Twenty (Simple) Questions

Joint work with
Yuval Dagan, Ariel Gabizon, Daniel Kane, Shay Moran

Yuval Filmus, 26 April 2021, HUJI CS Colloquium

The Game of 20 Questions

The Game of 20 Questions

Bob

Thinks of a number between 1 to *n*

The Game of 20 Questions

Alice

Finds the number by asking Yes/No questions

Bob

Thinks of a number between 1 to n

The Game of 20 Questions

Alice

Finds the number by asking Yes/No questions

Bob

Thinks of a number between 1 to n

Cooperative!

The Game of 20 Questions

Alice

Finds the number by asking Yes/No questions

binary search: log*n* questions

Bob

Thinks of a number between 1 to *n*

Cooperative!

Distributional 20 Questions

Distributional 20 Questions

Bob

Samples a number between 1 to n **according to μ**

Distributional 20 Questions

Alice

Finds the number by asking Yes/No questions

Bob

Samples a number between 1 to n **according to μ**

Distributional 20 Questions

Alice

Finds the number by asking Yes/No questions

Bob

Samples a number between 1 to n **according to μ**

μ known to both parties!

Distributional 20 Questions

Alice

Finds the number by asking Yes/No questions

Huffman's algorithm: H(μ)+1 questions on average

Bob

Samples a number between 1 to n **according to μ**

μ known to both parties!

Distributional 20 Questions

Alice

Finds the number by asking Yes/No questions

H(μ) = entropy of μ = amortized # questions when solving many games in parallel

Huffman's algorithm: H(μ)+1 questions on average

Bob

Samples a number between 1 to n **according to μ**

μ known to both parties!

Distributional 20 Questions

Huffman's algorithm could involve complicated questions:

Distributional 20 Questions

Huffman's algorithm could involve complicated questions:

Hunter Desportes

Is x one of 2,3,5,7,11,13?

Distributional 20 Questions

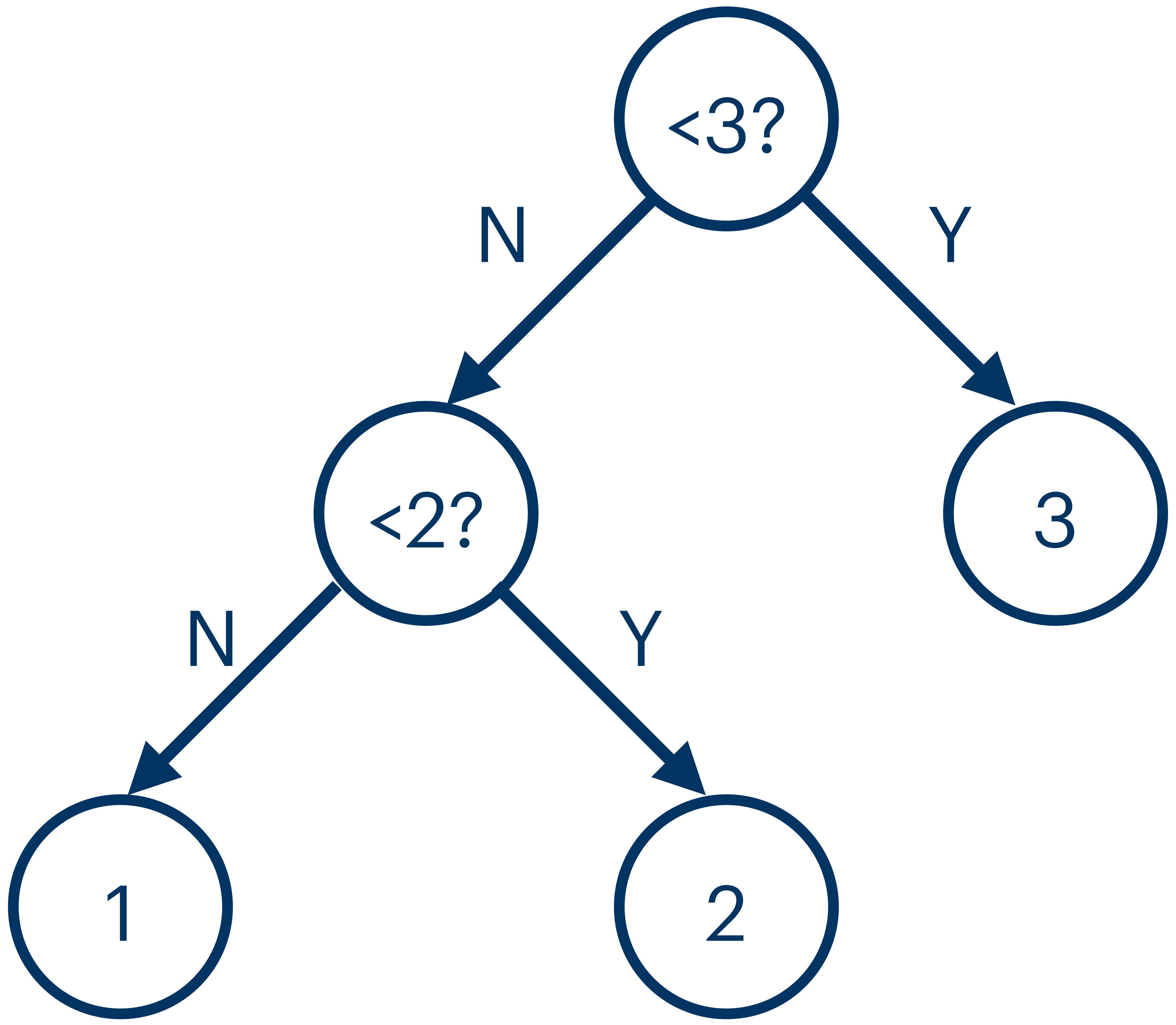
Huffman's algorithm could involve complicated questions:

Hunter Desportes

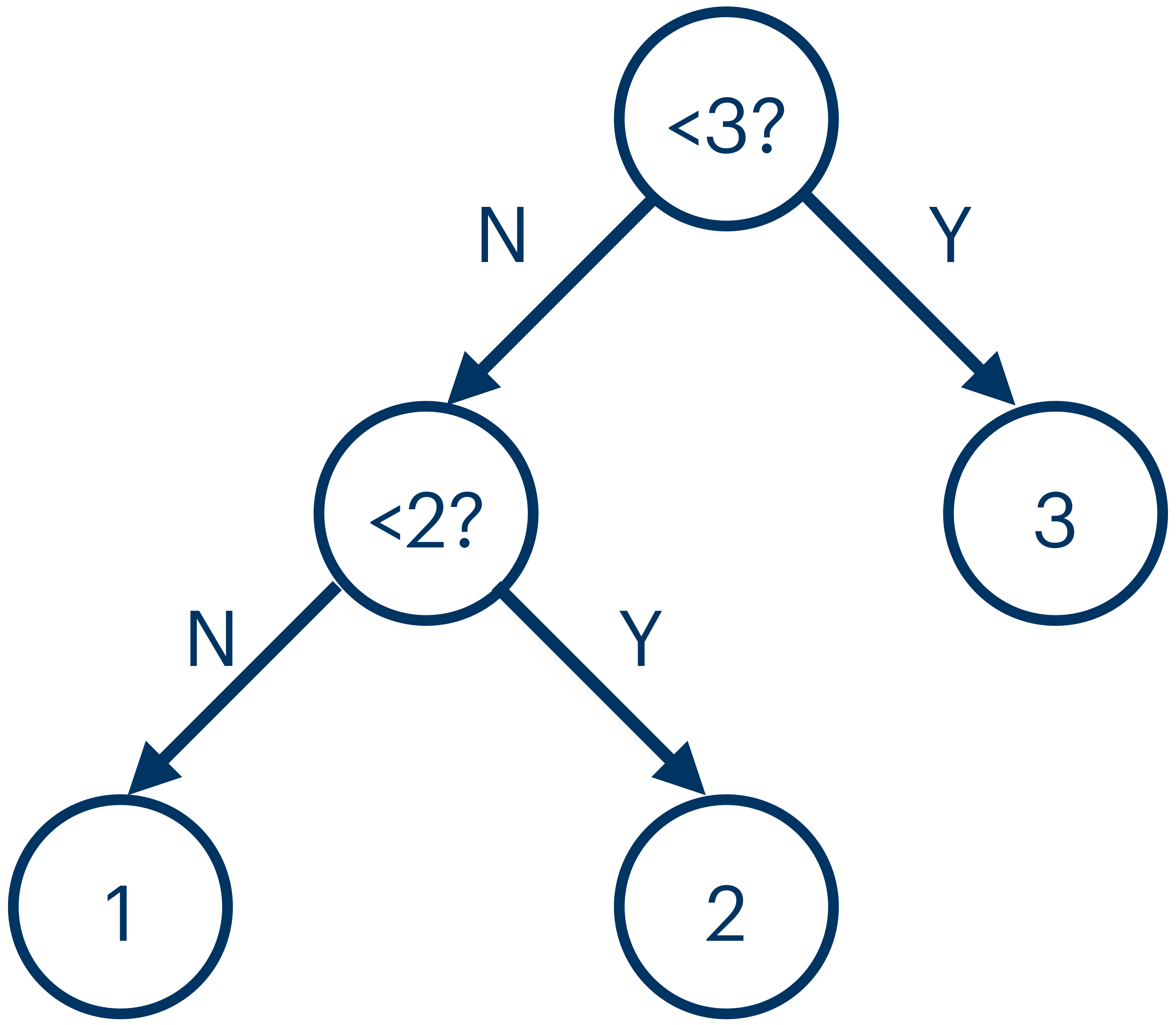
Is x one of
2,3,5,7,11,13?

What can we accomplish using simple questions?

Binary Search Trees

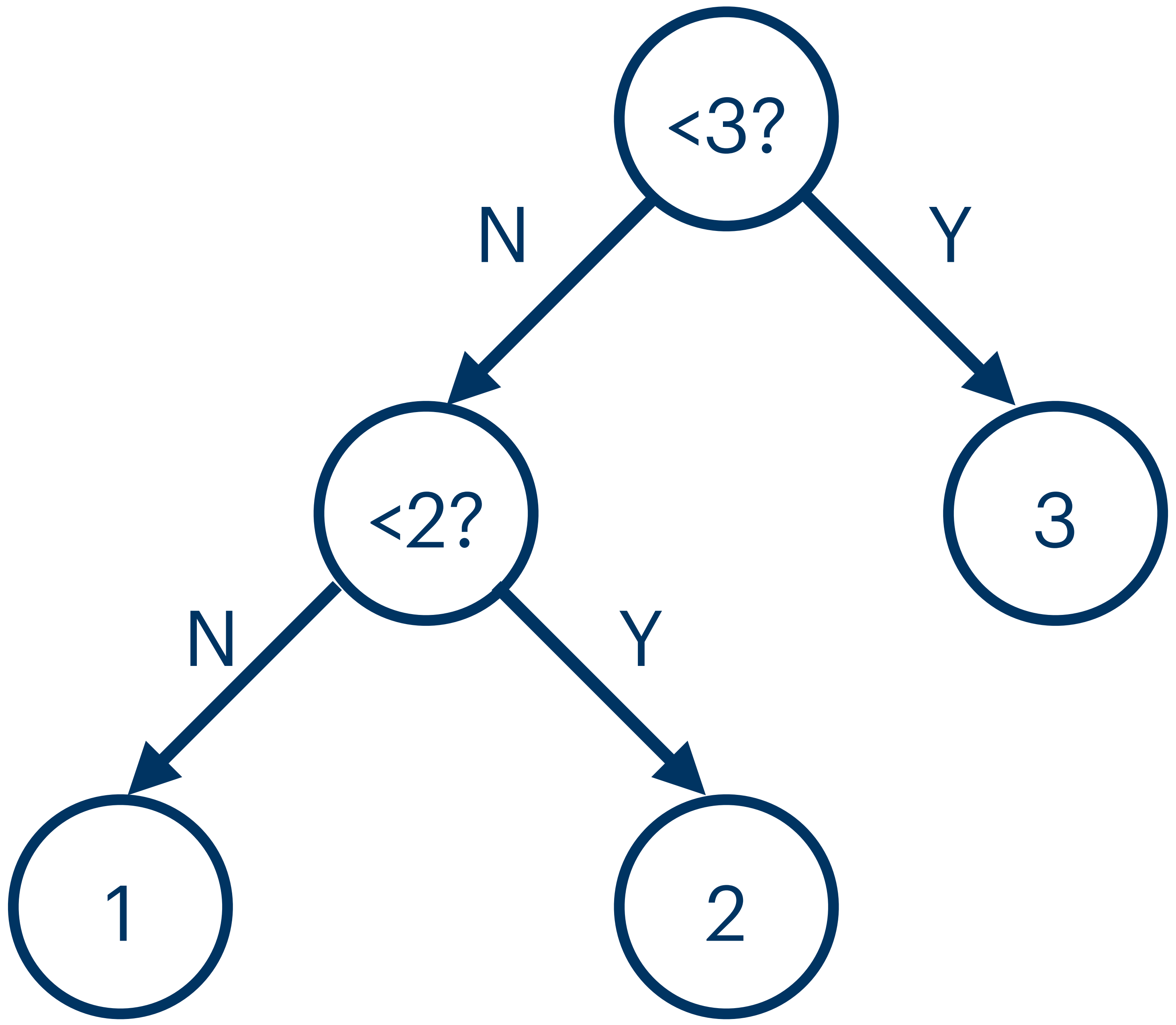


Binary Search Trees



Gilbert and Moore: optimal BST achieves $H(\mu)+2$

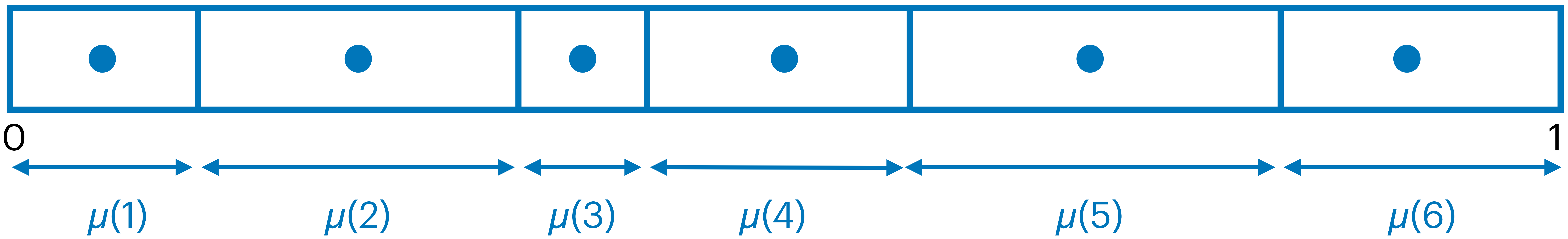
Binary Search Trees



Gilbert and Moore: optimal BST achieves H(μ)+2

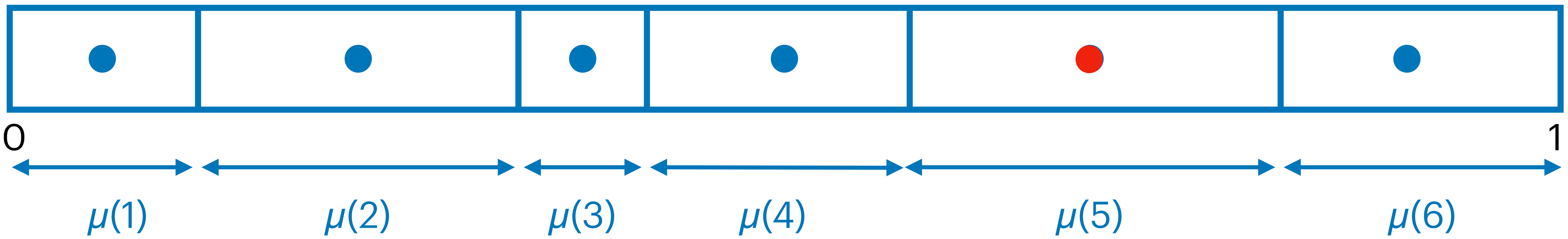
(optimal for distributions concentrated on some $x \in \{2,\ldots,n-1\}$)

Gilbert–Moore Algorithm



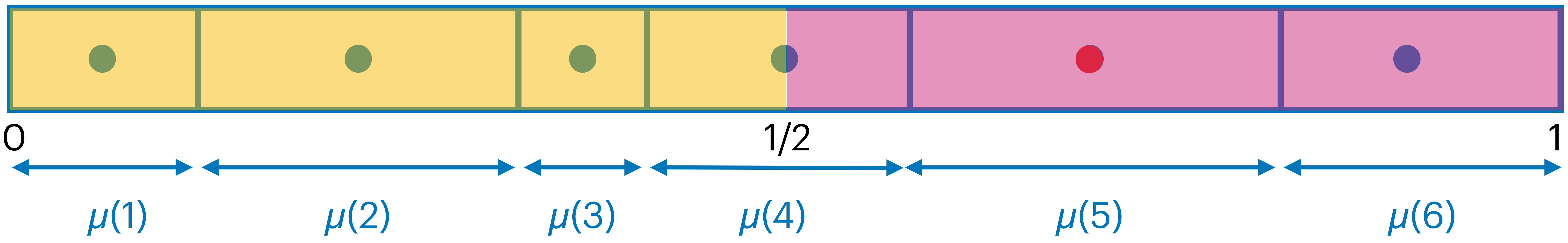
Binary search over [0,1]

Gilbert–Moore Algorithm



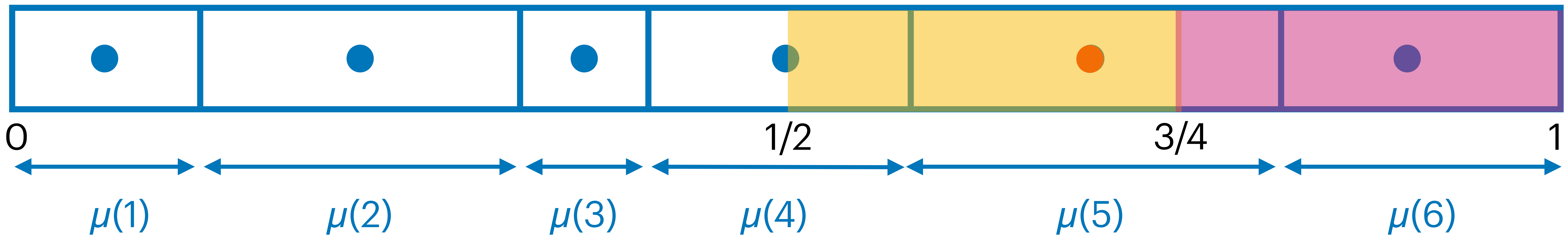
Binary search over [0,1]

Gilbert–Moore Algorithm



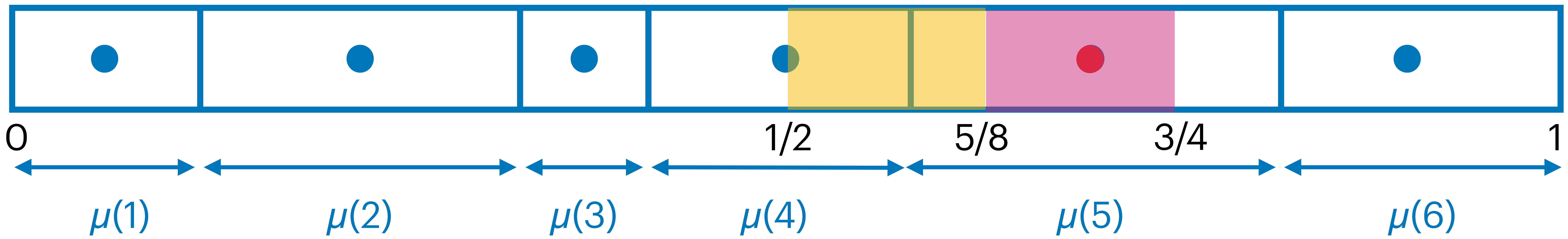
Binary search over [0,1]

Gilbert–Moore Algorithm



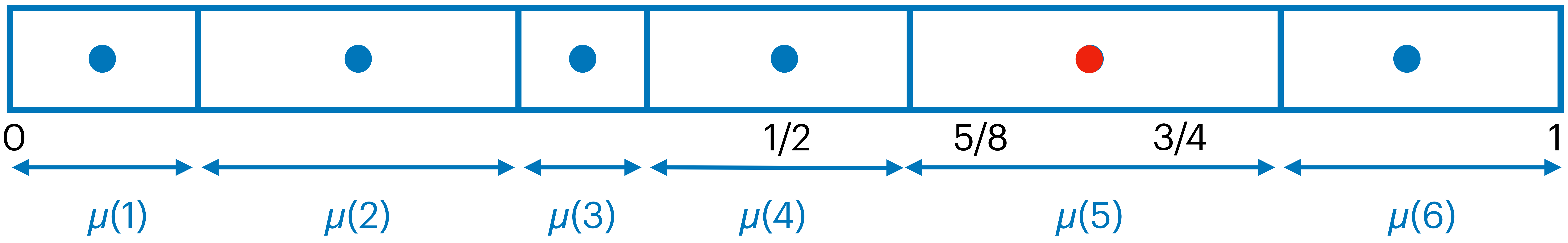
Binary search over [0,1]

Gilbert–Moore Algorithm



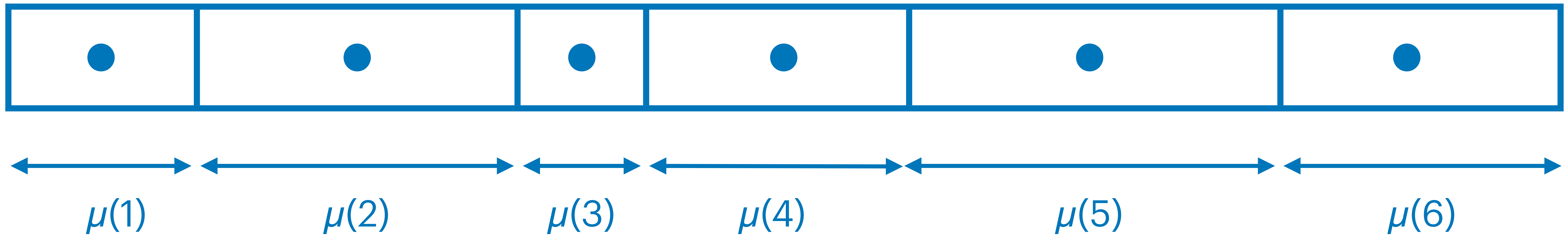
Binary search over [0,1]

Gilbert–Moore Algorithm



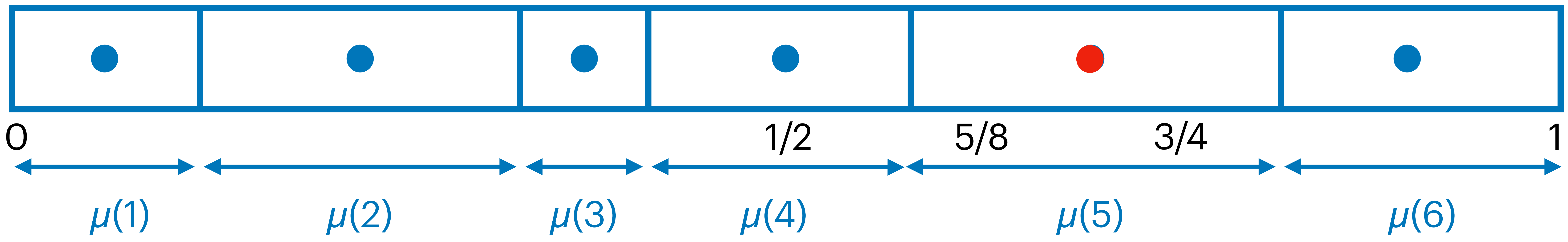
Binary search over [0,1]

Rissanen–Horibe Algorithm



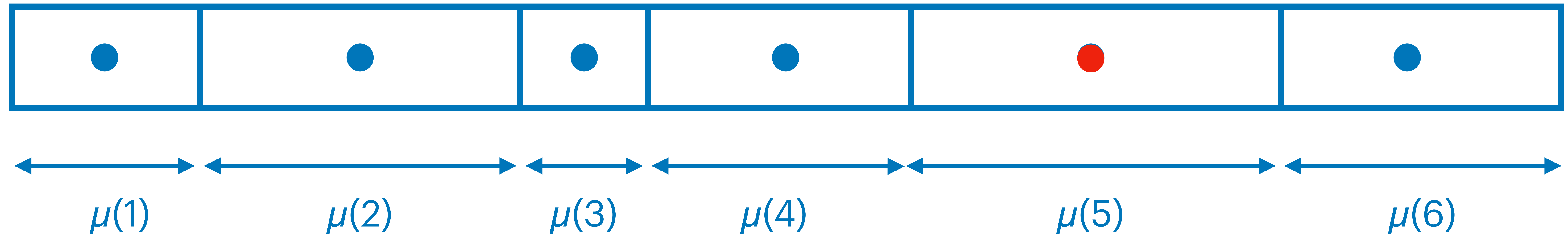
Ask most informative question

Gilbert–Moore Algorithm



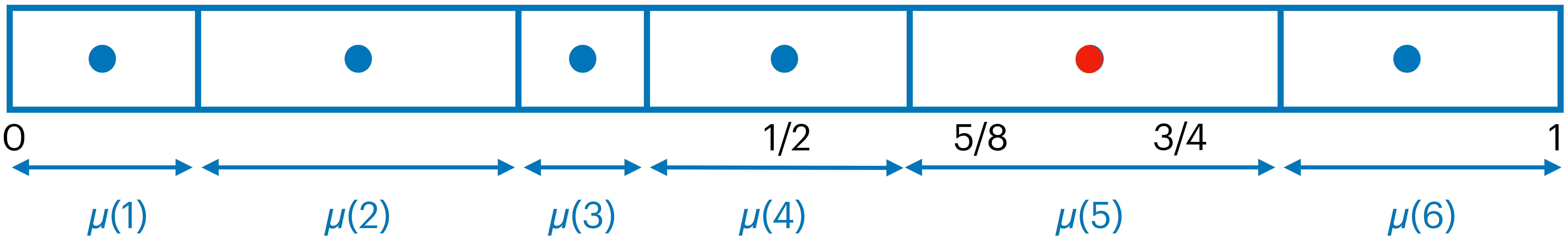
Binary search over [0,1]

Rissanen–Horibe Algorithm



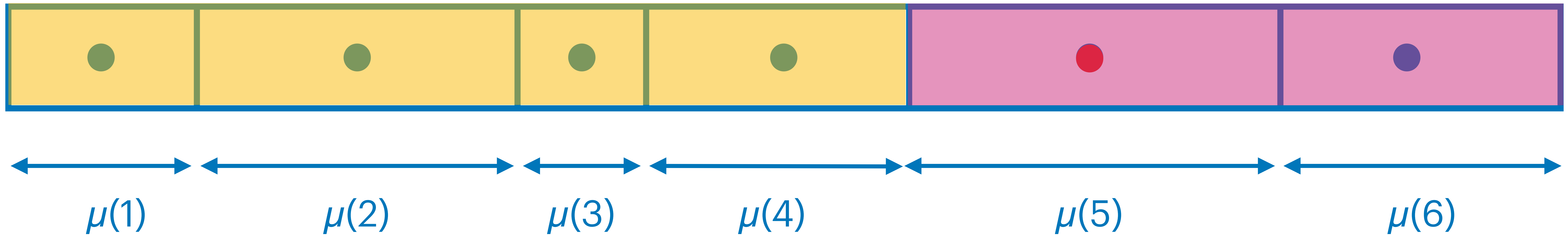
Ask most informative question

Gilbert–Moore Algorithm



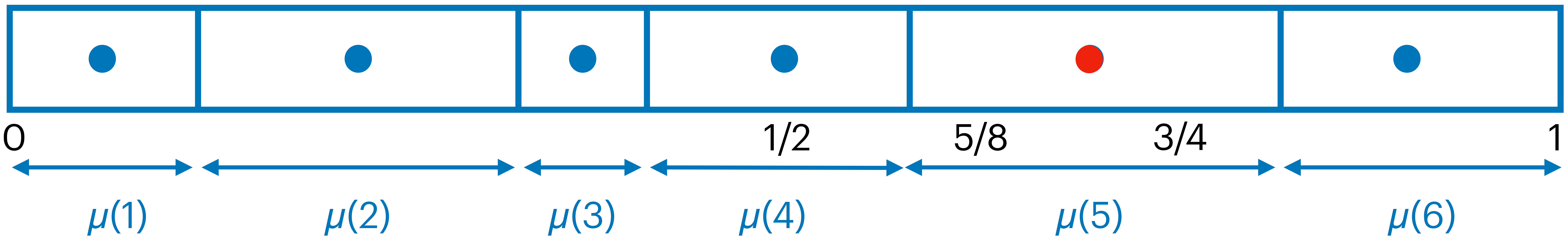
Binary search over [0,1]

Rissanen–Horibe Algorithm



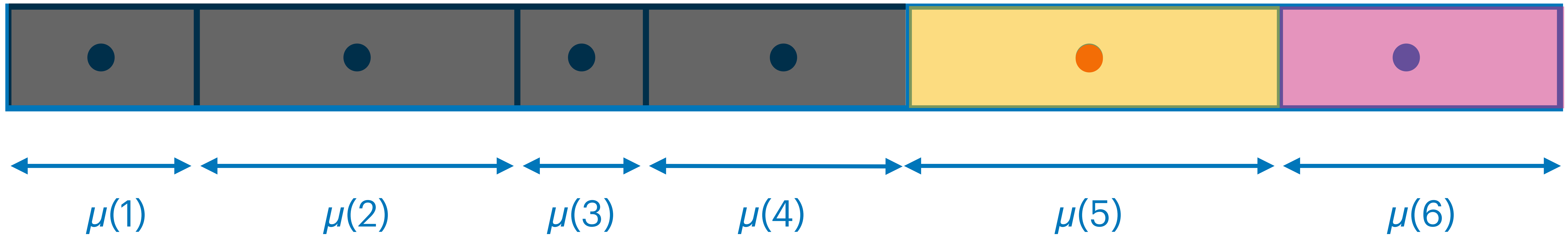
Ask most informative question

Gilbert–Moore Algorithm



Binary search over [0,1]

Rissanen–Horibe Algorithm

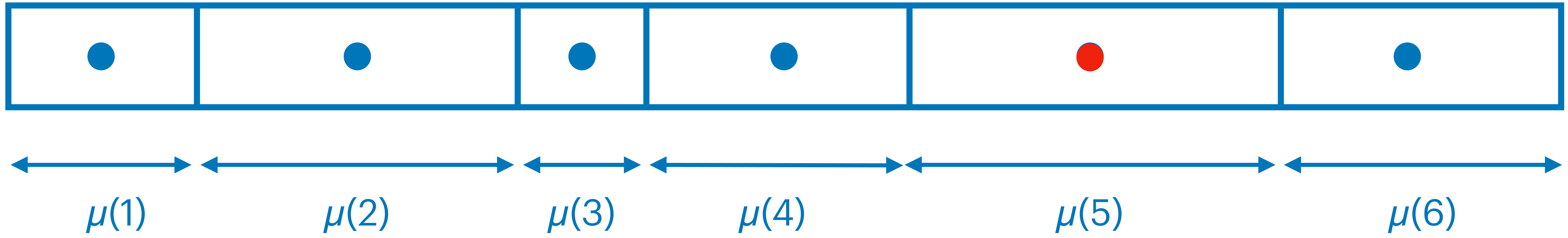


Ask most informative question

Gilbert–Moore Algorithm

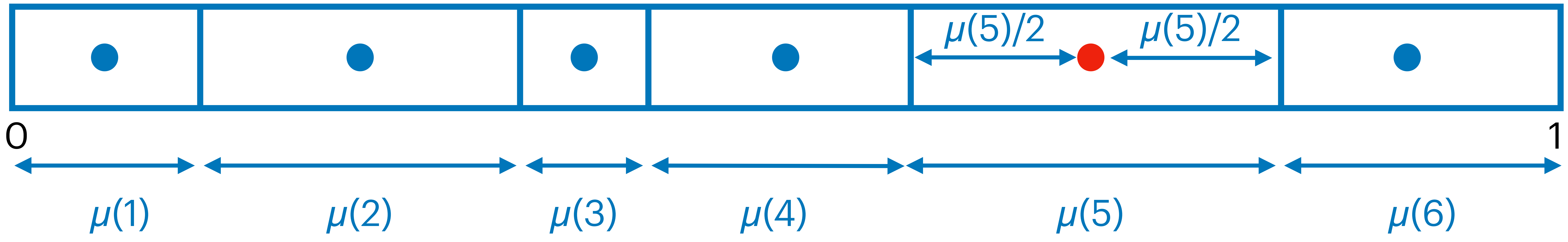
Both algorithms guarantee $H(\mu)+2$

Rissanen–Horibe Algorithm

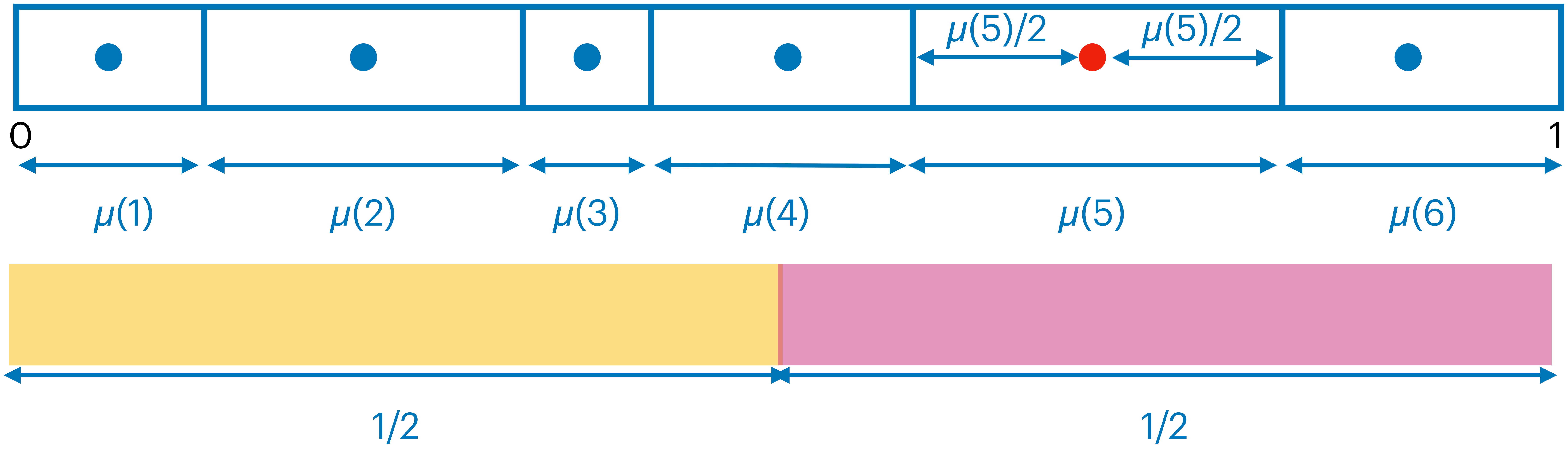


Ask most informative question

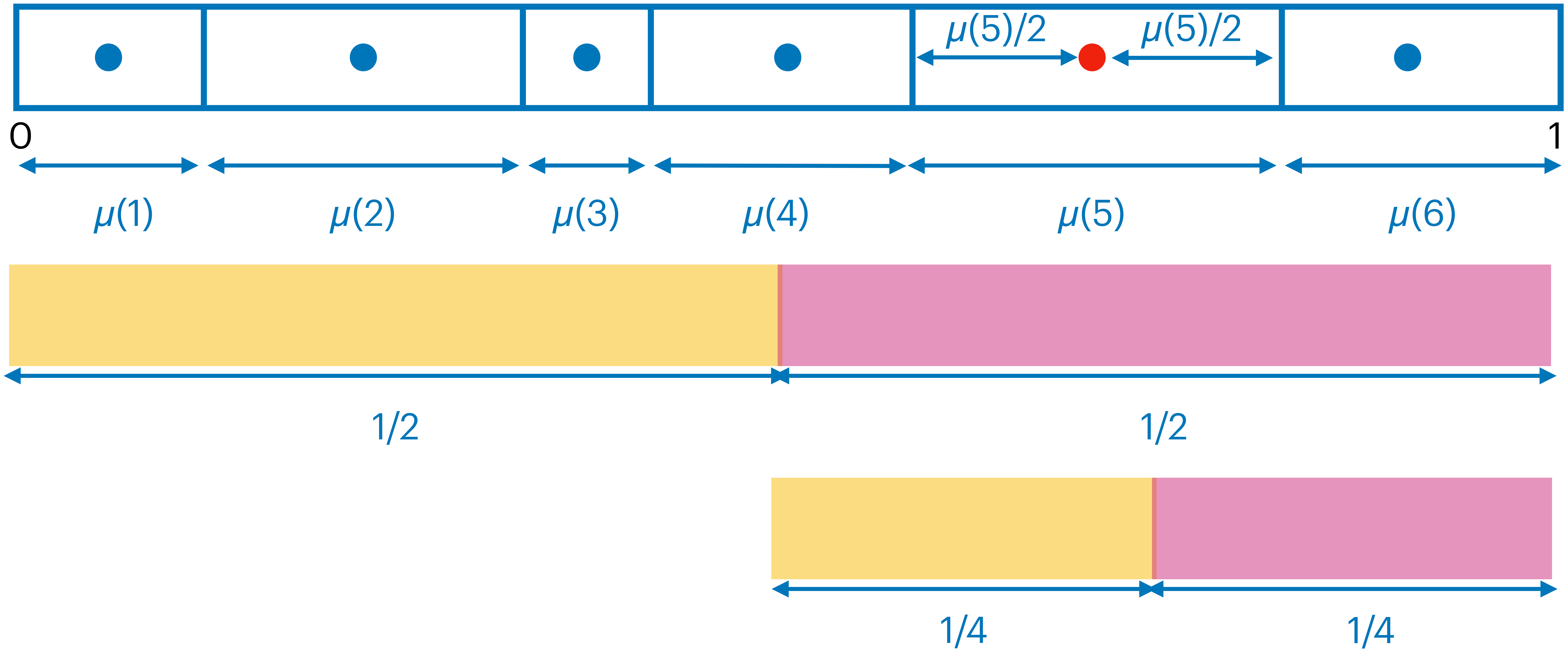
Analysis of Gilbert–Moore



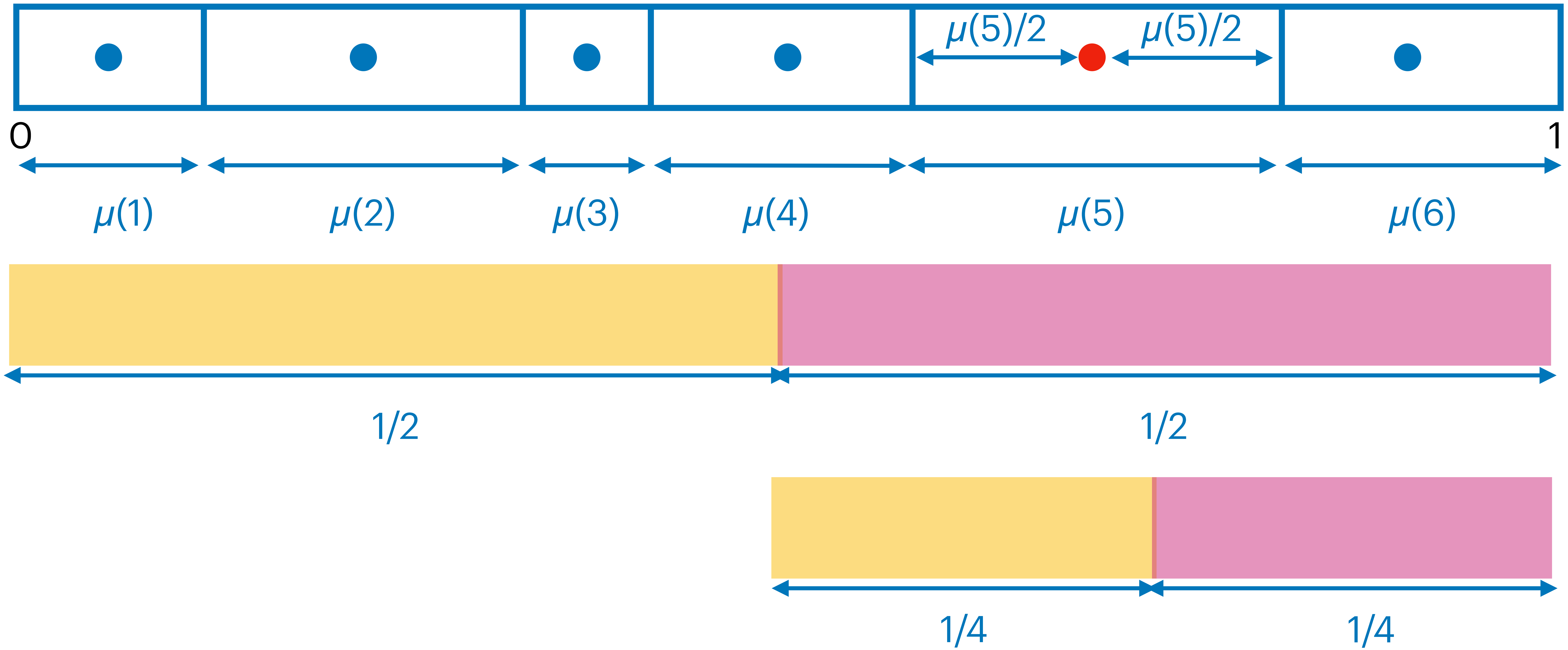
Analysis of Gilbert–Moore



Analysis of Gilbert–Moore

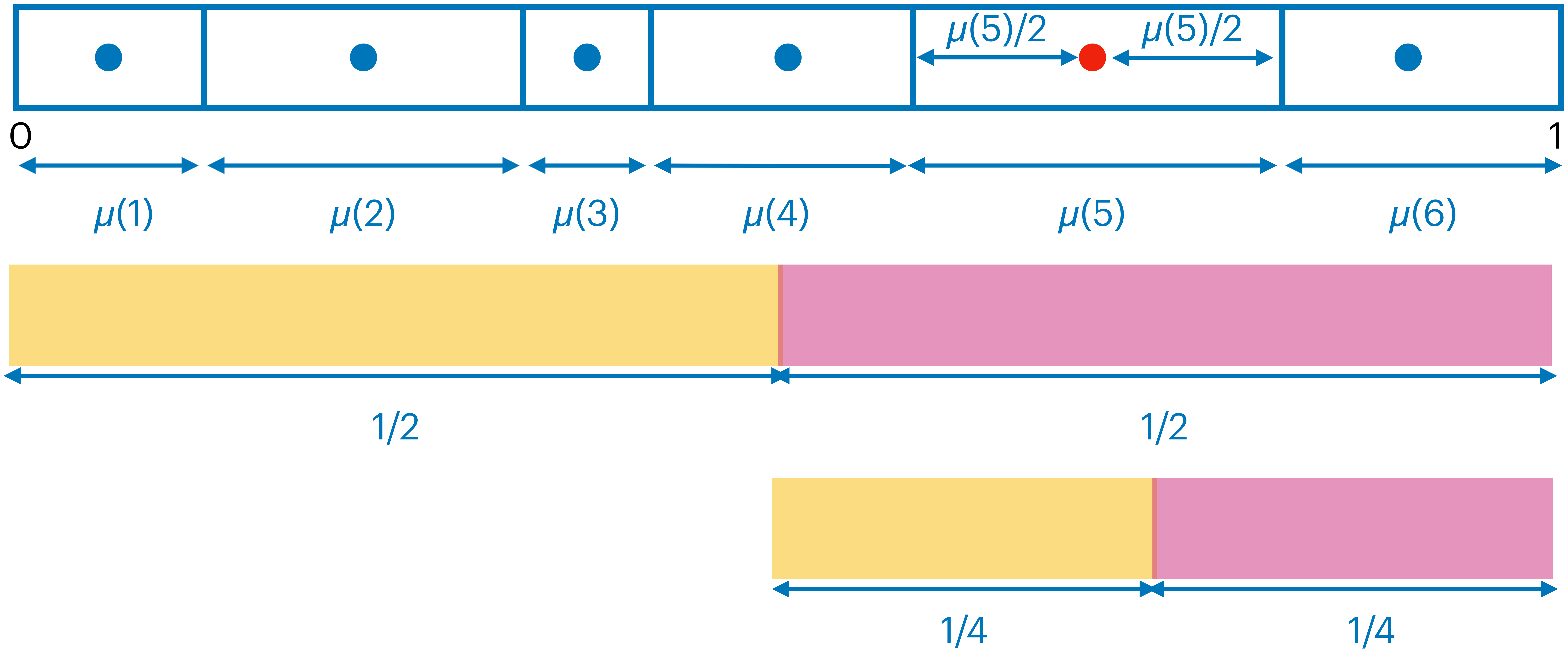


Analysis of Gilbert–Moore



After k questions, zero in on interval of length 2^{-k}

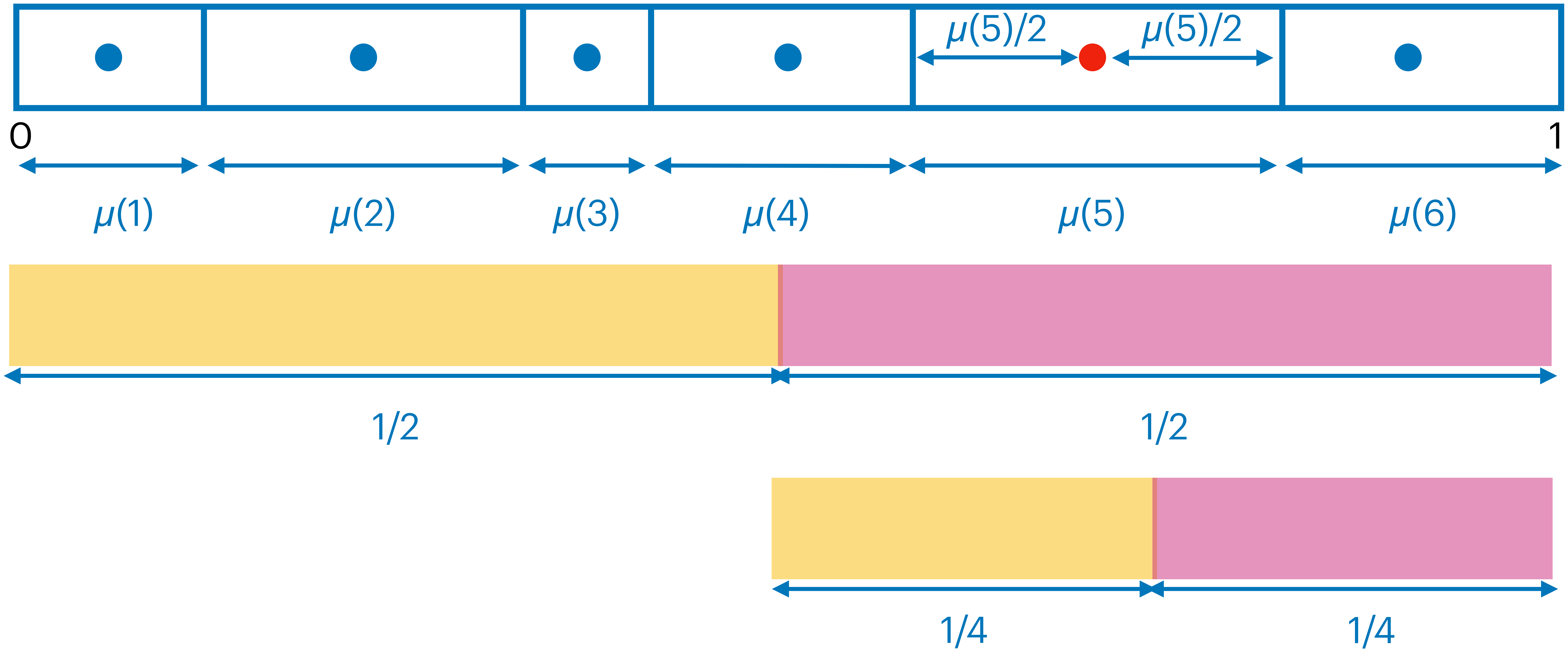
Analysis of Gilbert–Moore



After k questions, zero in on interval of length 2^{-k}

Can stop once interval has length at most $\mu(x)/2$

Analysis of Gilbert–Moore

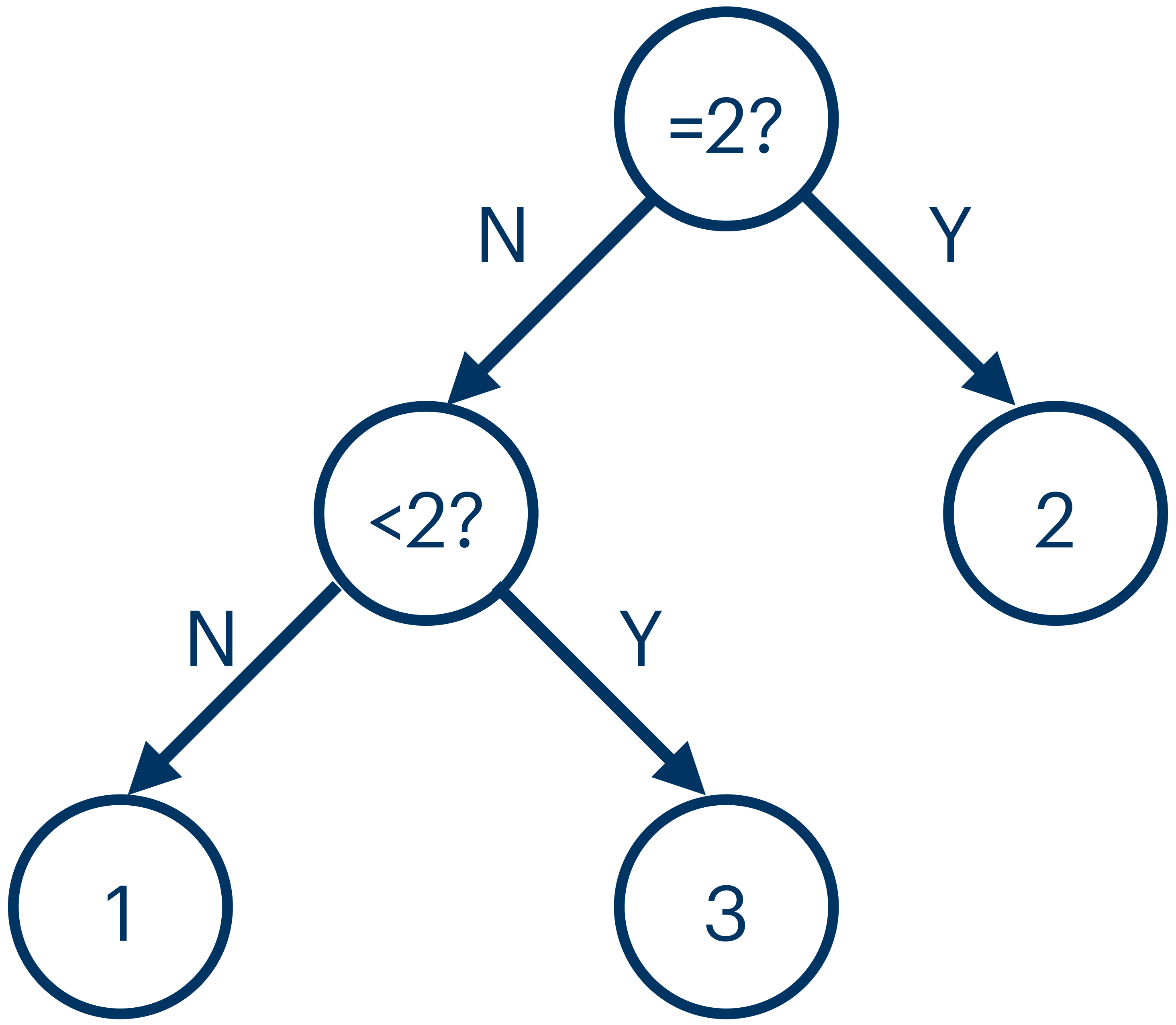


After k questions, zero in on interval of length 2^{-k}

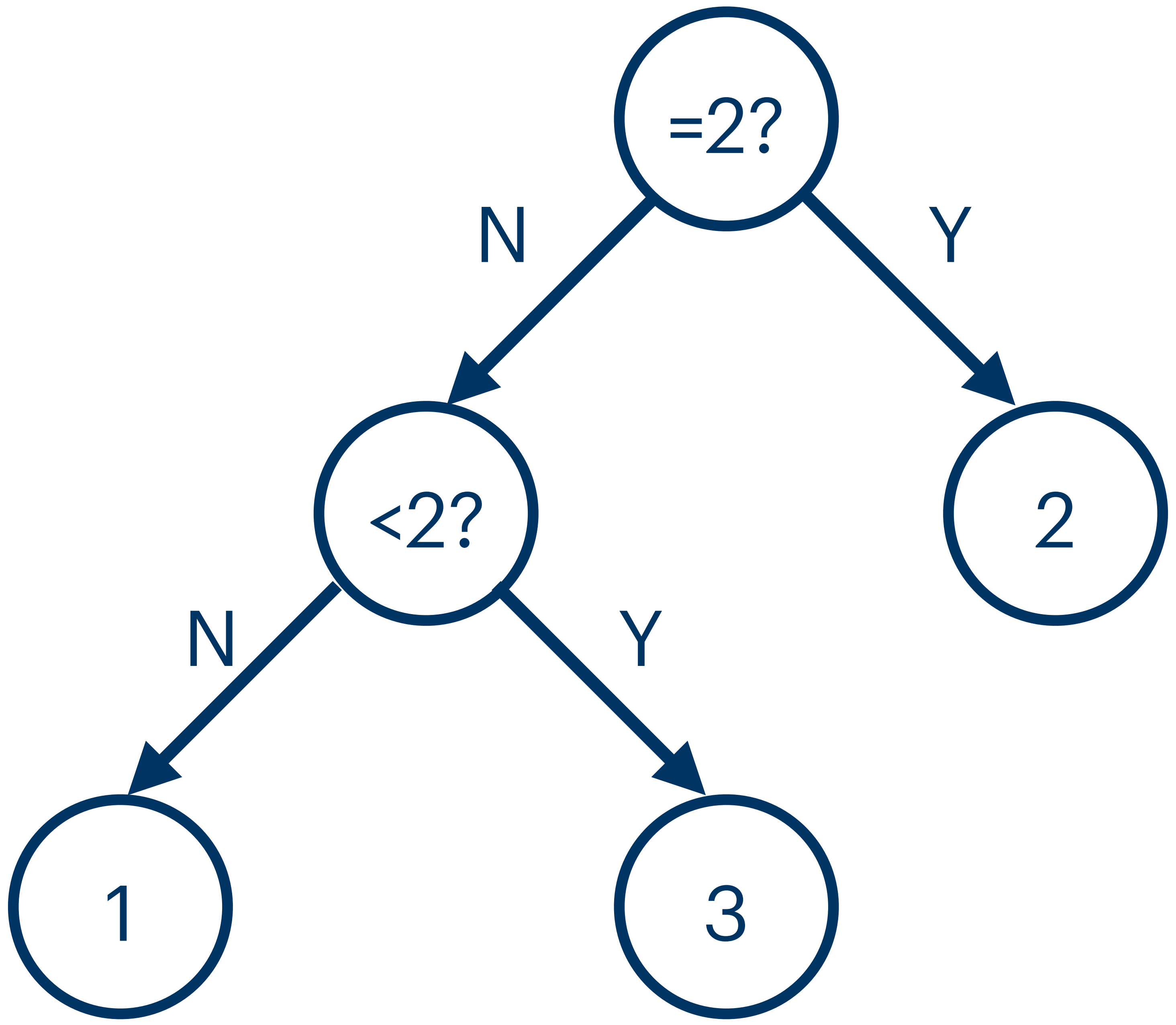
Can stop once interval has length at most $\mu(x)/2$

Stop after $\lceil \log(2/\mu(x)) \rceil < \log(1/\mu(x))+2$ questions

Binary Split Trees

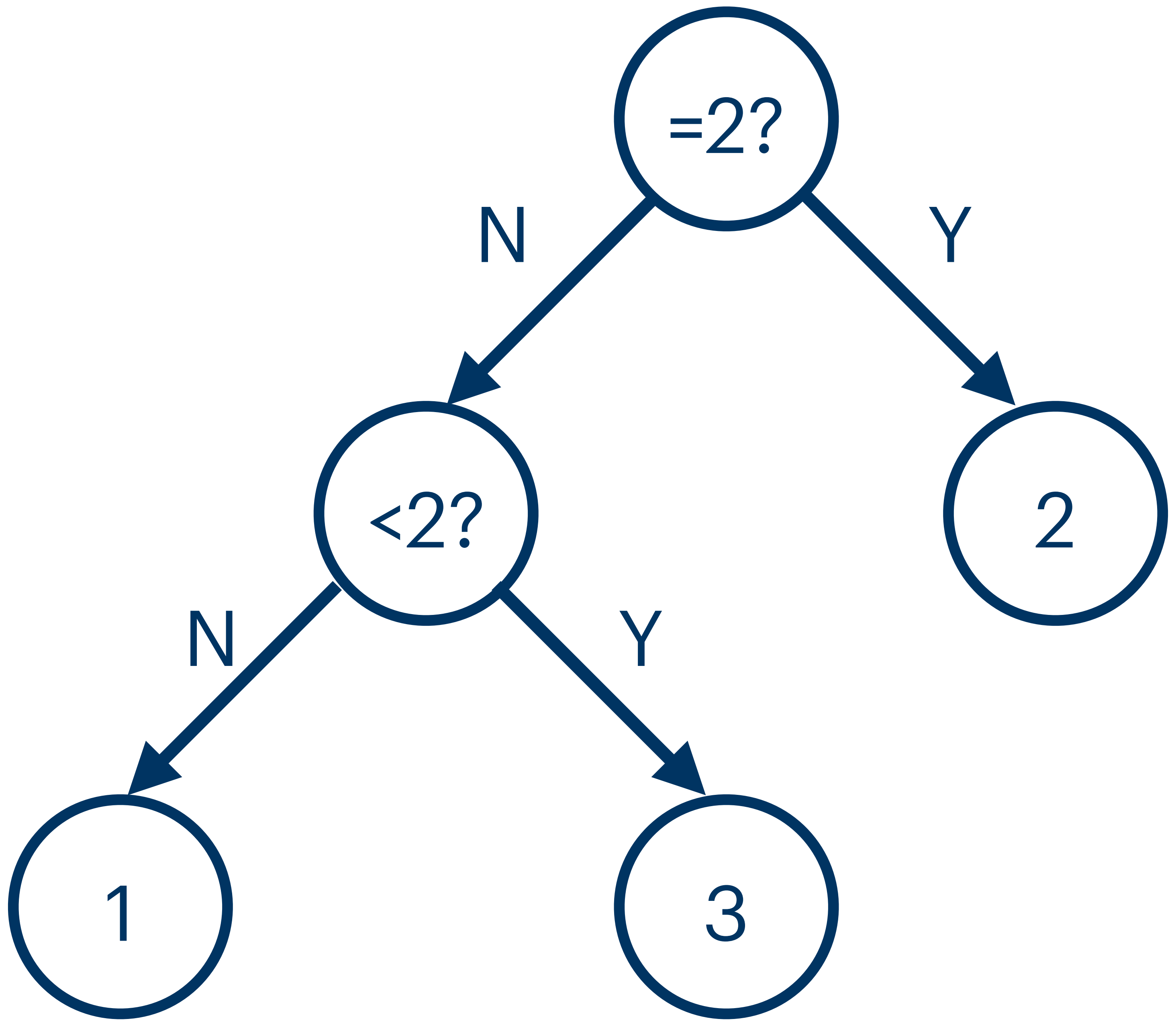


Binary Split Trees



We show: optimal binary split tree achieves H(μ)+1

Binary Split Trees



We show: optimal binary split tree achieves H(μ)+1

Same performance guarantee as Huffman!

Our Algorithm

Our Algorithm

If most probable element *i* has probability ≥0.3:

Ask if $x = i$

Otherwise:

Ask most informative “<” question

Our Algorithm

If most probable element i has probability ≥0.3:

  Ask if $x = i$

Otherwise:

  Ask most informative “<” question

Our Algorithm

If most probable element *i* has probability ≥0.3:

  Ask if $x = i$

Otherwise:

  Ask most informative "<" question

1	2	3

Our Algorithm

If most probable element *i* has probability ≥0.3:

  Ask if $x = i$

Otherwise:

  Ask most informative “<” question

Why do we care?

Chunked Binary Split Trees

word 0	word 1	word 2	word 3

Chunked Binary Split Trees

word 0	word 1	word 2	word 3
=AB01?	<BDBB?	=0010?	<0042?
<C0A1?			

Chunked Binary Split Trees

word 0	word 1	word 2	word 3
=AB01?	<BDBB?	=0010?	<0042?
<C0A1?			

Performance on *w* words: H(μ)+*w*

Chunked Binary Split Trees

word 0	word 1	word 2	word 3
=AB01?	<BDBB?	=0010?	<0042?
<C0A1?			

Performance on w words: $H(\mu)+w$

Number of different questions: $2wn^{1/w}$

Chunked Binary Split Trees

word 0	word 1	word 2	word 3
=AB01?	<BDBB?	=0010?	<0042?
<C0A1?			

Performance on w words: $H(\mu)+w$

Number of different questions: $2wn^{1/w}$

Optimal for redundancy w!

Playing 20 Questions with a Liar

Playing 20 Questions with a Liar

Bob

Thinks of a number
between 1 to n

Playing 20 Questions with a Liar

Alice

Finds the number by
asking Yes/No questions

Bob

Thinks of a number
between 1 to n

Playing 20 Questions with a Liar

Alice

Finds the number by asking Yes/No questions

Bob

Thinks of a number between 1 to n

Bob allowed to lie k times

Playing 20 Questions with a Liar

Alice

Finds the number by
asking Yes/No questions

optimal cost:
log*n*+*k*loglog*n*

Bob

Thinks of a number
between 1 to *n*

Bob allowed to lie k times

Distributional 20 Questions with a Liar

Distributional 20 Questions with a Liar

Bob

Samples a number
between 1 to n
according to μ

Distributional 20 Questions with a Liar

Alice

Finds the number by asking Yes/No questions

Bob

Samples a number between 1 to n **according to μ**

Distributional 20 Questions with a Liar

Alice

Finds the number by asking Yes/No questions

Bob

Samples a number between 1 to n **according to μ**

Bob allowed to lie k times

Distributional 20 Questions with a Liar

Alice

Finds the number by asking Yes/No questions

Optimal cost: $H(\mu)+kH_2(\mu)$ on average

Bob

Samples a number between 1 to n **according to μ**

Bob allowed to lie k times

Distributional 20 Questions with a Liar

Alice

$H(\mu) = E[\log 1/\mu]$
$H_2(\mu) = E[\log\log 1/\mu]$

Bob

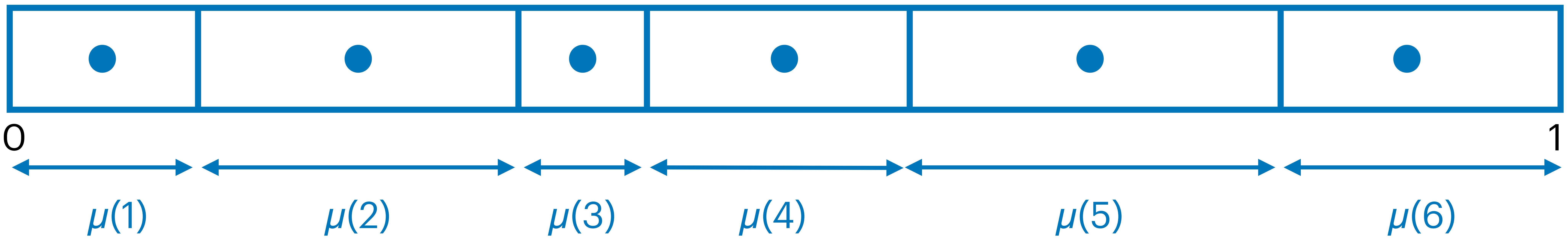
Optimal cost:
$H(\mu)+kH_2(\mu)$
on average

Finds the number by asking Yes/No questions

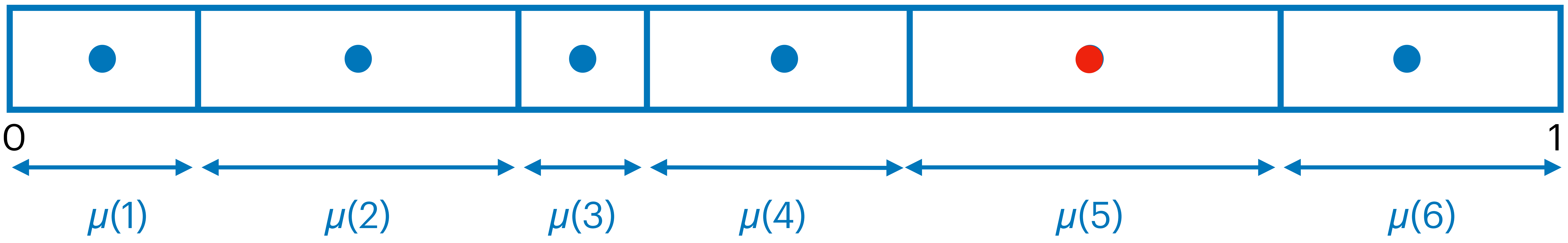
Samples a number between 1 to n **according to μ**

Bob allowed to lie k times

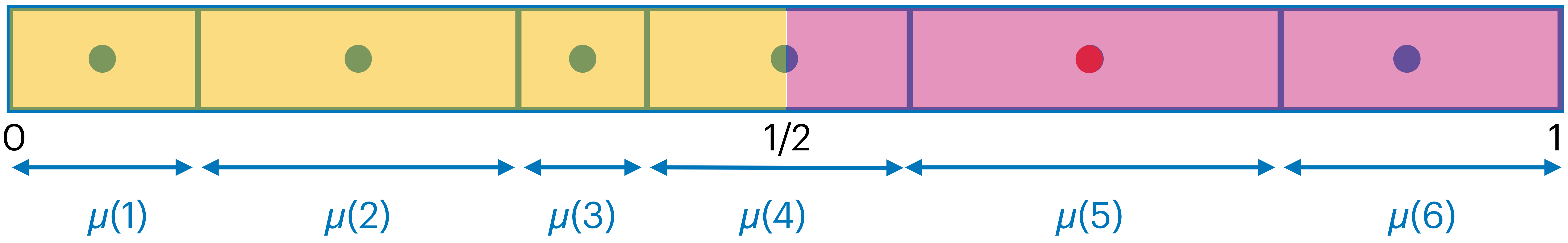
Gilbert–Moore with Lies



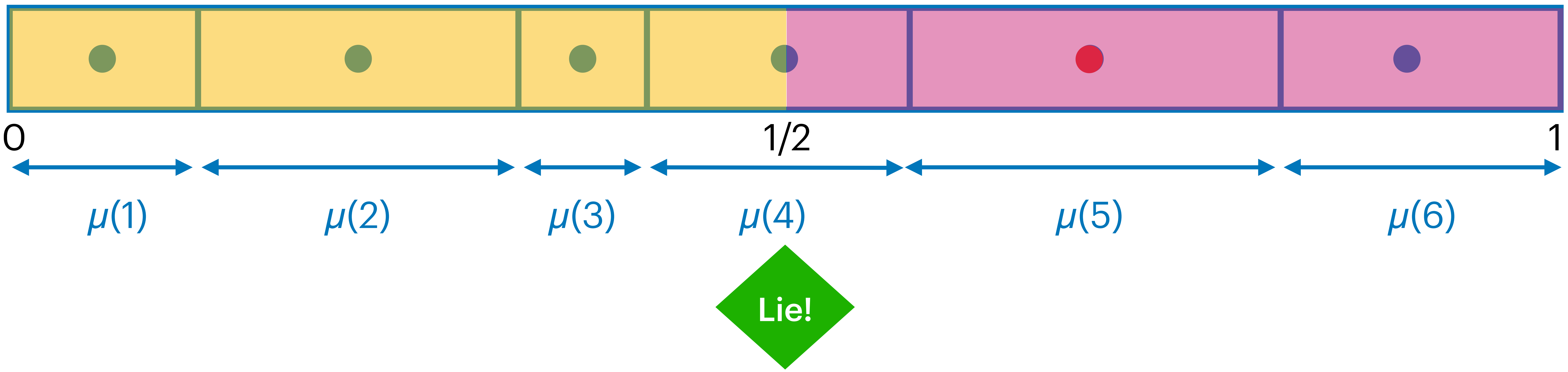
Gilbert–Moore with Lies



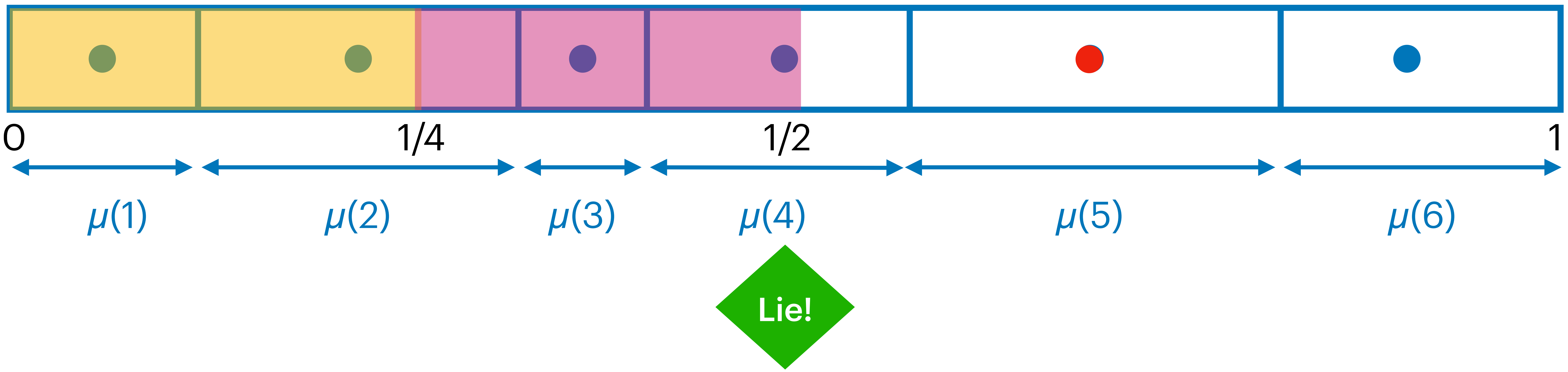
Gilbert–Moore with Lies



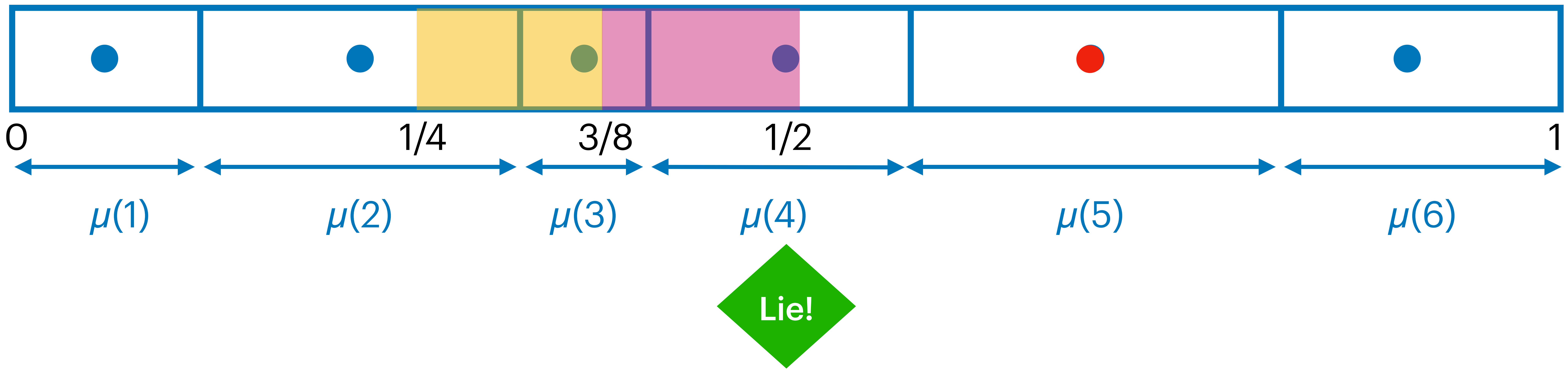
Gilbert–Moore with Lies



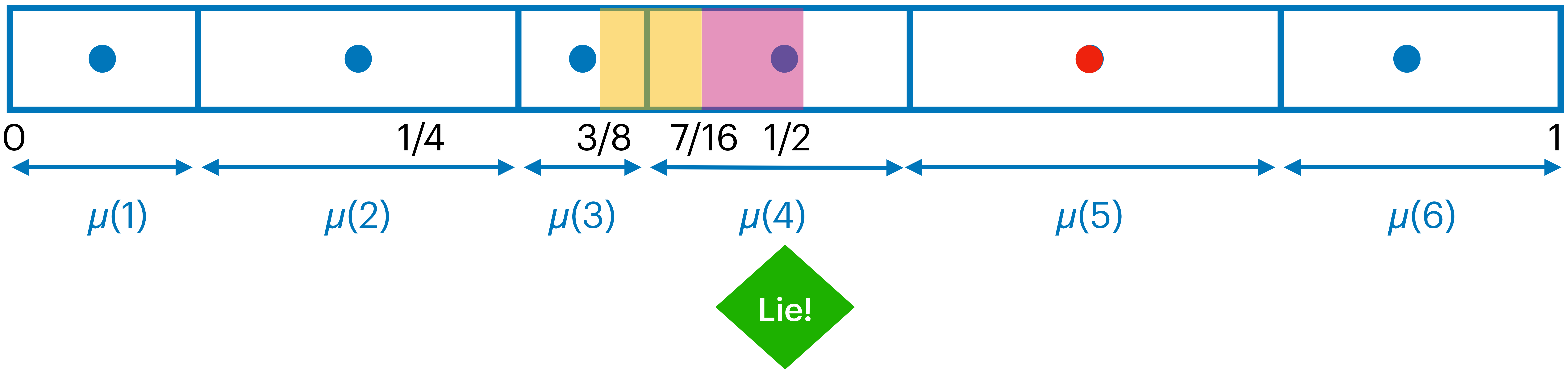
Gilbert–Moore with Lies



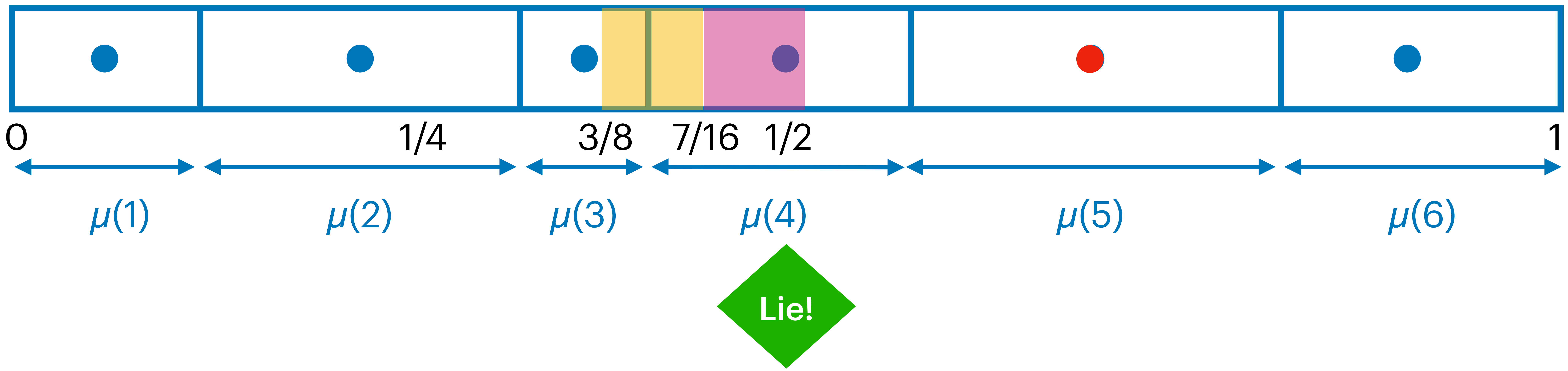
Gilbert–Moore with Lies



Gilbert–Moore with Lies

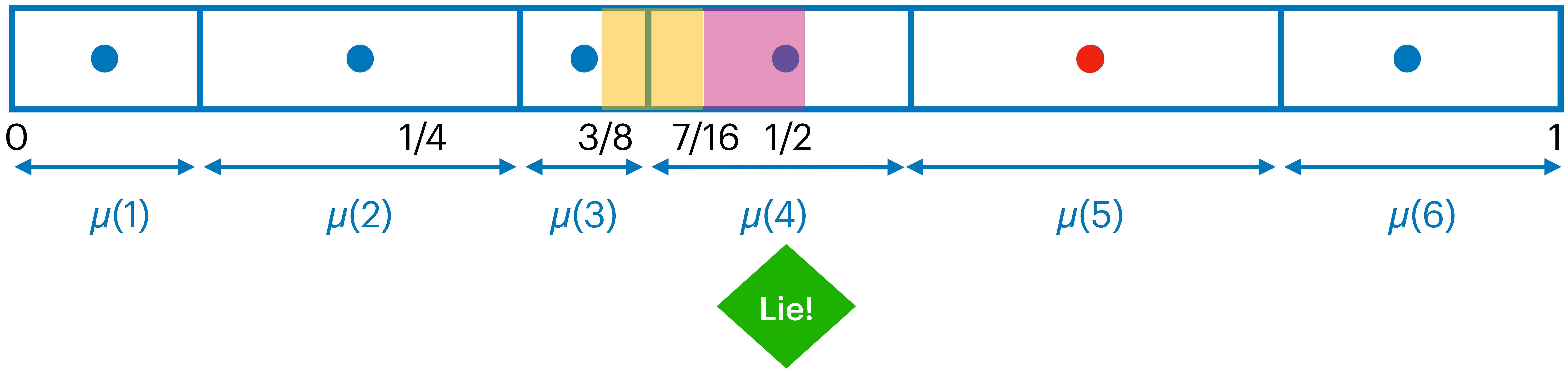


Gilbert–Moore with Lies



After first lie, answer always “>” – suspicious!

Gilbert–Moore with Lies



After first lie, answer always “>” – suspicious!

Figure out true answer, possibly rollback

Why $kH_2(\mu)$ is right overhead?

$H(\mu) = E[\log 1/\mu]$ $H_2(\mu) = E[\log\log 1/\mu]$

Why $k\mathrm{H}_2(\mu)$ is right overhead?

$\mathrm{H}(\mu) = \mathrm{E}[\log 1/\mu]$ $\mathrm{H}_2(\mu) = \mathrm{E}[\log\log 1/\mu]$

Lower bound:

Why $kH_2(\mu)$ is right overhead?

$H(\mu) = E[\log 1/\mu]$ $H_2(\mu) = E[\log\log 1/\mu]$

Lower bound:

At end of game, Alice knows both x and positions where Bob lied

Why $kH_2(\mu)$ is right overhead?

$H(\mu) = E[\log 1/\mu]$ $\quad\quad$ $H_2(\mu) = E[\log\log 1/\mu]$

Lower bound:

At end of game, Alice knows both x and positions where Bob lied

Game lasts for $\approx\log(1/\mu(x))$ rounds

Why $kH_2(\mu)$ is right overhead?

$H(\mu) = E[\log 1/\mu]$ **$H_2(\mu) = E[\log\log 1/\mu]$**

Lower bound:

- At end of game, Alice knows both x and positions where Bob lied
- Game lasts for $\approx\log(1/\mu(x))$ rounds
- Each lie position requires Alice to find $\log\log(1/\mu(x))$ more bits

Why $k\mathrm{H}_2(\mu)$ is right overhead?

$\mathrm{H}(\mu) = \mathrm{E}[\log 1/\mu]$ $\mathrm{H}_2(\mu) = \mathrm{E}[\log\log 1/\mu]$

Lower bound:

At end of game, Alice knows both x and positions where Bob lied

Game lasts for $\approx\log(1/\mu(x))$ rounds

Each lie position requires Alice to find $\log\log(1/\mu(x))$ more bits

Upper bound:

Why $kH_2(\mu)$ is right overhead?

$H(\mu) = E[\log 1/\mu]$ $\qquad$ $H_2(\mu) = E[\log\log 1/\mu]$

Lower bound:

At end of game, Alice knows both x and positions where Bob lied

Game lasts for $\approx\log(1/\mu(x))$ rounds

Each lie position requires Alice to find $\log\log(1/\mu(x))$ more bits

Upper bound:

Length of suspicion interval balances "false positive" and overhead

Why $kH_2(\mu)$ is right overhead?

$H(\mu) = E[\log 1/\mu]$ **$H_2(\mu) = E[\log\log 1/\mu]$**

Lower bound:

At end of game, Alice knows both x and positions where Bob lied

Game lasts for $\approx\log(1/\mu(x))$ rounds

Each lie position requires Alice to find $\log\log(1/\mu(x))$ more bits

Upper bound:

Length of suspicion interval balances “false positive” and overhead

Optimal choice turns out to be $\log(\text{depth})\approx\log\log(1/\mu(x))$

Why $kH_2(\mu)$ is right overhead?

$H(\mu) = E[\log 1/\mu]$ $H_2(\mu) = E[\log\log 1/\mu]$

Lower bound:

At end of game, Alice knows both x and positions where Bob lied

Game lasts for $\approx\log(1/\mu(x))$ rounds

Each lie position requires Alice to find $\log\log(1/\mu(x))$ more bits

Upper bound:

Length of suspicion interval balances “false positive” and overhead

Optimal choice turns out to be log(depth)$\approx\log\log(1/\mu(x))$

Cost incurred once per lie

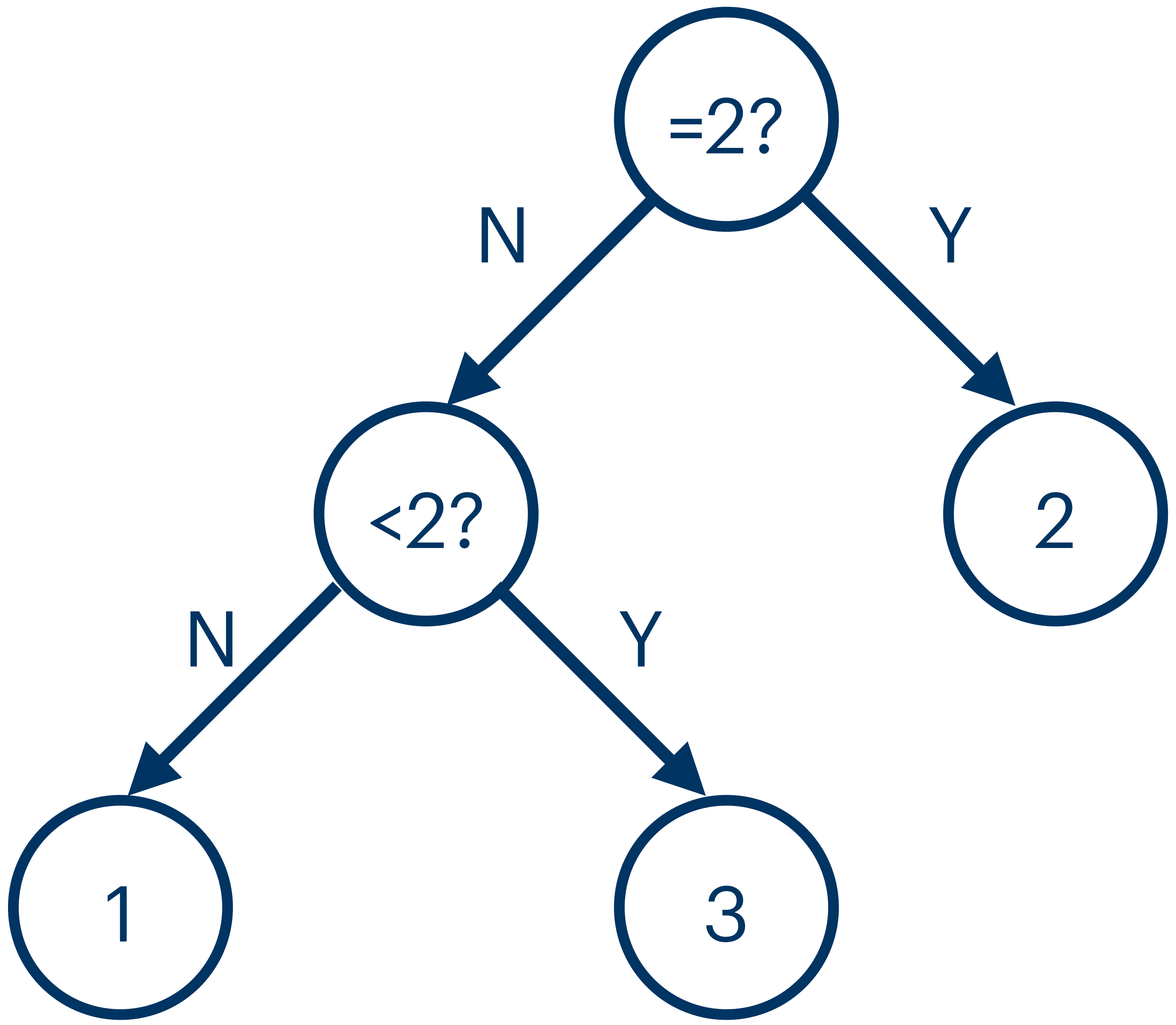
Matching Huffman’s algorithm exactly

Matching Huffman's algorithm exactly

Can we match Huffman *exactly* using
a subset of all possible questions?

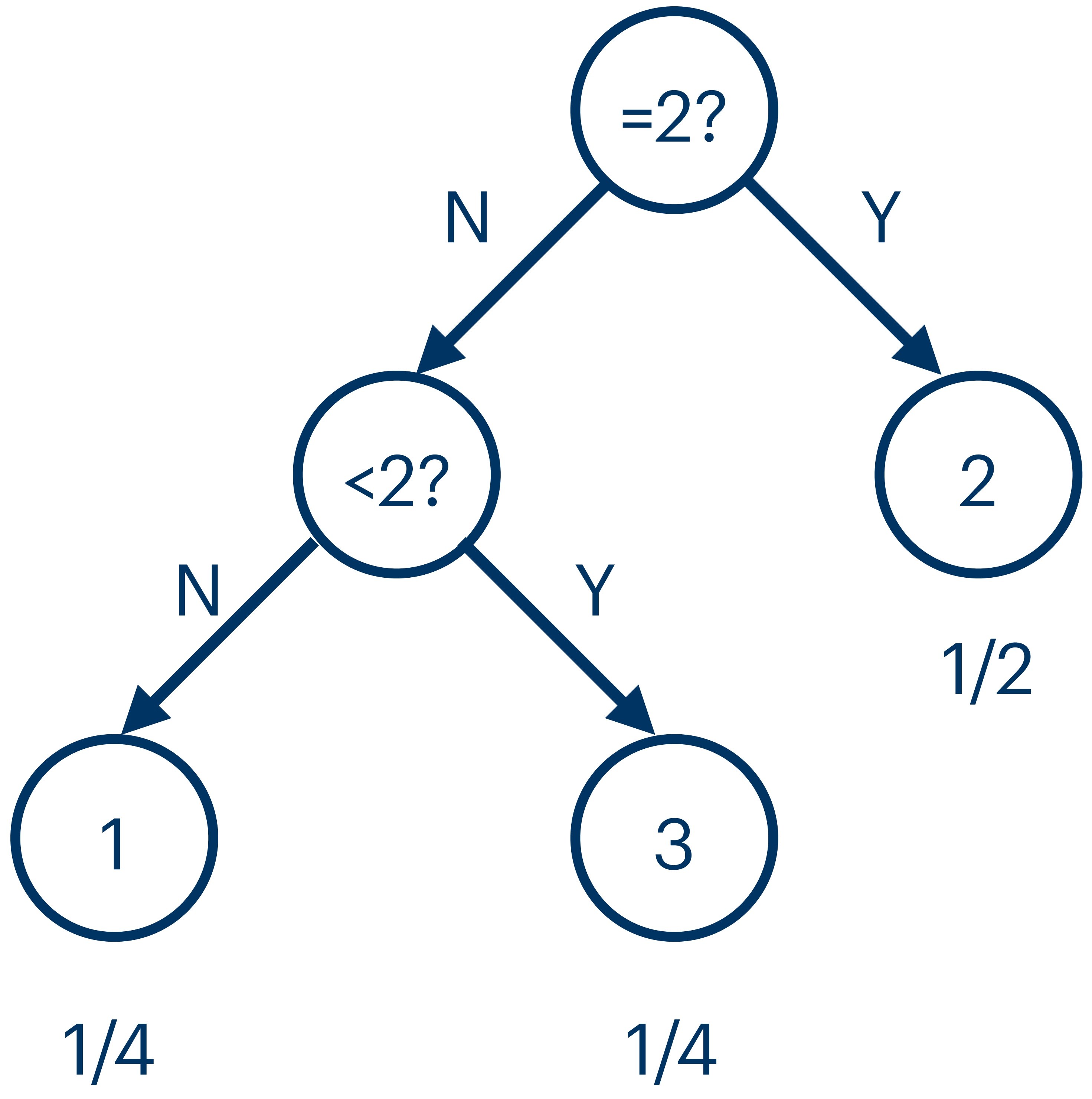
Matching Huffman's algorithm exactly

Can we match Huffman *exactly* using a subset of all possible questions?



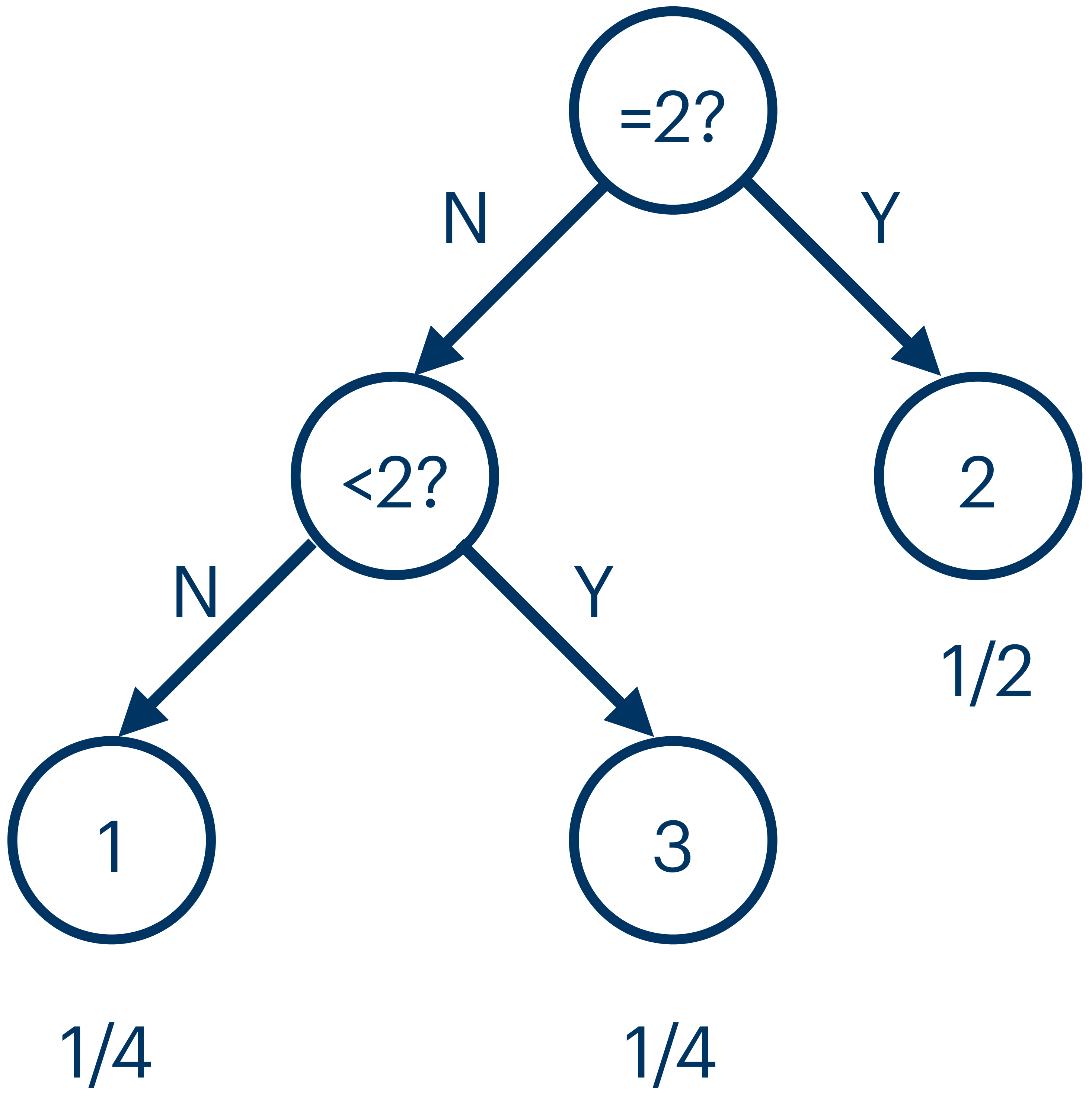
Matching Huffman's algorithm exactly

Can we match Huffman *exactly* using a subset of all possible questions?



Matching Huffman's algorithm exactly

Can we match Huffman *exactly* using a subset of all possible questions?



Enough to handle "dyadic" distributions

Matching Huffman's algorithm exactly

Enough to show:
Each dyadic distribution μ has a strategy
using $H(\mu)$ questions in expectation

Matching Huffman's algorithm exactly

Enough to show:
Each dyadic distribution μ has a strategy using $H(\mu)$ questions in expectation

Equivalently:
Can always find question splitting μ evenly

Matching Huffman's algorithm exactly

Enough to show:
Each dyadic distribution μ has a strategy using $H(\mu)$ questions in expectation

Equivalently:
Can always find question splitting μ evenly

1/4	1/16	1/8	1/8	1/16	1/16	1/16	1/4
$\mu(1)$	$\mu(2)$	$\mu(3)$	$\mu(4)$	$\mu(5)$	$\mu(6)$	$\mu(7)$	$\mu(8)$

Matching Huffman's algorithm exactly

Enough to show:
Each dyadic distribution μ has a strategy
using H(μ) questions in expectation

Equivalently:
Can always find question splitting μ evenly

1/4	1/16	1/8	1/8	1/16	1/16	1/16	1/4
$\mu(1)$	$\mu(2)$	$\mu(3)$	$\mu(4)$	$\mu(5)$	$\mu(6)$	$\mu(7)$	$\mu(8)$

Matching Huffman's algorithm exactly

Goal: Can always find question splitting μ evenly

1/4	1/16	1/8	1/8	1/16	1/16	1/16	1/4
$\mu(1)$	$\mu(2)$	$\mu(3)$	$\mu(4)$	$\mu(5)$	$\mu(6)$	$\mu(7)$	$\mu(8)$

Matching Huffman's algorithm exactly

Goal: Can always find question splitting μ evenly

1/4	1/16	1/8	1/8	1/16	1/16	1/16	1/4
$\mu(1)$	$\mu(2)$	$\mu(3)$	$\mu(4)$	$\mu(5)$	$\mu(6)$	$\mu(7)$	$\mu(8)$

Example: all subsets of $\{1,\ldots,n/2\}$ + all subsets of $\{n/2+1,\ldots,n\}$

Matching Huffman's algorithm exactly

Goal: Can always find question splitting μ evenly

1/4	1/16	1/8	1/8	1/16	1/16	1/16	1/4
$\mu(1)$	$\mu(2)$	$\mu(3)$	$\mu(4)$	$\mu(5)$	$\mu(6)$	$\mu(7)$	$\mu(8)$

Example: all subsets of $\{1,\ldots,n/2\}$ + all subsets of $\{n/2+1,\ldots,n\}$

Either $\mu(\{1,\ldots,n/2\}) \geq 1/2$ or $\mu(\{n/2+1,\ldots,n\}) \geq 1/2$, say the former

Matching Huffman's algorithm exactly

Goal: Can always find question splitting μ evenly

1/4	1/16	1/8	1/8	1/16	1/16	1/16	1/4
$\mu(1)$	$\mu(2)$	$\mu(3)$	$\mu(4)$	$\mu(5)$	$\mu(6)$	$\mu(7)$	$\mu(8)$

Example: all subsets of $\{1,\ldots,n/2\}$ + all subsets of $\{n/2+1,\ldots,n\}$

Either $\mu(\{1,\ldots,n/2\}) \geq 1/2$ or $\mu(\{n/2+1,\ldots,n\}) \geq 1/2$, say the former

Arrange elements in non-increasing order of probability

1/4	1/8	1/8	1/16
$\mu(1)$	$\mu(3)$	$\mu(4)$	$\mu(2)$

Matching Huffman's algorithm exactly

Goal: Can always find question splitting μ evenly

1/4	1/16	1/8	1/8	1/16	1/16	1/16	1/4
$\mu(1)$	$\mu(2)$	$\mu(3)$	$\mu(4)$	$\mu(5)$	$\mu(6)$	$\mu(7)$	$\mu(8)$

Example: all subsets of $\{1,\ldots,n/2\}$ + all subsets of $\{n/2+1,\ldots,n\}$

Either $\mu(\{1,\ldots,n/2\}) \geq 1/2$ or $\mu(\{n/2+1,\ldots,n\}) \geq 1/2$, say the former

Arrange elements in non-increasing order of probability

1/4	1/8	1/8	1/16
$\mu(1)$	$\mu(3)$	$\mu(4)$	$\mu(2)$

Some prefix sums to exactly 1/2

Matching Huffman's algorithm exactly

Example: all subsets of $\{1,\ldots,n/2\}$ + all subsets of $\{n/2+1,\ldots,n\}$

Matching Huffman's algorithm exactly

Example: all subsets of $\{1,\ldots,n/2\}$ + all subsets of $\{n/2+1,\ldots,n\}$

Size: 1.4142^n, best known explicit construction

Matching Huffman's algorithm exactly

Example: all subsets of $\{1,\ldots,n/2\}$ + all subsets of $\{n/2+1,\ldots,n\}$

Size: 1.4142^n, best known explicit construction

Random construction gives 1.25^n, which is optimal!

Matching Huffman's algorithm exactly

Example: all subsets of $\{1,\ldots,n/2\}$ + all subsets of $\{n/2+1,\ldots,n\}$

Size: 1.4142^n, best known explicit construction

Random construction gives 1.25^n, which is optimal!

Construction: choose 1.25^n random sets of every size

Matching Huffman's algorithm exactly

Example: all subsets of $\{1,\ldots,n/2\}$ + all subsets of $\{n/2+1,\ldots,n\}$

Size: 1.4142^n, best known explicit construction

Random construction gives 1.25^n, which is optimal!

Construction: choose 1.25^n random sets of every size

Optimal number of questions for Huffman + ε: $n^{O(1/\varepsilon)}$

What about smooth distributions?

What about smooth distributions?

Huffman worst case $H(\mu)+1$ only obtained when μ almost constant

What about smooth distributions?

Huffman worst case $H(\mu)+1$ only obtained when μ almost constant

What happens when all probabilities in μ are small?

What about smooth distributions?

Huffman worst case $H(\mu)+1$ only obtained when μ almost constant

What happens when all probabilities in μ are small?

Gallager: cannot go below $H(\mu)+0.086$

What about smooth distributions?

Huffman worst case $H(\mu)+1$ only obtained when μ almost constant

What happens when all probabilities in μ are small?

Gallager: cannot go below $H(\mu)+0.086$

Achieved for uniform distributions!

What about smooth distributions?

Huffman worst case $H(\mu)+1$ only obtained when μ almost constant

What happens when all probabilities in μ are small?

Gallager: cannot go below $H(\mu)+0.086$

Achieved for uniform distributions!

What do we get with “<“ questions? With “<“ and “=“ questions?

What about smooth distributions?

Huffman worst case $H(\mu)+1$ only obtained when μ almost constant

What happens when all probabilities in μ are small?

Gallager: cannot go below $H(\mu)+0.086$

Achieved for uniform distributions!

What do we get with “<“ questions? With “<“ and “=“ questions?

“<“ questions: $H(\mu)+1.086$ [Nakatsu]

What about smooth distributions?

Huffman worst case H(μ)+1 only obtained when μ almost constant

What happens when all probabilities in μ are small?

Gallager: cannot go below H(μ)+0.086

Achieved for uniform distributions!

What do we get with “<“ questions? With “<“ and “=“ questions?

“<“ questions: H(μ)+1.086 [Nakatsu]

“<“ and “=“ questions: between H(μ)+0.501 and H(μ)+0.586

Many open questions

Many open questions

- Questions with d>2 answers? Mehalel: $1.25 \to 1 + (d-1)/d^{d/(d-1)}$

Many open questions

- Questions with $d>2$ answers? Mehalel: $1.25 \to 1 + (d-1)/d^{d/(d-1)}$
- Other models of errors? At most p fraction, at most q fraction in every prefix

Many open questions

- Questions with d>2 answers? Mehalel: $1.25 \to 1 + (d-1)/d^{d/(d-1)}$
- Other models of errors? At most p fraction, at most q fraction in every prefix
- What happens if we limit worst-case number of questions?

Many open questions

- Questions with $d>2$ answers? Mehalel: $1.25 \to 1 + (d-1)/d^{d/(d-1)}$
- Other models of errors? At most p fraction, at most q fraction in every prefix
- What happens if we limit worst-case number of questions?
- Fast algorithms for finding optimal binary split trees?

Many open questions

- Questions with d>2 answers? Mehalel: 1.25 → $1 + (d-1)/d^{d/(d-1)}$
- Other models of errors? At most p fraction, at most q fraction in every prefix
- What happens if we limit worst-case number of questions?
- Fast algorithms for finding optimal binary split trees?

Thank You!