match to s' without reducing the total difference between the rest of the servers, then we have a contradiction.
- Note that $r < s'$ and $s \notin [r, s']$.
- 8 Cases: 3 possible cases where $r = \min\{r, s, s', r'\}$, 2 cases where $r' = \min\{r, s, s', r'\}$, and 3 cases where $s = \min\{r, s, s', r'\}$.

No Skip Necessary

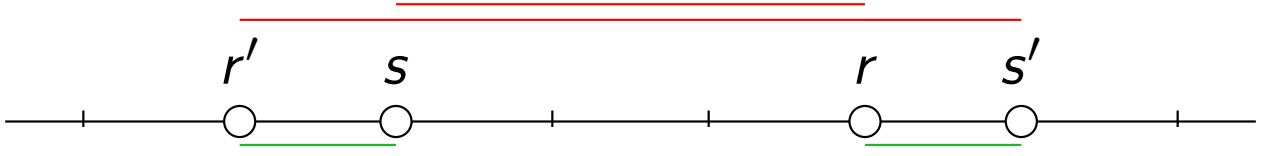
Case 1



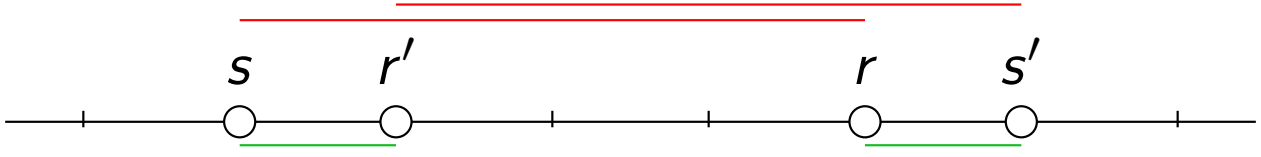
Case 2



Case 5

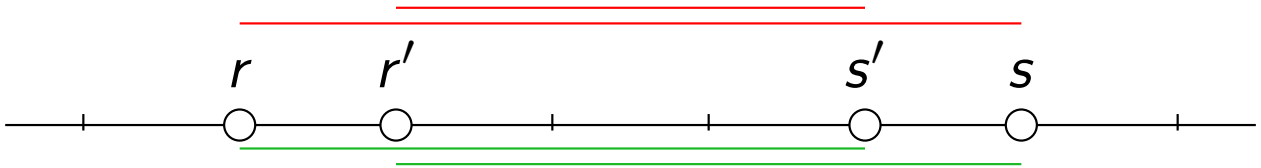


Case 6

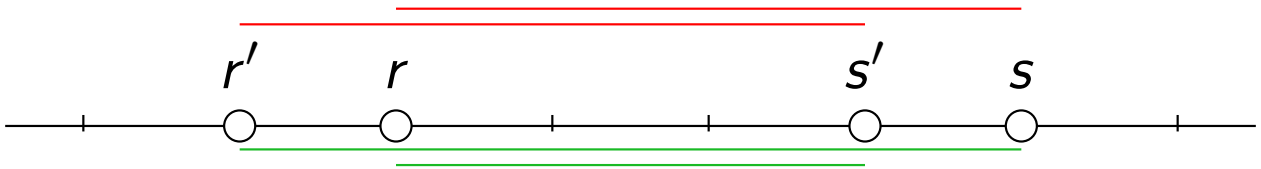


No Skip Necessary

Case 3

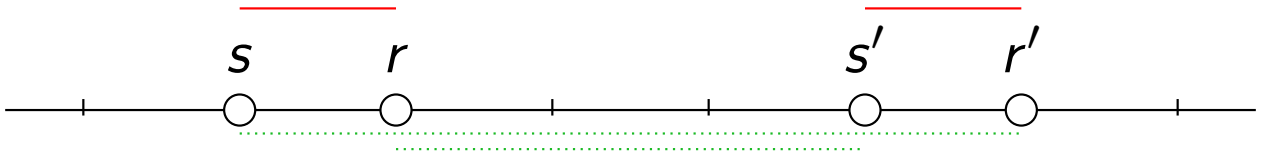


Case 4



No Skip Necessary

Case 7



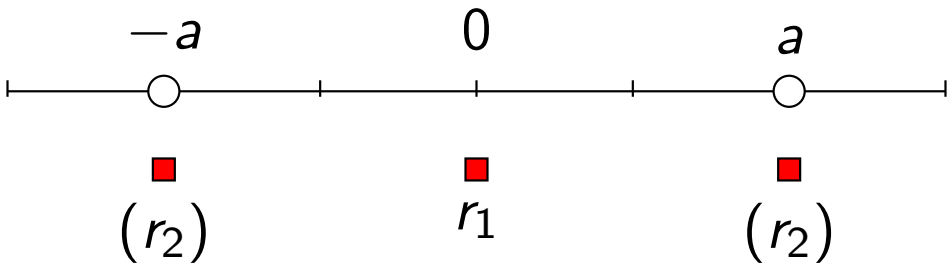
Case 8

No Skip Necessary

Case 7'/8'

Lowerbound for Min-Weight

Consider the following instance with $n = 2$ servers:



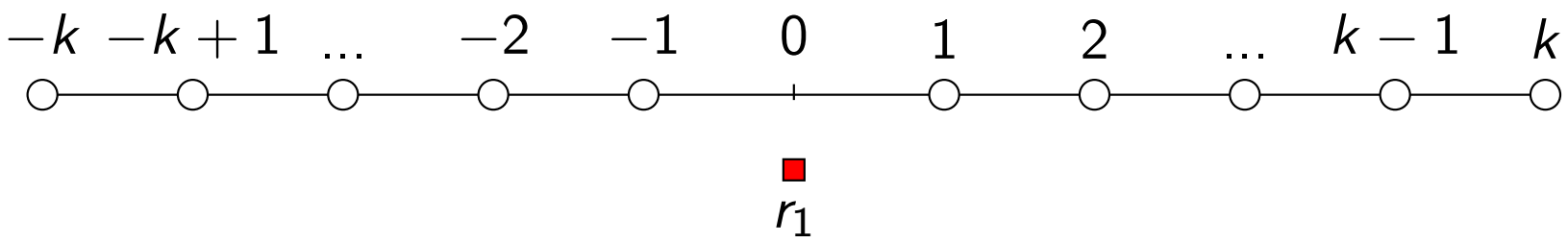
The request sequence is $R = (0, \pm a)$, ie the second request is either $\{-a, a\}$ with uniform probability.
Hence for any algorithm $A \in \mathcal{A}$:

$$\mathrm{E}\left[c\left(A\left(I\right)\right)\right] = \frac{1}{2}\left(a + 0\right) + \frac{1}{2}\left(a + 2a\right) = \frac{4a}{2} = 2a.$$

However the optimal cost will always be just a, hence $\rho \geq \frac{2a}{a} = 2$.

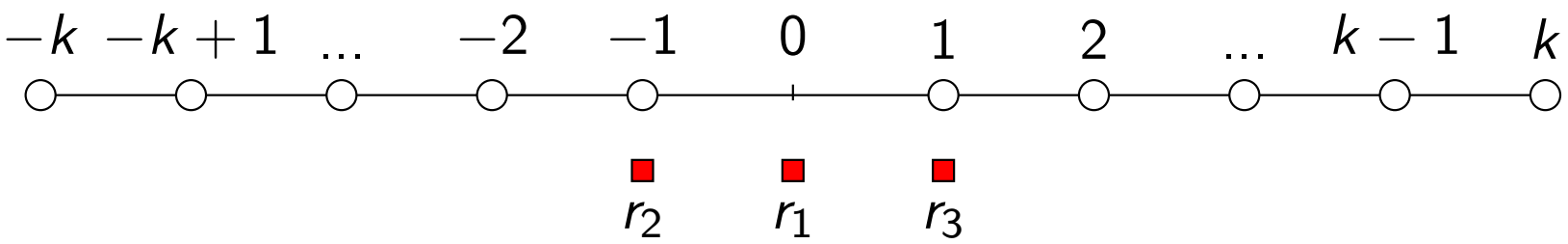
Lowerbound for Randomized Algorithms under the Bottleneck Objective

Consider the following instance with $n = 2k$ servers:



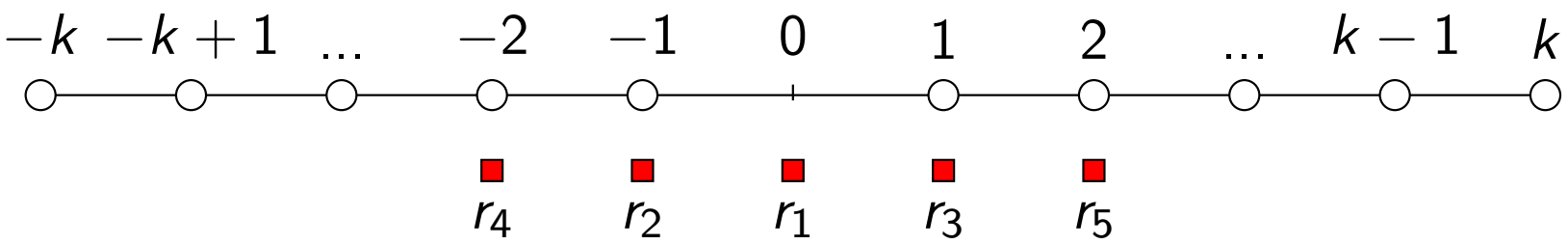
Lowerbound for Randomized Algorithms under the Bottleneck Objective

Consider the following instance with $n = 2k$ servers:



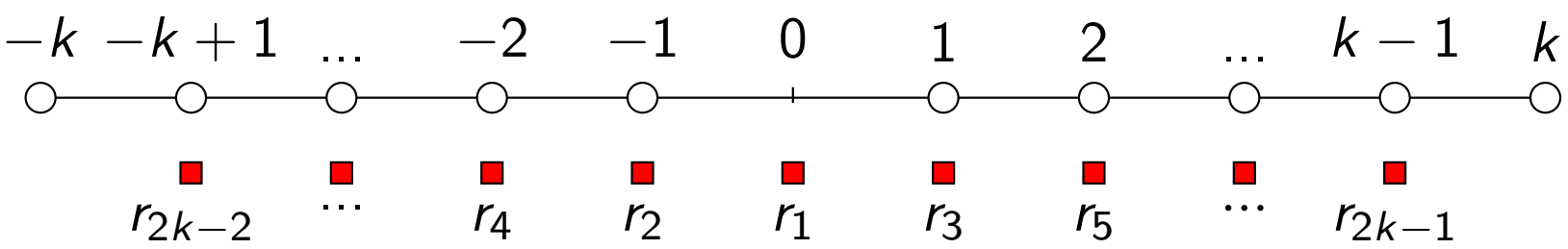
Lowerbound for Randomized Algorithms under the Bottleneck Objective

Consider the following instance with $n = 2k$ servers:



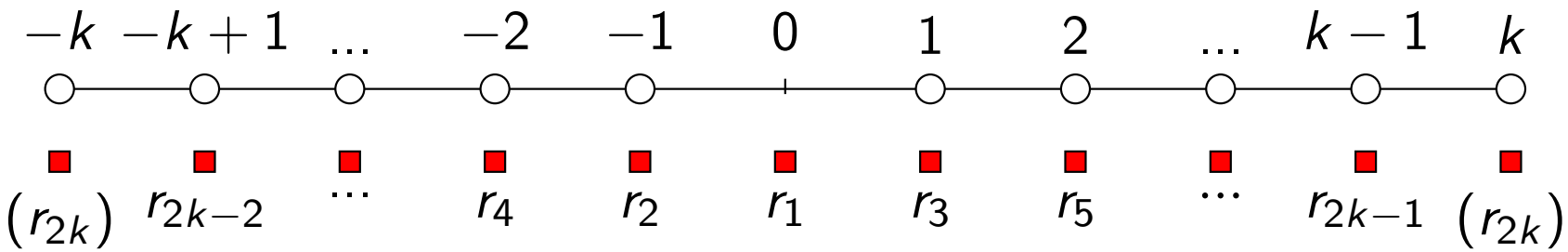
Lowerbound for Randomized Algorithms under the Bottleneck Objective

Consider the following instance with $n = 2k$ servers:



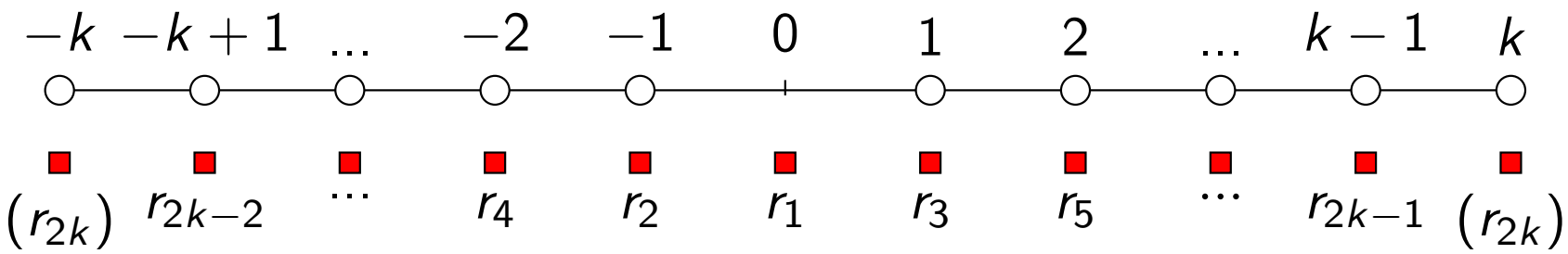
Lowerbound for Randomized Algorithms under the Bottleneck Objective

Consider the following instance with $n = 2k$ servers:



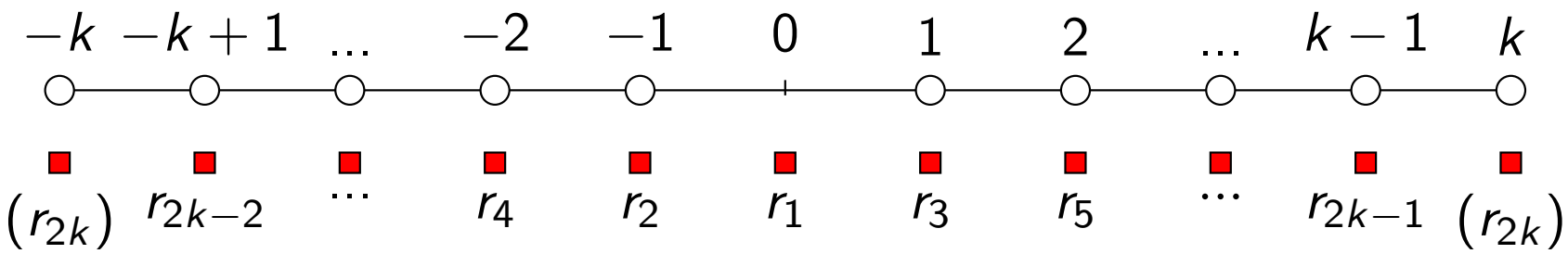
The final request r_{2k} is either $\{-k, k\}$ with uniform probability.

Lowerbound for Randomized Algorithms under the Bottleneck Objective



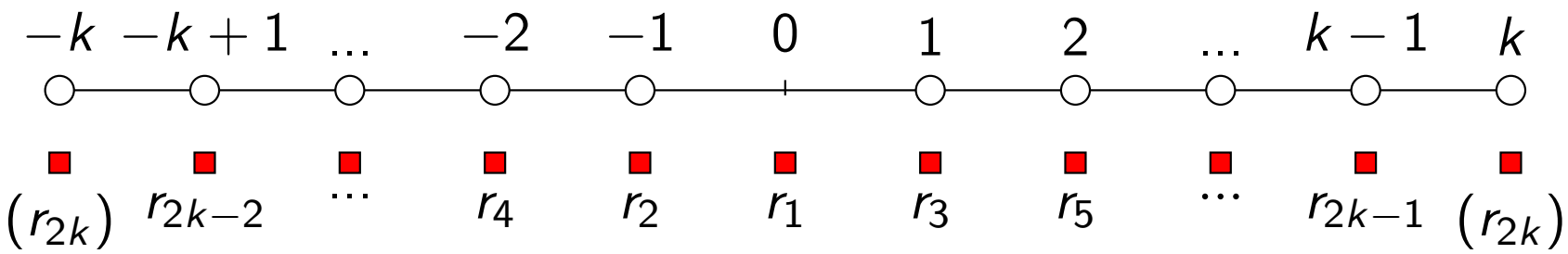
- The important thing here is actually only the last request

Lowerbound for Randomized Algorithms under the Bottleneck Objective



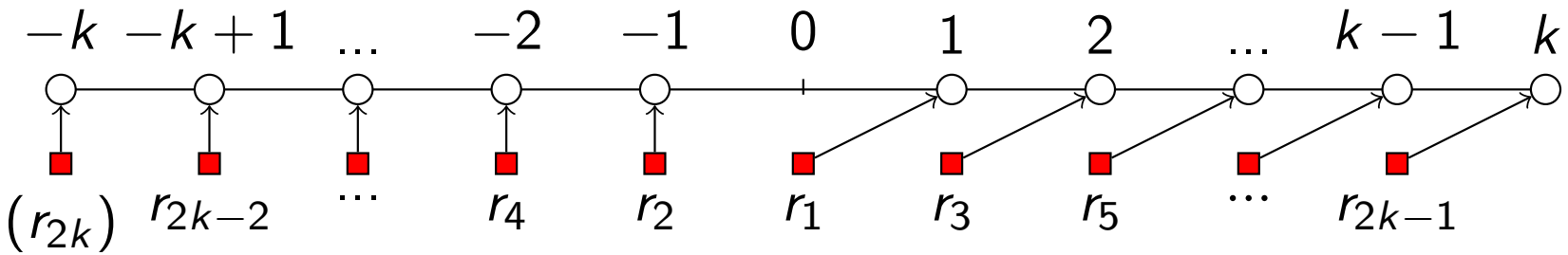
- The important thing here is actually only the last request
- After the first $2k-1$ requests, there will only be one available server, call it s

Lowerbound for Randomized Algorithms under the Bottleneck Objective



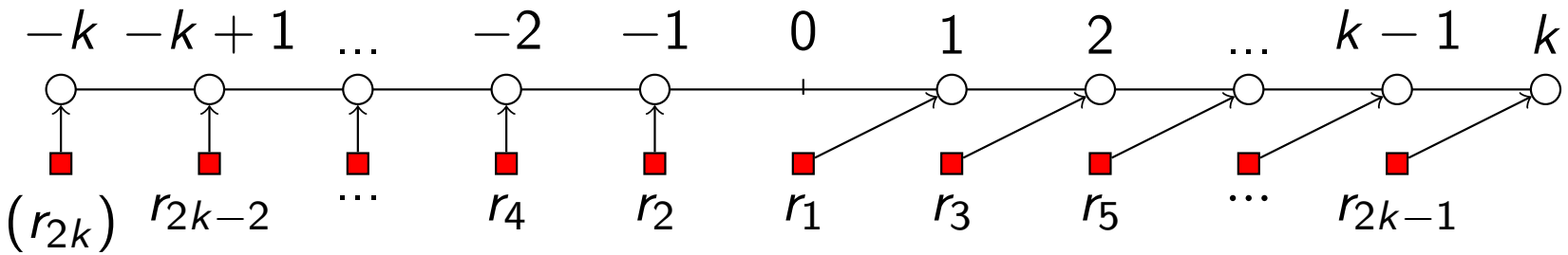
- The important thing here is actually only the last request
- After the first $2k-1$ requests, there will only be one available server, call it s
- $\mathrm{E}\left[|s - r_{2k}|\right] = \frac{1}{2}(k-s) + \frac{1}{2}(s+k) = k$

Lowerbound for Randomized Algorithms under the Bottleneck Objective



- The important thing here is actually only the last request
- After the first $2k-1$ requests, there will only be one available server, call it s
- $\mathrm{E}\left[|s - r_{2k}|\right] = \frac{1}{2}(k-s) + \frac{1}{2}(s+k) = k$
- The optimal bottleneck edge cost will always be just 1

Lowerbound for Randomized Algorithms under the Bottleneck Objective



- The important thing here is actually only the last request
- After the first $2k-1$ requests, there will only be one available server, call it s
- $\mathrm{E}\left[|s - r_{2k}|\right] = \frac{1}{2}(k-s) + \frac{1}{2}(s+k) = k$
- The optimal bottleneck edge cost will always be just 1
- Hence $\rho \geq k = \frac{n}{2}$

Concluding Remarks

Deterministic:

	Lowerbound	Best known Algorithm	Competitive Ratio
Total:	$9+\epsilon$ [1]	Permutation [4][5]	$2n-1$
Bottleneck:	$1.5n$ [3]	Permutation	$2n-1$

Randomized:

	Lowerbound	Best known Algorithm	Competitive Ratio
Total:	2	Harmonic [2]	$O(\log(n))$
Bottleneck:	$\frac{n}{2}$	Permutation	$2n-1$

Acknowledgements

- Christine Chung
- Perry Susskind
- Gabriella Silva

Resources

B. Fuchs, W. Hochstättler, and W. Kern.

Online matching on a line.

In *Electronic Notes In Discrete Mathematics*. Elsevier, 2003.

Anupam Gupta and Kevin Lewi.

The online metric matching problem for doubling metrics.

In Artur Czumaj, Kurt Mehlhorn, Andrew Pitts, and Roger Wattenhofer, editors, *Automata, Languages, and Programming*, volume 7391 of *Lecture Notes in Computer Science*, pages 424–435. Springer Berlin Heidelberg, 2012.

R. Idury and A. A. Schäffer.

A better lower bound for on-line bottleneck matching.

http://www.ncbi.nlm.nih.gov/core/assets/cbb/files/Firehouse.pdf.

Accessed: 2015-09-7.

B. Kalyanasundaram and K. Pruhs.

Online weighted matching.

J. Algorithms, 14(3):478–488, 1993.

S. Khuller, S. G. Mitchell, and V. V. Vazirani.

On-line algorithms for weighted bipartite matching and stable marriages.

Theoretical Computer Science, 127:251–264, 1994.

Theorem (Yao's Principle)

Given a probability distribution p over $\mathcal{A}_{\mathcal{D}}$ and q over $\mathcal{I}$, where A is some algorithm chosen according to p and I some instance chosen according to q, then

$$\max_{i \in \mathcal{I}} \mathrm{E}\left[c\left(A\left(i\right)\right)\right] \geq \min_{a \in \mathcal{A}_{\mathcal{D}}} \mathrm{E}\left[c\left(a\left(I\right)\right)\right].$$