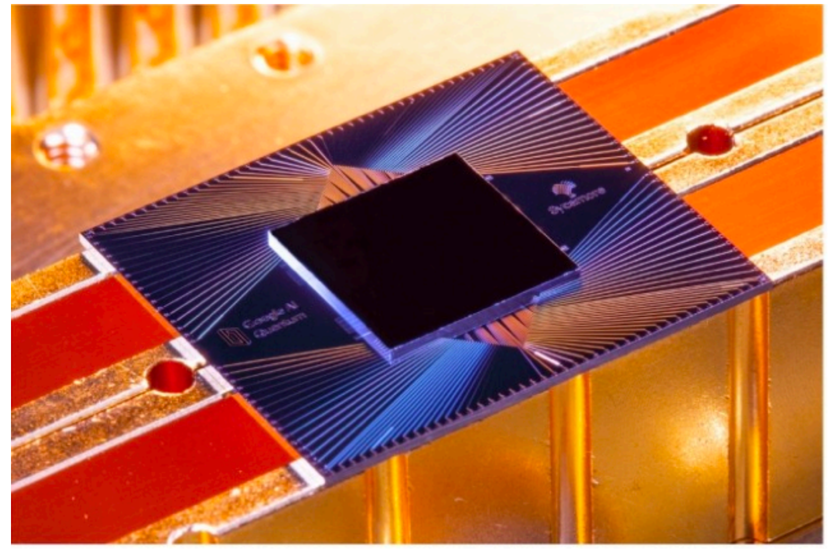
# Quantum Time-Space Tradeoffs by Recording Queries

Yassine Hamoudi, Frédéric Magniez

IRIF, Université de Paris

arXiv: 2002.08944

## Google Sycamore's calculation



**Time** ≈ 5 minutes
**Memory** ≈ 53 qubits
+ few megabytes

## Simulation by Schrödinger-Feynman algorithm


**Time** ≈ 10,000 years
**Memory** ≈ 1 Petabyte

## Simulation by Schrödinger algorithm


**Time** ≈ 2.5 days
**Memory** ≈ 250 Petabytes

1. Time and Space in the Query Model

2. The Collision Pairs Finding Problem

3. Lower Bounds by Recording Queries

1

# Time and Space in the Query Model

# Classical Query Model

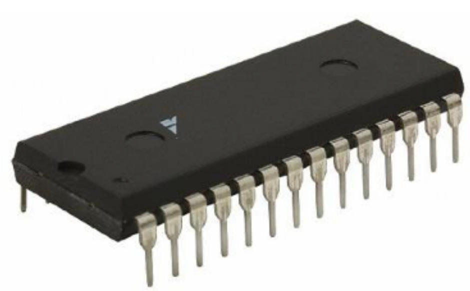


*Input*

*Queries*

*Computer*

$x = (x_1, \ldots, x_N)$



Read Only Memory

$i$

$x_i$

Read-Write Memory

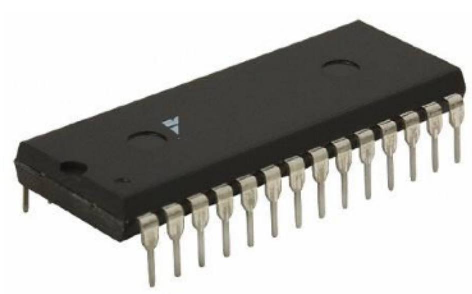
# Classical Query Model

*Input*

*Queries*

*Computer*

$x = (x_1, \ldots, x_N)$



Read Only Memory

$\xleftarrow{\; i \;}$

$\xrightarrow[x_i]{}$

Read-Write Memory

**Time T** = number of <u>queries</u> to the input

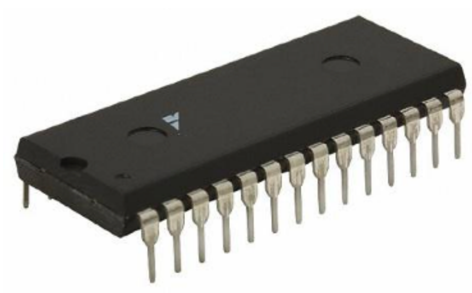
**Space S** = number of <u>bits</u> in the computer's memory

# Classical Query Model



*Input* | *Queries* | *Computer*

$x = (x_1, \ldots, x_N)$



Read Only Memory

$\xleftarrow{\;i\;}$

$\xrightarrow[\;x_i\;]{}$

Read-Write Memory

**Time T** = number of <u>queries</u> to the input

**Space S** = number of <u>bits</u> in the computer's memory

⟶ The number of queries is a lower bound on the actual computation time.

⟶ If S = ∞ then T ≤ N is sufficient (load the entire input in the computer's memory).

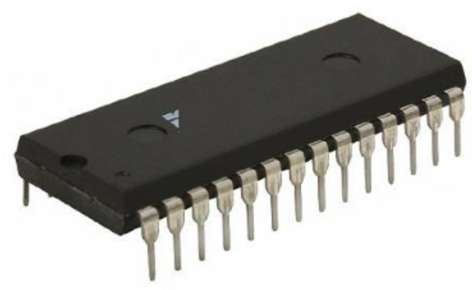
⟶ We are interested in the case "T or S << N".

# Classical Query Model



*Input* | *Queries* | *Computer*

$x = (x_1, \ldots, x_N)$



Read Only Memory

$i$ ←

→ $x_i$

Read-Write Memory

**Time T** = number of <u>queries</u> to the input

**Space S** = number of <u>bits</u> in the computer's memory

**Time-Space Tradeoffs:**

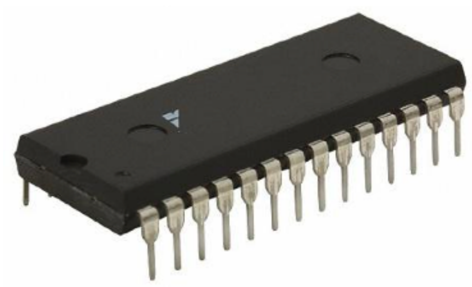
*[Beame'91]* Sorting N numbers requires time T and space S such that **$TS \geq \Omega(N^2)$**.

# Classical Query Model



*Input* | *Queries* | *Computer*

$x = (x_1, \ldots, x_N)$



Read Only Memory

$\xleftarrow{\;i\;}$

$\xrightarrow[x_i]{}$


Read-Write Memory

**Time T** = number of <u>queries</u> to the input

**Space S** = number of <u>bits</u> in the computer's memory

## Time-Space Tradeoffs:

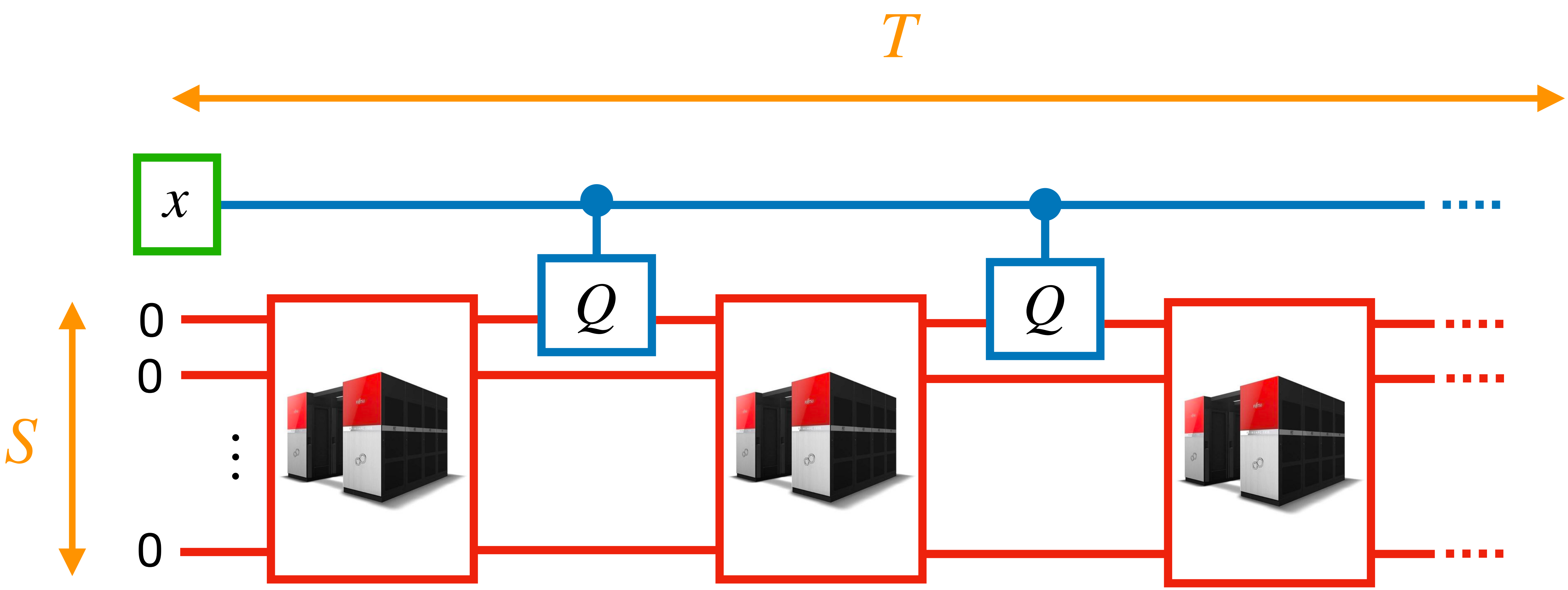
*[Beame'91]* Sorting N numbers requires time T and space S such that **TS ≥ Ω(N²)**.

*[Klauck et al.'07]* Boolean Multiplication of two NxN matrices requires **TS ≥ Ω(N³)**.

…

# Classical Query Model

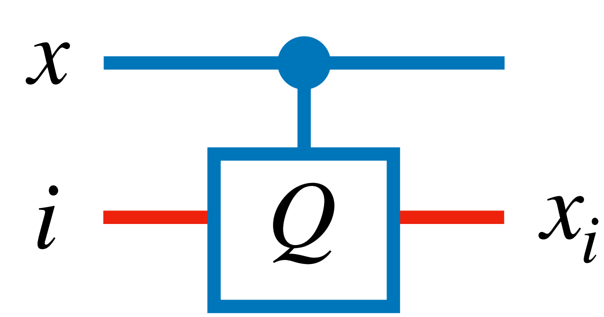





- Initially, the memory is filled with S zeros.

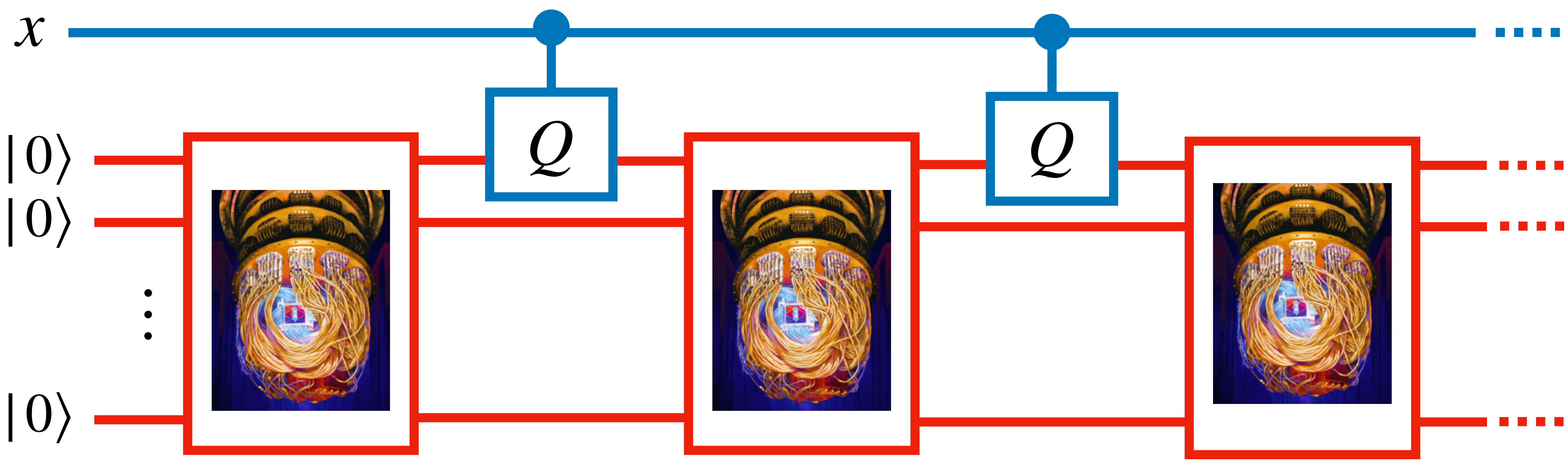
- The computation alternates between T queries and T memory updates.

- The “Query Operator” $Q$ is:



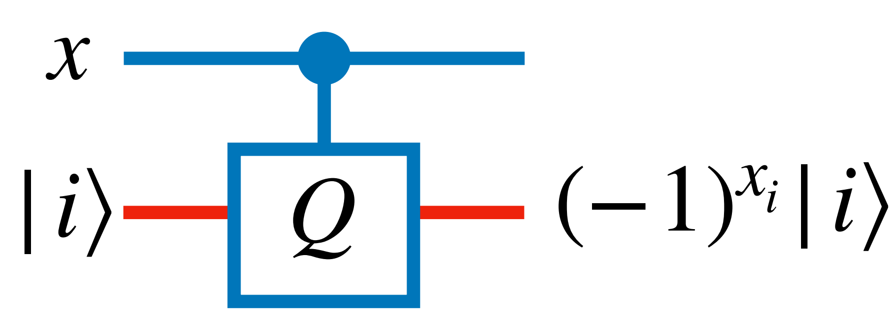
# Quantum Query Model





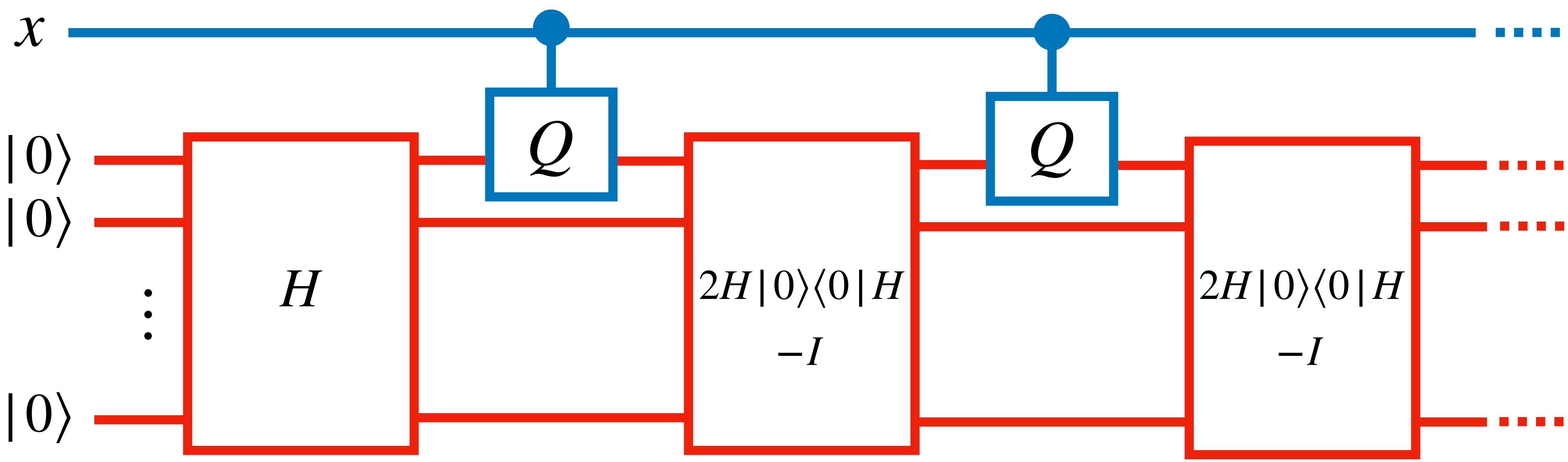

- The memory is made of **S qubits**, initially set to $|0\rangle$.

- The quantum "Query Operator" $Q$ is:
  (when $x_i \in \{0,1\}$)




- The computation alternates between T **quantum queries** and T **unitary updates/ measurements** of the memory.

# Quantum Query Model




**Example:** Grover's Search

$$\begin{cases} T = O\left(\sqrt{N}\right) \\ S = \log(N) \end{cases}$$

# Quantum Time-Space Tradeoffs



**Our focus in this talk:** quantum time-space tradeoff **lower bounds**.

**Very few existing results:**

*[Klauck et al.'07]* Sorting N numbers requires $\mathbf{T^2S \geq \Omega(N^3)}$.

# Quantum Time-Space Tradeoffs



**Our focus in this talk:** quantum time-space tradeoff **lower bounds**.

**Very few existing results:**

*[Klauck et al.'07]* Sorting N numbers requires $\mathbf{T^2S \geq \Omega(N^3)}$.

*[Klauck et al.'07]* Boolean Matrix-Matrix Multiplication requires $\mathbf{T^2S \geq \Omega(N^5)}$.

*[Klauck et al.'07]* Boolean Matrix-Vector Multiplication requires $\mathbf{T^2S \geq \Omega(N^3)}$.

*[Ambainis et al.'09]* Evaluating $Ax \geq (t,\ldots,t)$ requires $\mathbf{T^2S \geq \Omega(tN^3)}$ when $S < N/t$.

$\mathbf{TS \geq \Omega(N^2)}$ when $S > N/t$.

**Our focus in this talk:** quantum time-space tradeoff **lower bounds**.

**Very few existing results:**

*[Klauck et al.'07]* Sorting N numbers requires $\mathbf{T^2S \geq \Omega(N^3)}$.

*[Klauck et al.'07]* Boolean Matrix-Matrix Multiplication requires $\mathbf{T^2S \geq \Omega(N^5)}$.

*[Klauck et al.'07]* Boolean Matrix-Vector Multiplication requires $\mathbf{T^2S \geq \Omega(N^3)}$.

*[Ambainis et al.'09]* Evaluating $Ax \geq (t,\dots,t)$ requires $\mathbf{T^2S \geq \Omega(tN^3)}$ when $S < N/t$.

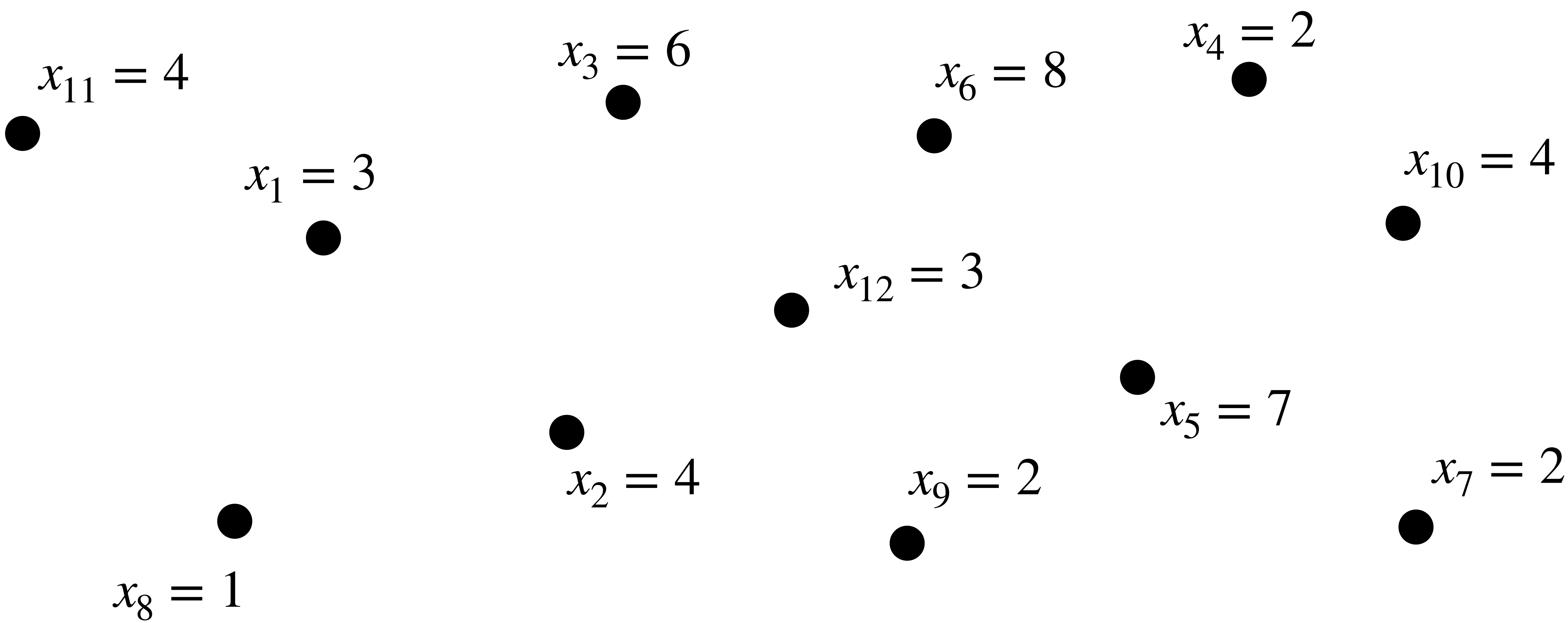
$\mathbf{TS \geq \Omega(N^2)}$ when $S > N/t$.

**Our contribution:** a new tradeoff for the **Collision Pairs Finding** problem.

2

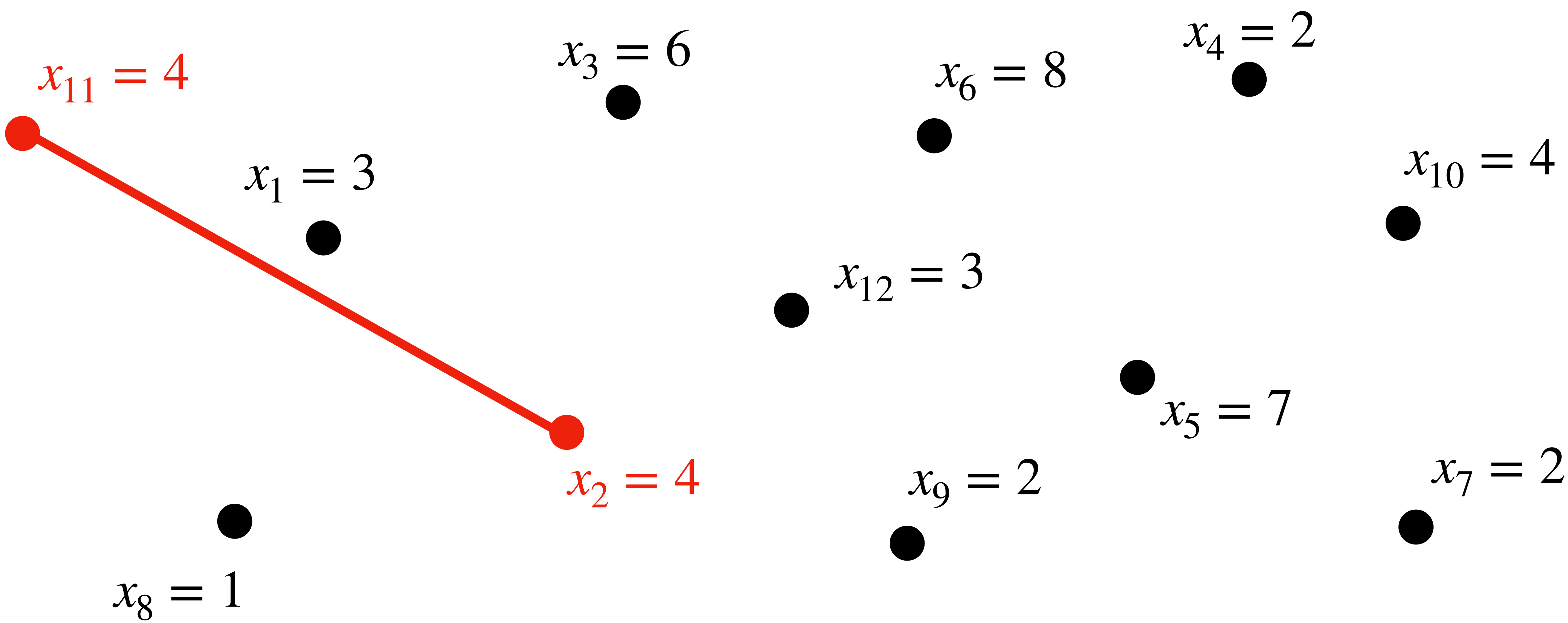
# The Collision Pairs Finding Problem

# Collision Pairs




**Collision pair: $x_i = x_j$**

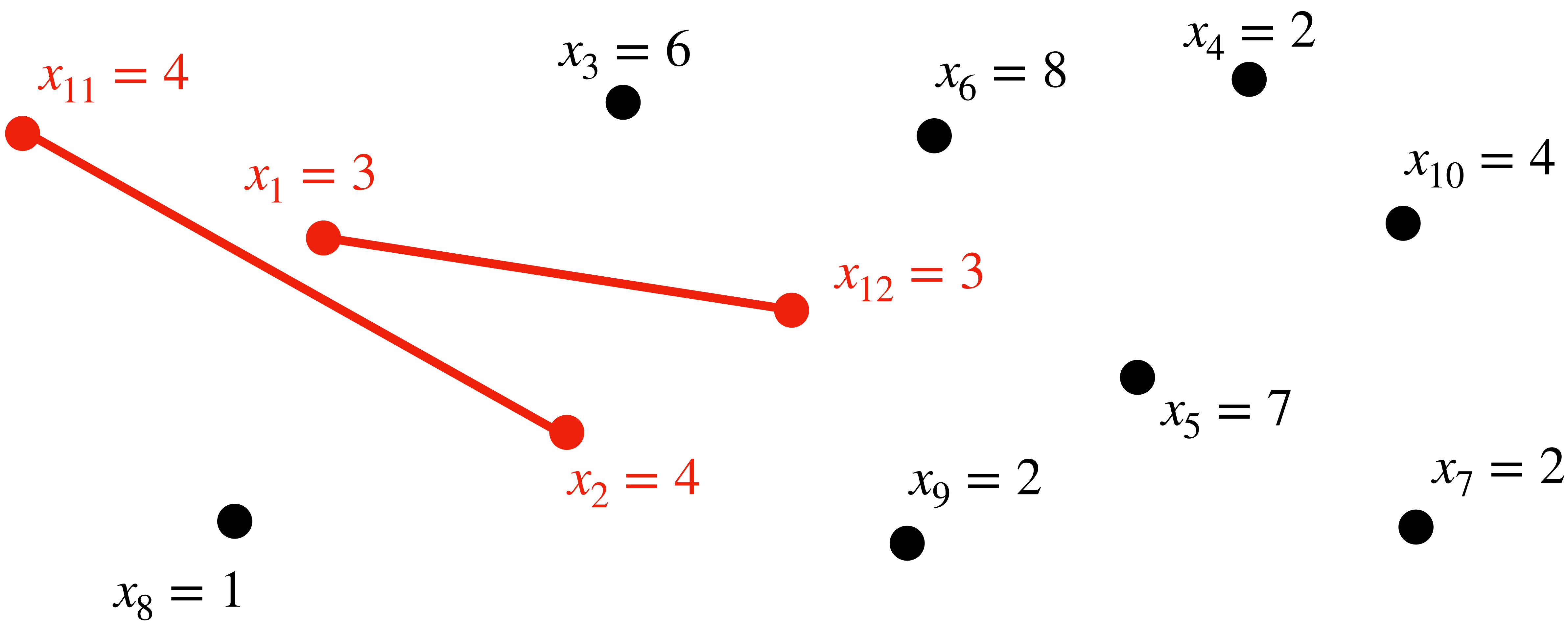
# Collision Pairs




**Collision pair: $x_i = x_j$**

# Collision Pairs






**Collision pair: $x_i = x_j$**

## K-Collision Pairs

Find K collision pairs in a **random** input $x_1, \ldots, x_N$ ~ [N].

## K-Collision Pairs

Find K collision pairs in a **random** input $x_1, \ldots, x_N$ ~ [N].

→ **A random input contains ~ Θ(N) collision pairs with high probability.**

## K-Collision Pairs

Find K collision pairs in a **random** input $x_1, \ldots, x_N$ ~ [N].

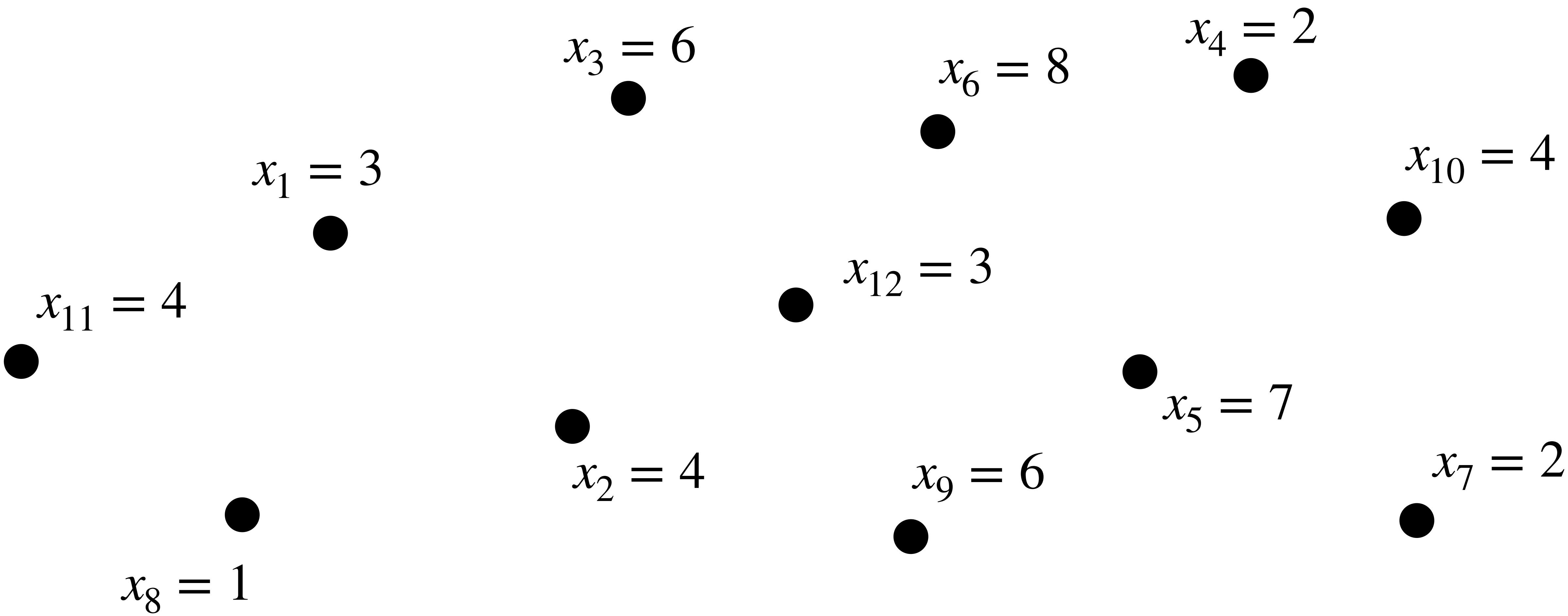
→ **A random input contains ~ Θ(N) collision pairs with high probability.**

→ **Finding collisions is an important problem in cryptanalysis:**

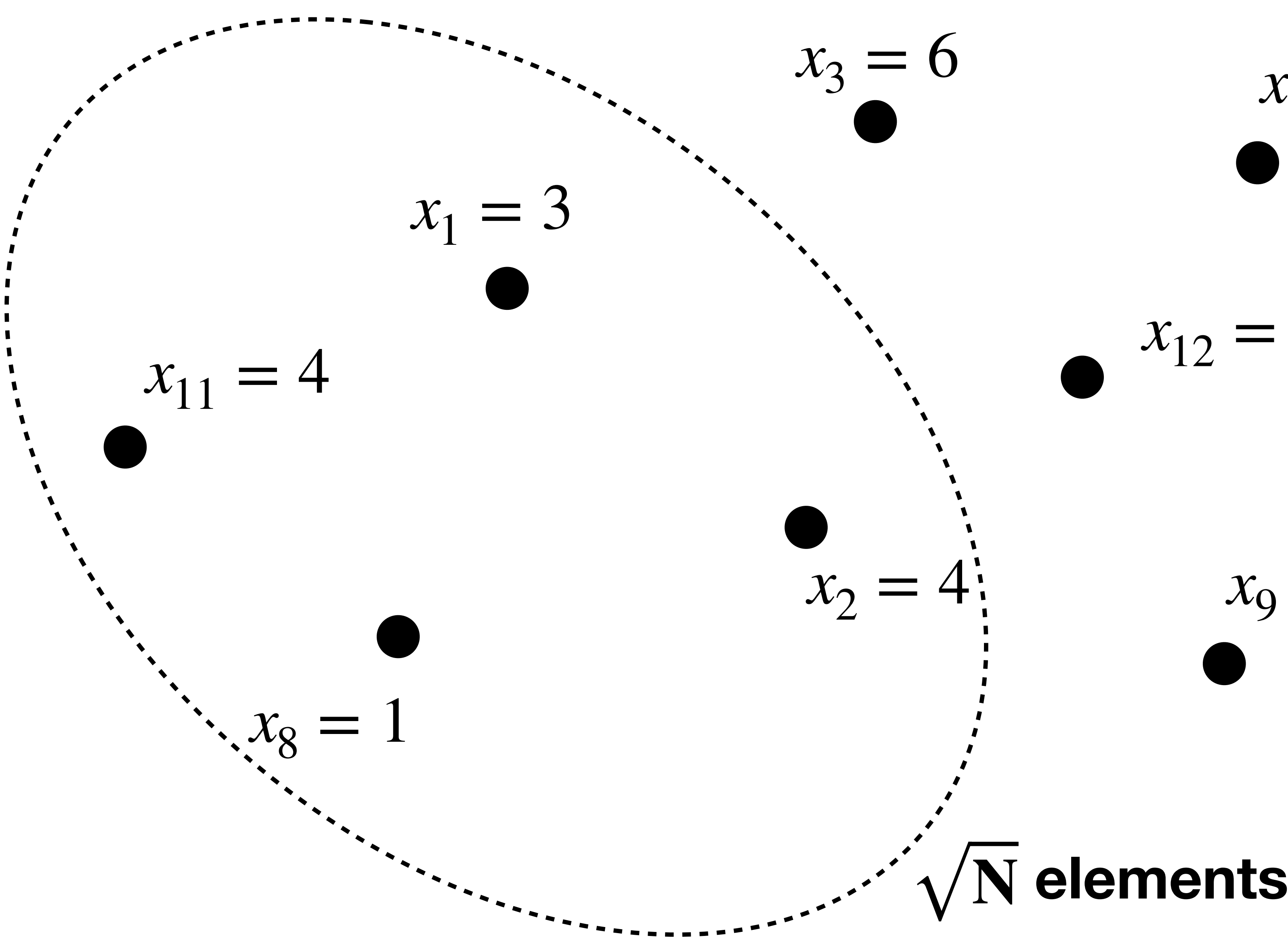
- preimage attacks on hash functions
- meet-in-the-middle attacks ← **requires to find many collisions**
- computing discrete logarithms
- …

# Birthday attack (K = 1)



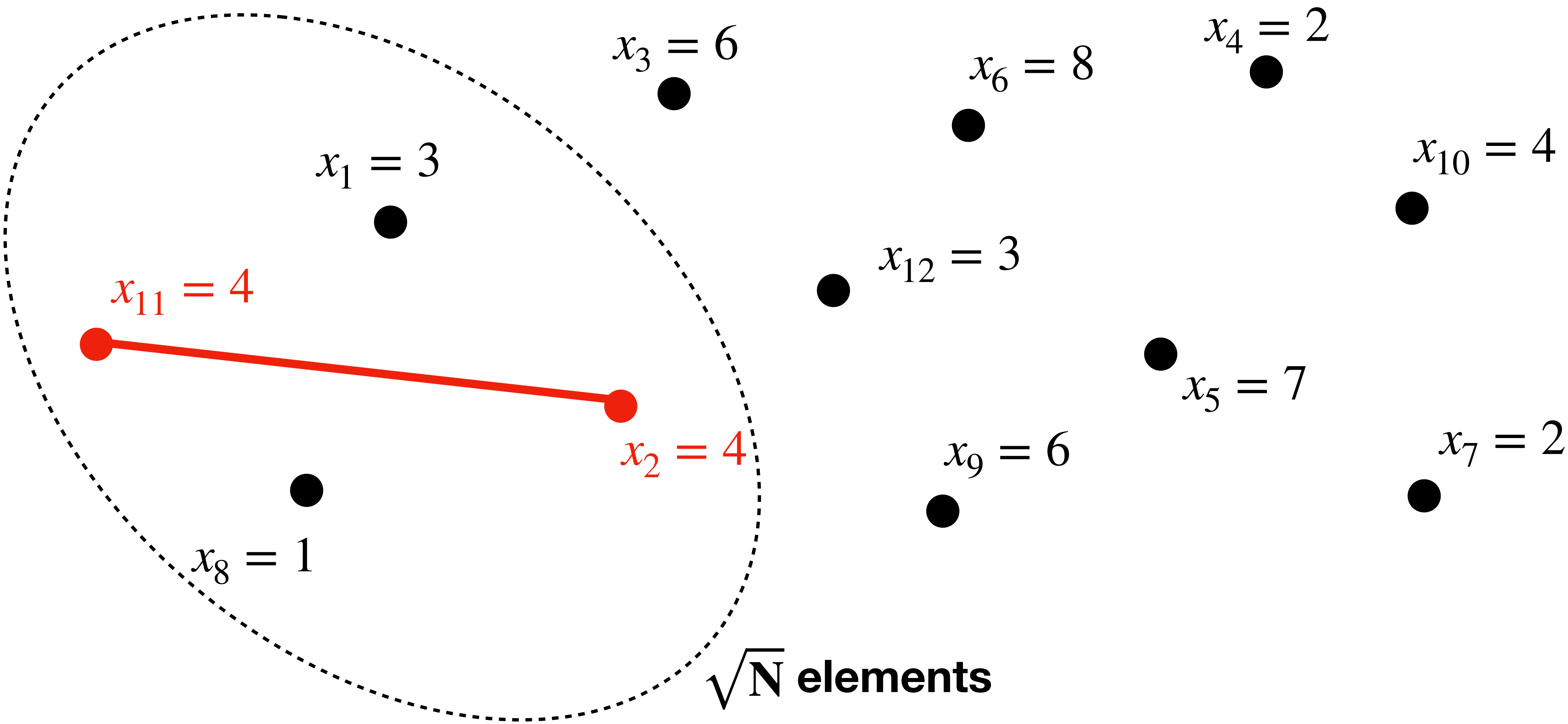


# Birthday attack (K = 1)



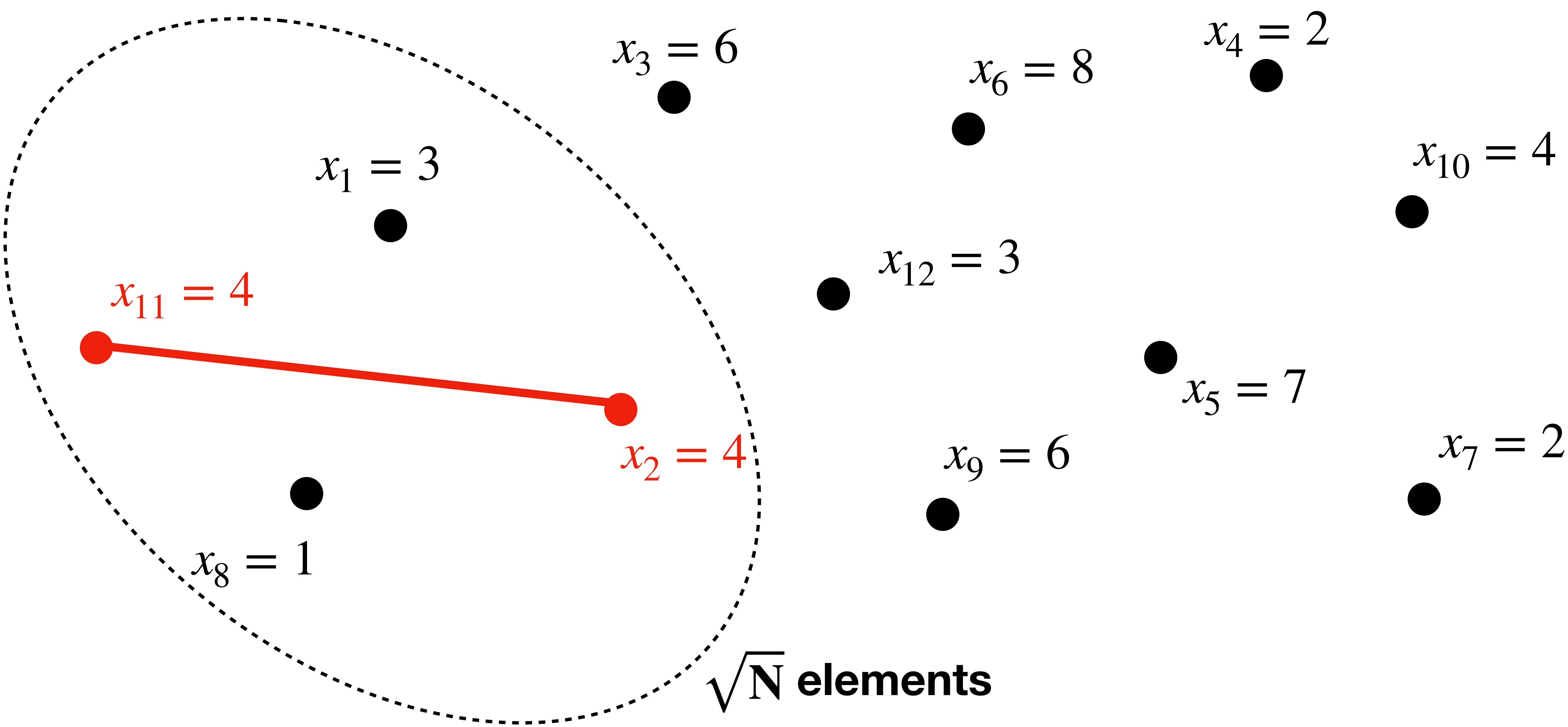

**Birthday attack**

# Birthday attack (K = 1)




**Birthday attack**

# Birthday attack (K = 1)

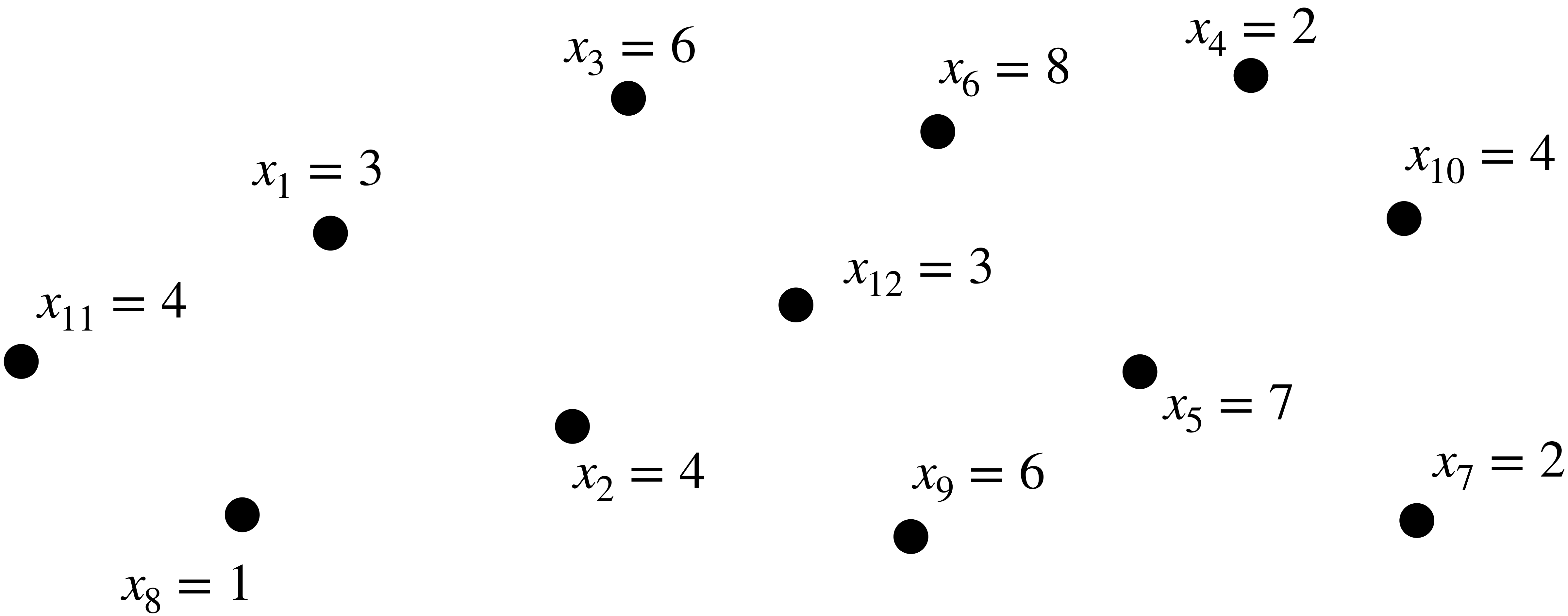



**Birthday attack**

$$T = O(\sqrt{N})$$

$$S = O(\sqrt{N})$$

# Birthday attack (K = 1)




**Birthday attack**

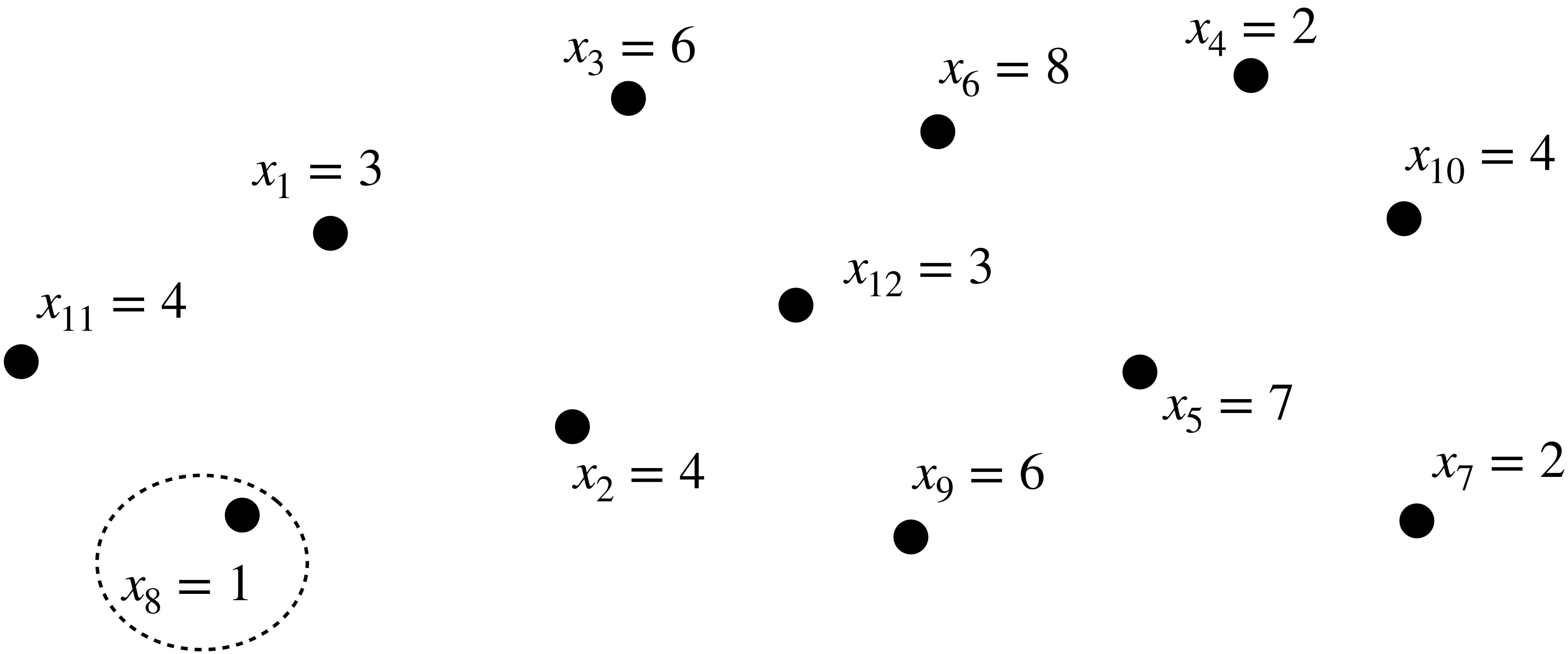
$$T = O\left(\sqrt{N}\right)$$

$$S = O\left(\sqrt{N}\right)$$

**Birthday attack +**
**Floyd's cycle finding**

# Birthday attack (K = 1)






**Birthday attack**

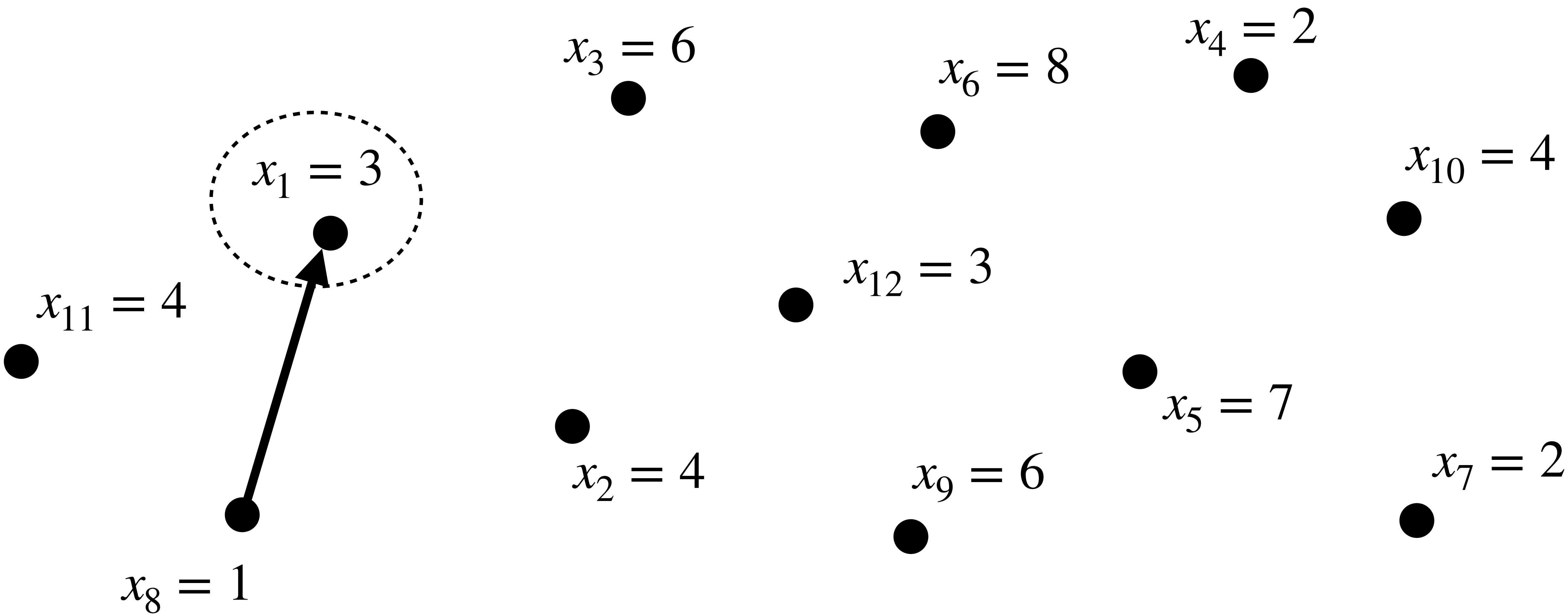
$$T = O(\sqrt{N})$$

$$S = O(\sqrt{N})$$

**Birthday attack +**
**Floyd's cycle finding**

# Birthday attack (K = 1)





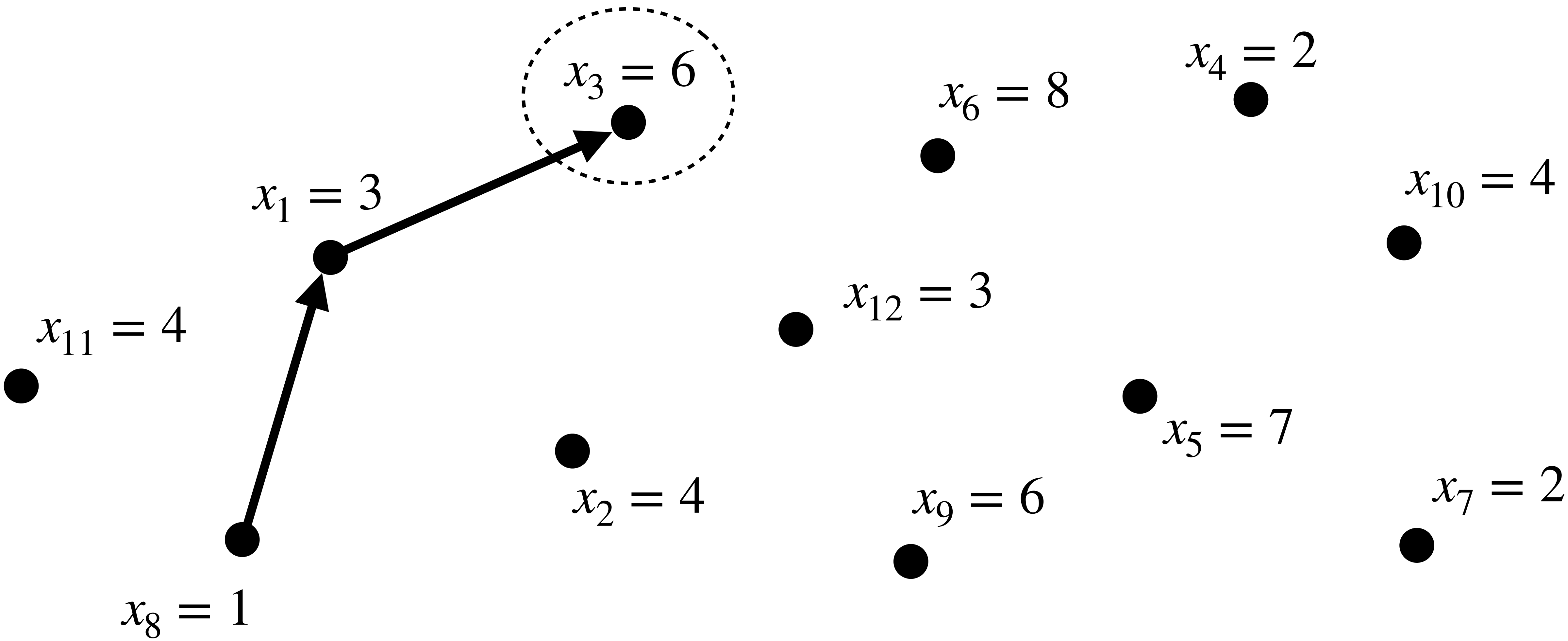

**Birthday attack**

$$T = O(\sqrt{N})$$

$$S = O(\sqrt{N})$$

**Birthday attack +**
**Floyd's cycle finding**

# Birthday attack (K = 1)




**Birthday attack**

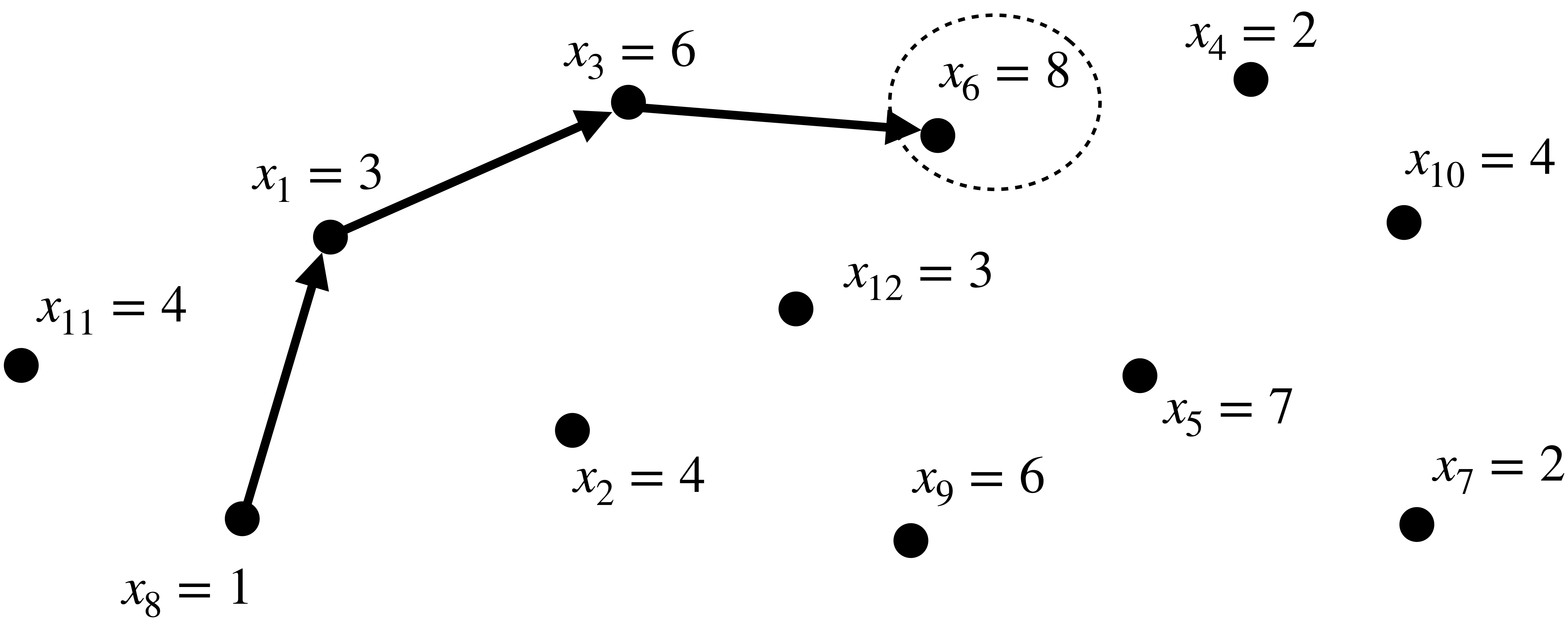
$$T = O(\sqrt{N})$$

$$S = O(\sqrt{N})$$

**Birthday attack + Floyd's cycle finding**

# Birthday attack (K = 1)






**Birthday attack**

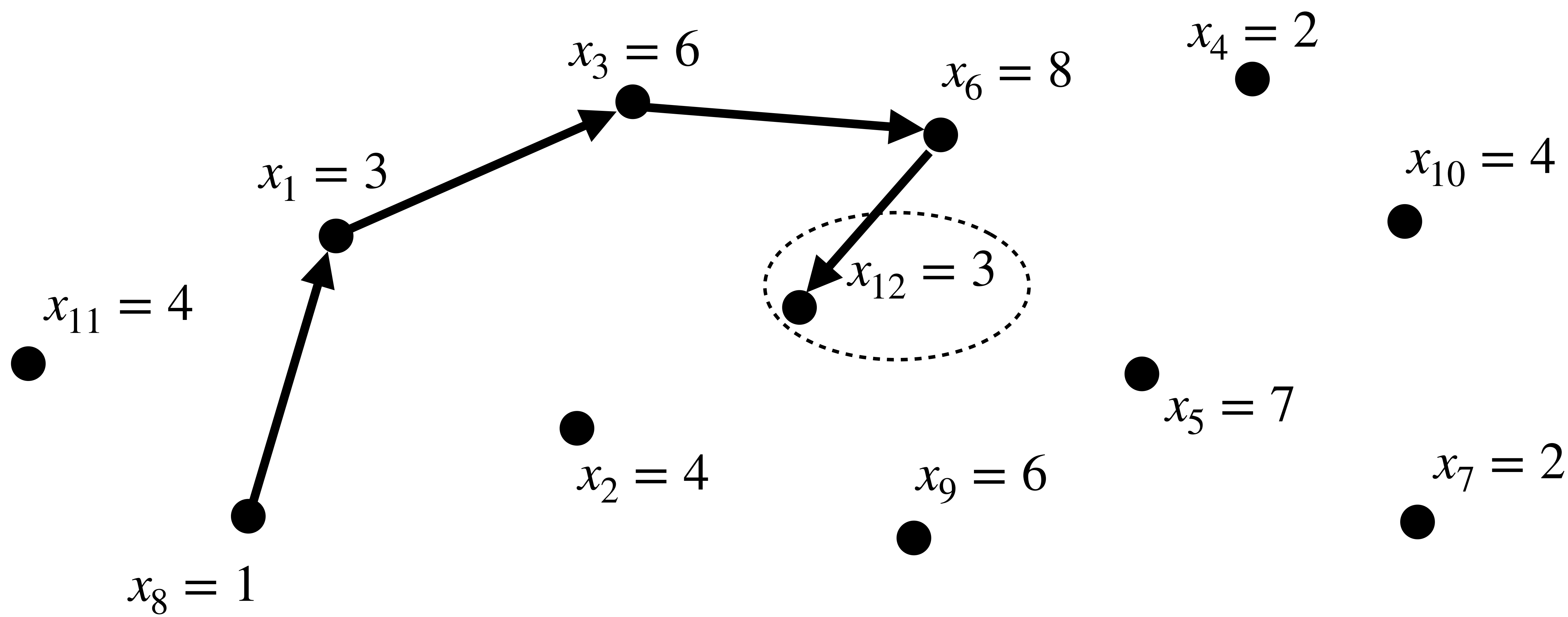
$$T = O(\sqrt{N})$$

$$S = O(\sqrt{N})$$

**Birthday attack + Floyd's cycle finding**

# Birthday attack (K = 1)






**Birthday attack**

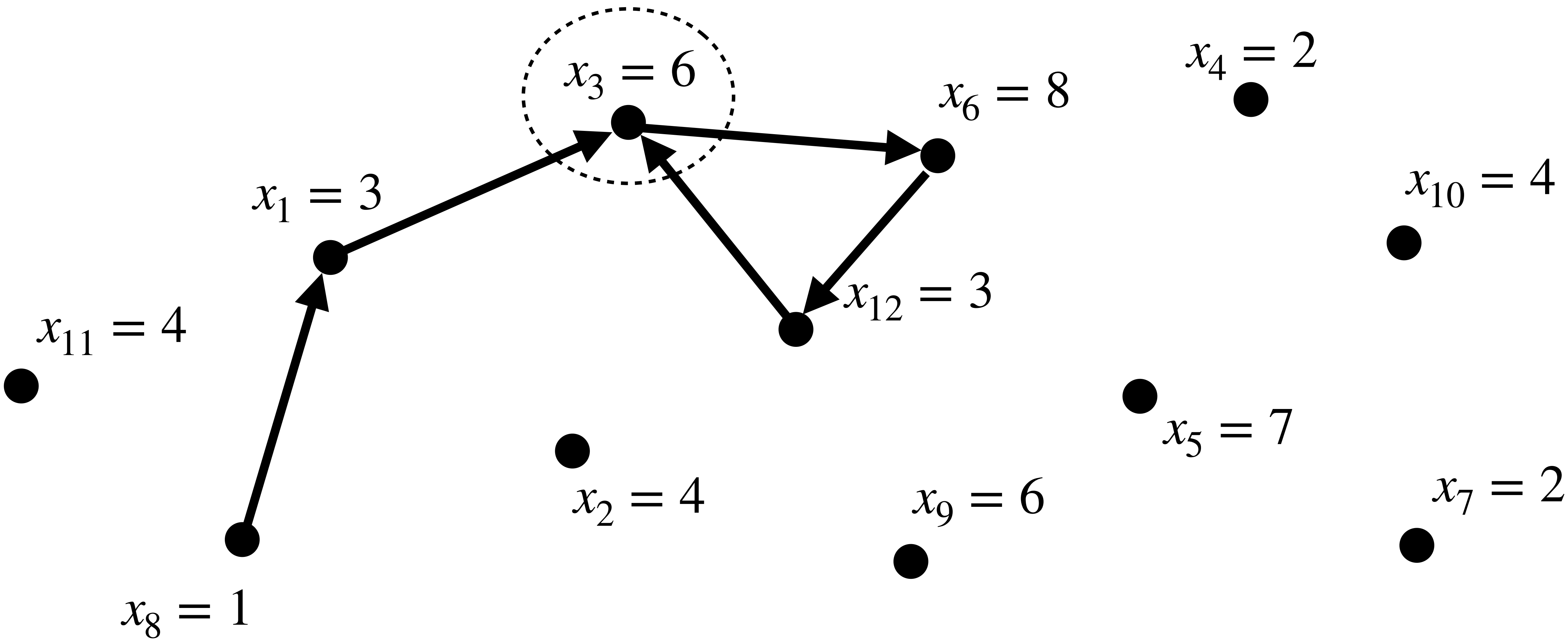
$$T = O(\sqrt{N})$$

$$S = O(\sqrt{N})$$

**Birthday attack +**
**Floyd's cycle finding**

# Birthday attack (K = 1)





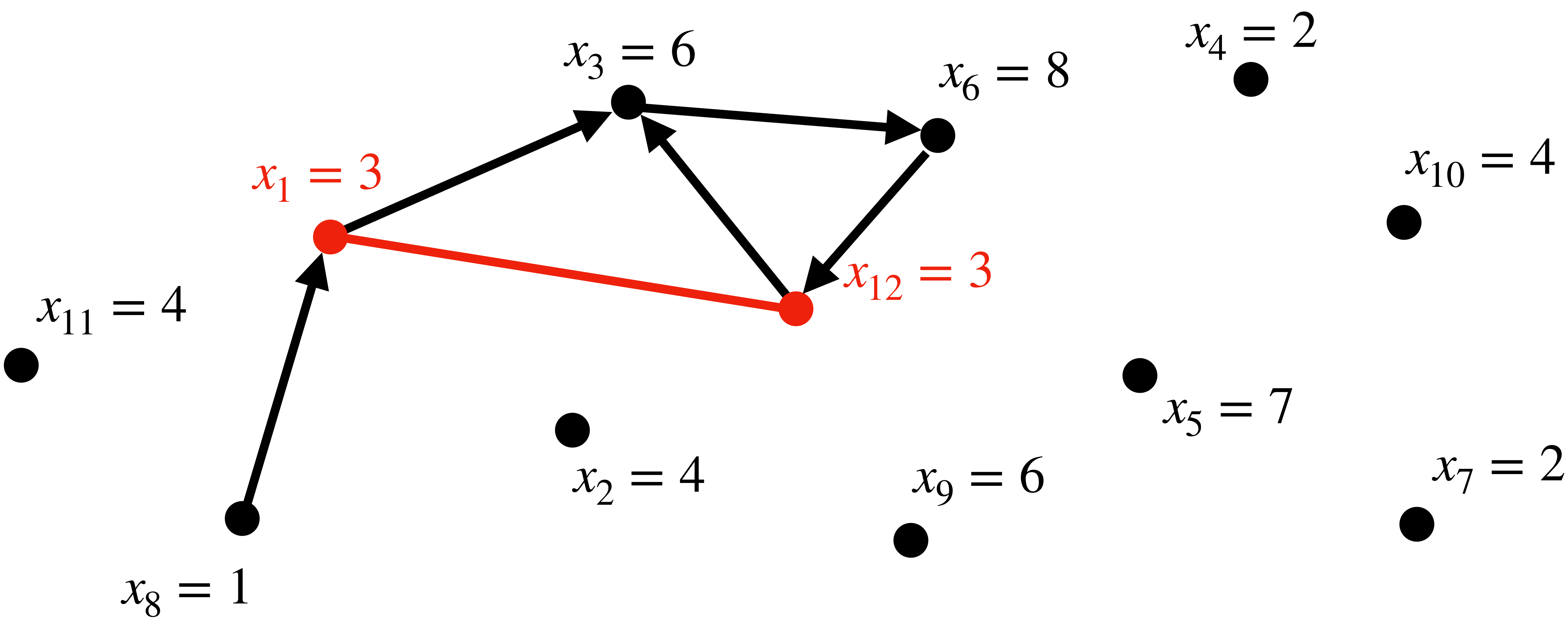

**Birthday attack**

$$T = O(\sqrt{N})$$

$$S = O(\sqrt{N})$$

**Birthday attack +**
**Floyd's cycle finding**

# Birthday attack (K = 1)




**Birthday attack**

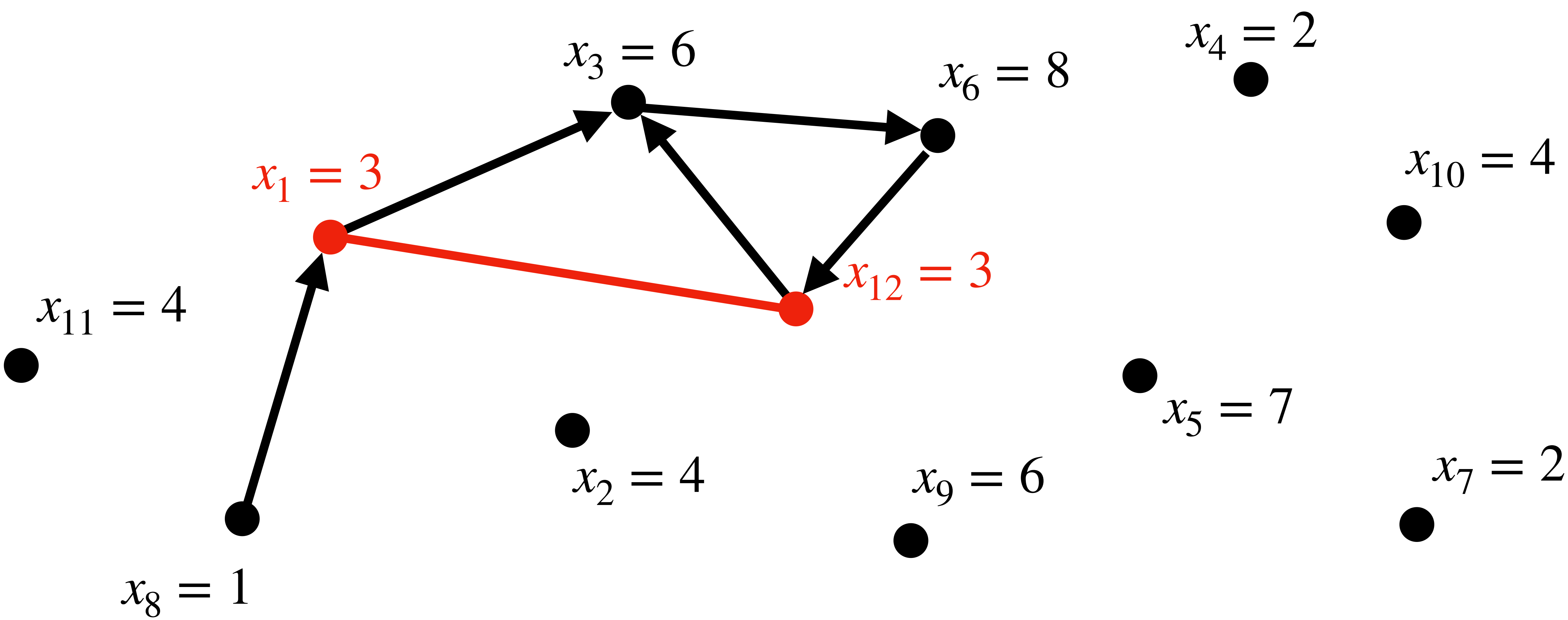
$$T = O(\sqrt{N})$$

$$S = O(\sqrt{N})$$

**Birthday attack + Floyd's cycle finding**

# Birthday attack (K = 1)






**Birthday attack**

$$T = O(\sqrt{N})$$

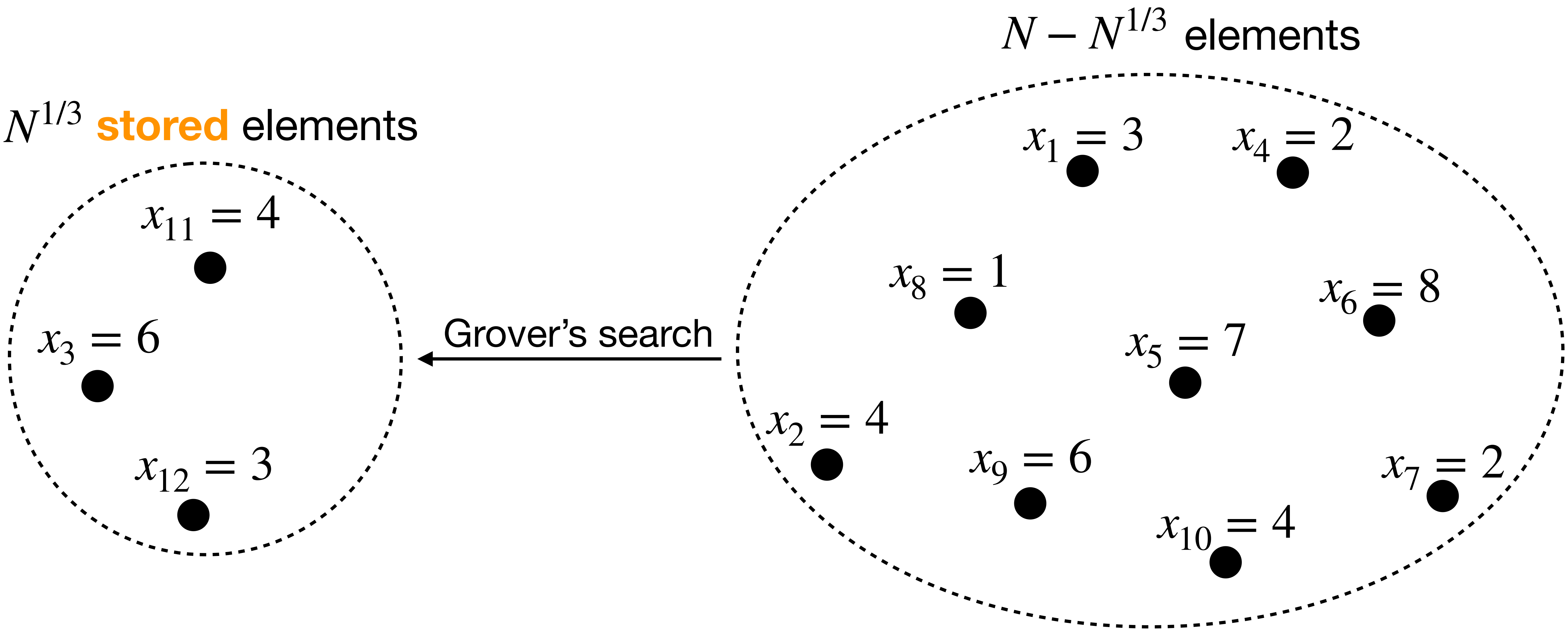
$$S = O(\sqrt{N})$$

**Birthday attack + Floyd's cycle finding**

$$T = O(\sqrt{N})$$

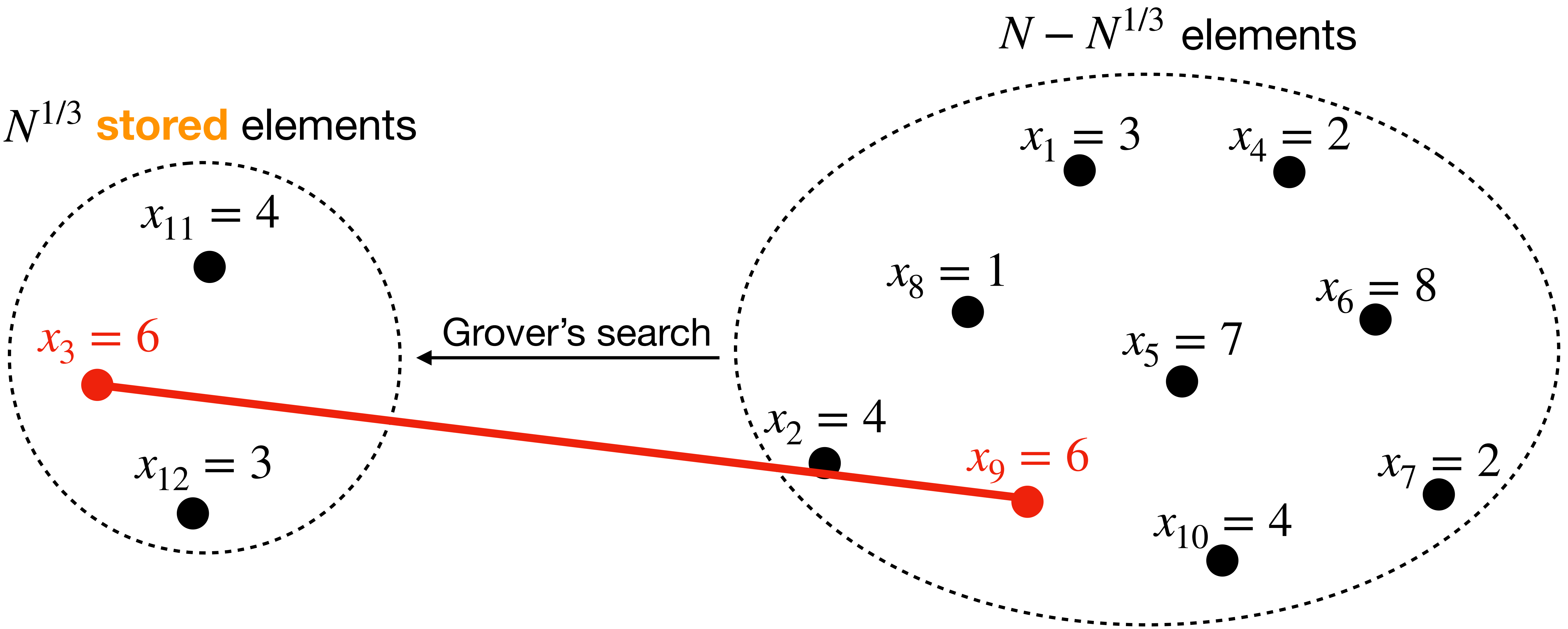
$$S = O(\log N)$$

# Quantum BHT algorithm (K = 1)




**Quantum BHT algorithm**

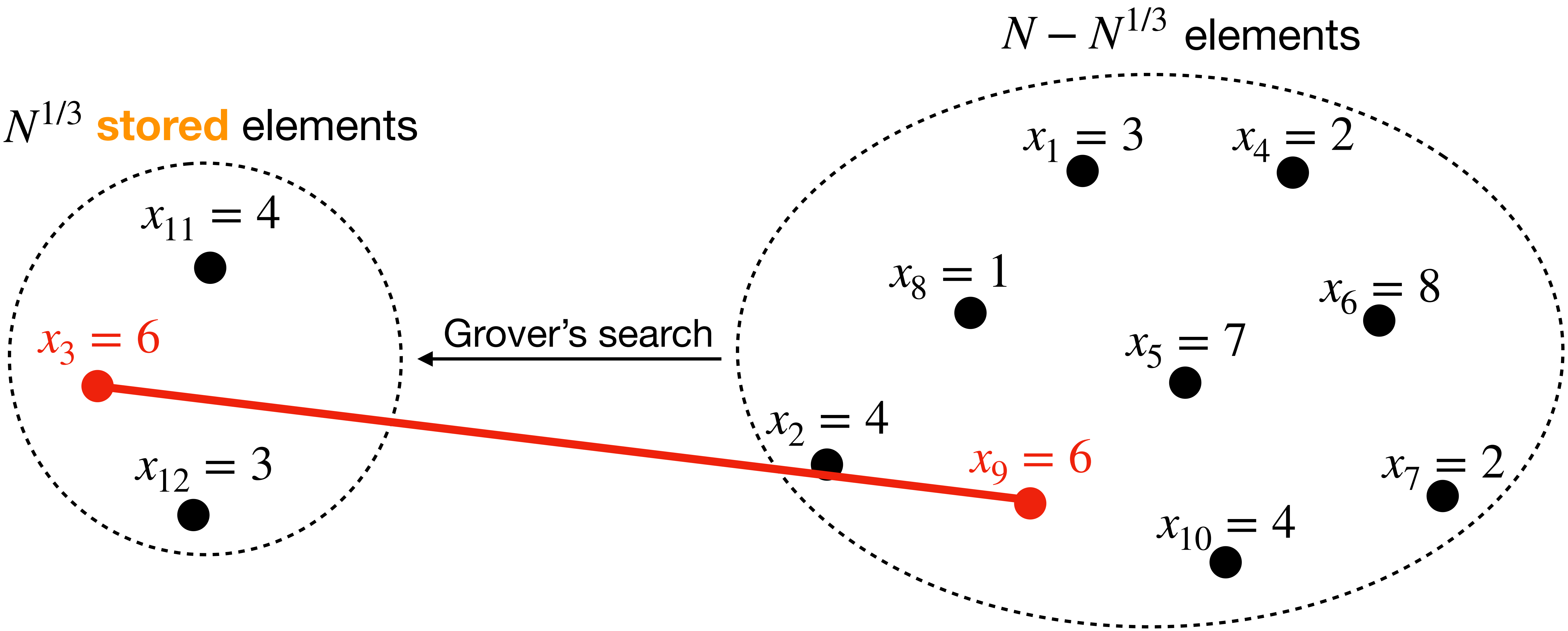
# Quantum BHT algorithm (K = 1)




**Quantum BHT algorithm**

# Quantum BHT algorithm (K = 1)






**Quantum BHT algorithm**

$$T = O\left(N^{1/3}\right)$$

$$S = O\left(N^{1/3}\right)$$

# Finding 1 collision pair



**Birthday attack +**
**Floyd's cycle finding**

$$T = O\left(\sqrt{N}\right)$$
$$S = O(\log N)$$

**vs**

**BHT algorithm**

$$T = O\left(N^{1/3}\right)$$
$$S = O\left(N^{1/3}\right)$$

The quantum BHT algorithm has a better time complexity, but a worst time-space tradeoff!

# Finding 1 collision pair



**Birthday attack + Floyd's cycle finding**

$$T = O\left(\sqrt{N}\right)$$

$$S = O(\log N)$$

**vs**

**BHT algorithm**

$$T = O\left(N^{1/3}\right)$$

$$S = O\left(N^{1/3}\right)$$

The quantum BHT algorithm has a better time complexity, but a worst time-space tradeoff!

A BHT attack on **SHA3-256** would require $S \approx 2^{256/3} \approx 2^{85}$ qubits!

# Finding 1 collision pair

| **Birthday attack + Floyd's cycle finding** | | **BHT algorithm** |
|---|---|---|
| $T = O(\sqrt{N})$ | **vs** | $T = O(N^{1/3})$ |
| $S = O(\log N)$ | | $S = O(N^{1/3})$ |

The quantum BHT algorithm has a better time complexity, but a worst time-space tradeoff!

A BHT attack on **SHA3-256** would require S ≈ $2^{256/3}$ ≈ $2^{85}$ qubits!

**Big open problem:** Is there a quantum algorithm with $T \leq o(\sqrt{N})$ and $S = O(\log N)$?

# Finding K collision pairs



| | Classical Tradeoff | Quantum Tradeoff |
|---|---|---|
| Upper bound | $T^2S \leq \tilde{O}(K^2N)$ when $\tilde{\Omega}(\log N) \leq S \leq \tilde{O}(K)$ <br> Parallel Collision Search [van Oorschot and Wiener'99] | $T^2S \leq \tilde{O}(K^2N)$ when $\tilde{\Omega}(\log N) \leq S \leq \tilde{O}(K^{2/3}N^{1/3})$ <br> Adaptation of the BHT algorithm |

# Finding K collision pairs



| | Classical Tradeoff | Quantum Tradeoff |
|---|---|---|
| Upper bound | $T^2S \leq \tilde{O}(K^2N)$ when $\tilde{\Omega}(\log N) \leq S \leq \tilde{O}(K)$<br>Parallel Collision Search [van Oorschot and Wiener'99] | $T^2S \leq \tilde{O}(K^2N)$ when $\tilde{\Omega}(\log N) \leq S \leq \tilde{O}(K^{2/3}N^{1/3})$<br>Adaptation of the BHT algorithm |
| Lower bound | $T^2S \geq \tilde{\Omega}(K^2N)$<br>[Dinur'20] | |

# Finding K collision pairs



| | Classical Tradeoff | Quantum Tradeoff |
|---|---|---|
| Upper bound | $T^2S \le \tilde{O}(K^2N)$ when $\tilde{\Omega}(\log N) \le S \le \tilde{O}(K)$ <br> Parallel Collision Search [van Oorschot and Wiener'99] | $T^2S \le \tilde{O}(K^2N)$ when $\tilde{\Omega}(\log N) \le S \le \tilde{O}(K^{2/3}N^{1/3})$ <br> Adaptation of the BHT algorithm |
| Lower bound | $T^2S \ge \tilde{\Omega}(K^2N)$ <br> [Dinur'20] | $T^3S \ge \tilde{\Omega}(K^3N)$ <br> Our result |

# Finding K collision pairs



| | Classical Tradeoff | Quantum Tradeoff |
|---|---|---|
| Upper bound | $T^2S \leq \tilde{O}(K^2N)$<br>when<br>$\tilde{\Omega}(\log N) \leq S \leq \tilde{O}(K)$<br><br>Parallel Collision Search<br>[van Oorschot and Wiener'99] | $T^2S \leq \tilde{O}(K^2N)$<br>when<br>$\tilde{\Omega}(\log N) \leq S \leq \tilde{O}(K^{2/3}N^{1/3})$<br><br>Adaptation of the BHT algorithm |
| Lower bound | $T^2S \geq \tilde{\Omega}(K^2N)$<br><br>[Dinur'20] | $T^3S \geq \tilde{\Omega}(K^3N)$<br><br>Our result |

We conjecture: $T^2S \geq \tilde{\Omega}(K^2N)$

# Finding K collision pairs



**Our result:** $T^3S \geq \tilde{\Omega}(K^3N)$

**Conjecture:** $T^2S \geq \tilde{\Omega}(K^2N)$

# Finding K collision pairs



**Our result:** $T^3S \geq \tilde{\Omega}(K^3N)$ **Conjecture:** $T^2S \geq \tilde{\Omega}(K^2N)$

- Our result is non-trivial when $K \geq \omega(1)$:

# Finding K collision pairs



**Our result:** $T^3S \geq \tilde{\Omega}(K^3N)$ **Conjecture:** $T^2S \geq \tilde{\Omega}(K^2N)$

- Our result is non-trivial when $K \geq \omega(1)$:
  - → For $K = 1$ and $S = \log(N)$ it gives $T \geq \tilde{\Omega}(N^{1/3})$, which is the same as the time-only lower bound [Aaronson and Shi'04].

# Finding K collision pairs



**Our result:** $T^3S \geq \tilde{\Omega}(K^3N)$ **Conjecture:** $T^2S \geq \tilde{\Omega}(K^2N)$

- Our result is non-trivial when $K \geq \omega(1)$:
  - → For $K = 1$ and $S = \log(N)$ it gives $T \geq \tilde{\Omega}(N^{1/3})$, which is the same as the time-only lower bound [Aaronson and Shi'04].
  - → For $K \geq \omega(1)$ and $S = \log(N)$ it gives $T \geq \tilde{\Omega}(KN^{1/3})$, whereas we prove that the best time-only lower bound is $T = \tilde{\Theta}(K^{2/3}N^{1/3})$.

# Finding K collision pairs



**Our result:** $T^3S \geq \tilde{\Omega}(K^3N)$ **Conjecture:** $T^2S \geq \tilde{\Omega}(K^2N)$

- Our result is non-trivial when $K \geq \omega(1)$:
  - → For $K = 1$ and $S = \log(N)$ it gives $T \geq \tilde{\Omega}(N^{1/3})$, which is the same as the time-only lower bound [Aaronson and Shi'04].
  - → For $K \geq \omega(1)$ and $S = \log(N)$ it gives $T \geq \tilde{\Omega}(KN^{1/3})$, whereas we prove that the best time-only lower bound is $T = \tilde{\Theta}(K^{2/3}N^{1/3})$.
- The conjecture $T^2S \geq \tilde{\Omega}(N^3)$ for $K = \tilde{\Theta}(N)$ is particularly interesting:

# Finding K collision pairs 

**Our result:** $T^3S \geq \tilde{\Omega}(K^3N)$ **Conjecture:** $T^2S \geq \tilde{\Omega}(K^2N)$

- Our result is non-trivial when $K \geq \omega(1)$:

  → For $K = 1$ and $S = \log(N)$ it gives $T \geq \tilde{\Omega}(N^{1/3})$, which is the same as the time-only lower bound [Aaronson and Shi'04].

  → For $K \geq \omega(1)$ and $S = \log(N)$ it gives $T \geq \tilde{\Omega}(KN^{1/3})$, whereas we prove that the best time-only lower bound is $T = \tilde{\Theta}(K^{2/3}N^{1/3})$.

- The conjecture **$T^2S \geq \tilde{\Omega}(N^3)$** for **$K = \tilde{\Theta}(N)$** is particularly interesting:

  → Time-space tradeoffs are generally easier to prove when the output is large.

# Finding K collision pairs



**Our result:** $T^3S \geq \tilde{\Omega}(K^3N)$ **Conjecture:** $T^2S \geq \tilde{\Omega}(K^2N)$

- Our result is non-trivial when $K \geq \omega(1)$:

  → For $K = 1$ and $S = \log(N)$ it gives $T \geq \tilde{\Omega}(N^{1/3})$, which is the same as the time-only lower bound [Aaronson and Shi'04].

  → For $K \geq \omega(1)$ and $S = \log(N)$ it gives $T \geq \tilde{\Omega}(KN^{1/3})$, whereas we prove that the best time-only lower bound is $T = \tilde{\Theta}(K^{2/3}N^{1/3})$.

- The conjecture $T^2S \geq \tilde{\Omega}(N^3)$ for $K = \tilde{\Theta}(N)$ is particularly interesting:

  → Time-space tradeoffs are generally easier to prove when the output is large.

  → If true, we show that it would implies $T^2S \geq \tilde{\Omega}(N^2)$ for **Element Distinctness.**

3

# Lower Bounds by Recording Queries

# Time-Space Lower Bounds from Time Lower Bounds



[Borodin et al.'81] : a general method to convert **Time-only lower bounds** directly into **Time-Space lower bounds.**

# Time-Space Lower Bounds from Time Lower Bounds



[Borodin et al.'81] : a general method to convert **Time-only lower bounds**

directly into **Time-Space lower bounds.**

→ The problem must have a **large output** (≠ decision problem).

→ The time lower bound is in the **exponentially small** success probability regime.

# Time-Space Lower Bounds from Time Lower Bounds

[Borodin et al.'81] : a general method to convert **Time-only lower bounds** directly into **Time-Space lower bounds.**

→ The problem must have a **large output** (≠ decision problem).

→ The time lower bound is in the **exponentially small** success probability regime.

**K-Search** *[Klauck et al.'07, Ambainis'10, …]*

Finding **K ones** in $x \in \{0,1\}^N, |x| = K$ with success probability at least $2^{-O(K)}$ requires time $T \geq \Omega\left(\sqrt{NK}\right)$.

$\Rightarrow$

All existing quantum time-space tradeoffs

# Time-Space Lower Bounds from Time Lower Bounds

[Borodin et al.'81] : a general method to convert **Time-only lower bounds** directly into **Time-Space lower bounds.**

→ The problem must have a **large output** (≠ decision problem).

→ The time lower bound is in the **exponentially small** success probability regime.

**K-Search** *[Klauck et al.'07, Ambainis'10, …]*

Finding **K ones** in $x \in \{0,1\}^N, |x| = K$ with success probability at least $2^{-O(K)}$ requires time $T \geq \Omega\left(\sqrt{NK}\right)$.

$\Rightarrow$ All existing quantum time-space tradeoffs

**K Collisions**

Finding **K collisions** in $x \sim [N]^N$ with success probability at least $2^{-O(K)}$ requires time $T \geq \Omega\left(K^{2/3}N^{1/3}\right)$.

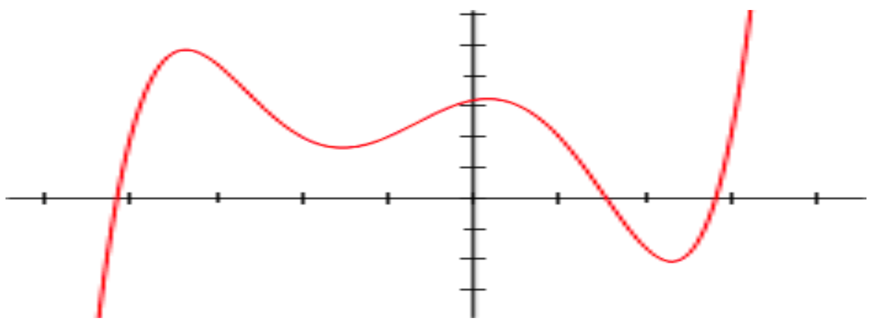
$\Rightarrow$ $T^3 S \geq \tilde{\Omega}(K^3 N)$ for K-Collision Pairs Finding

**Two main methods for proving quantum query lower bounds:**

**Two main methods for proving quantum query lower bounds:**

**Polynomial Method**

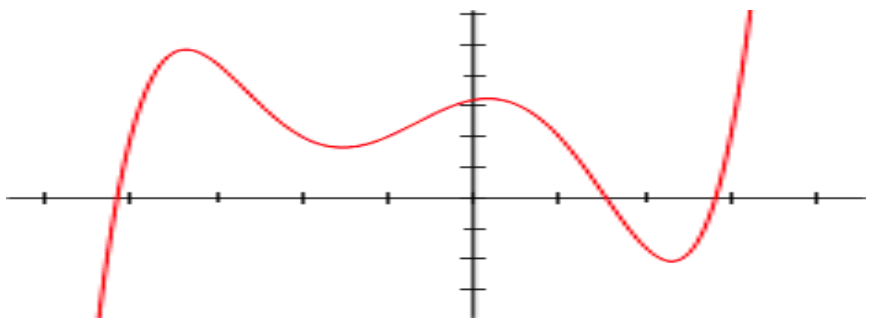
The acceptance probability of a T-query algorithm
is a polynomial in x of degree at most 2T.



**Two main methods for proving quantum query lower bounds:**

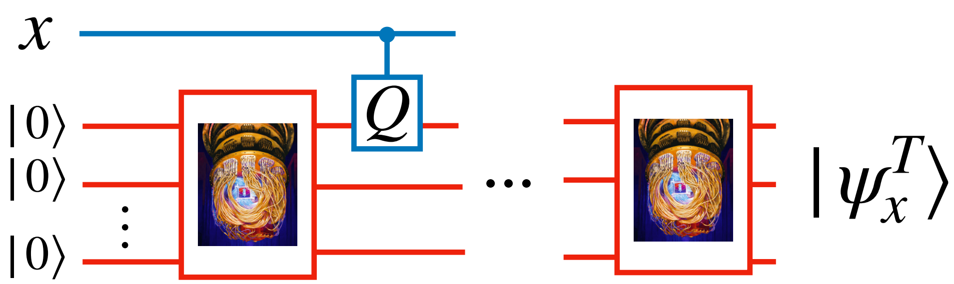
### Polynomial Method

The acceptance probability of a T-query algorithm is a polynomial in x of degree at most 2T.



### Adversary Method

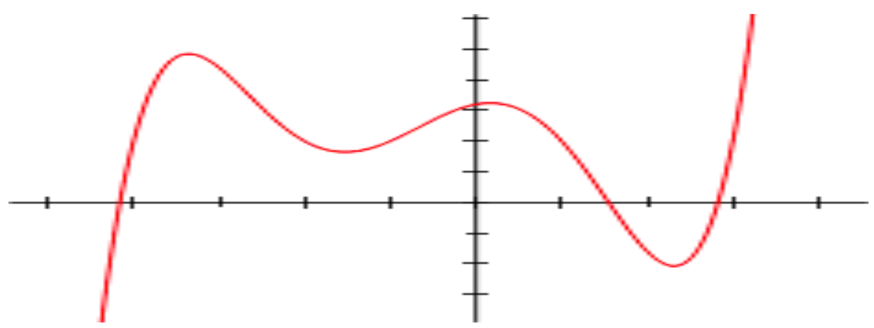
Bound the progress $W^t = \sum_{x,y} w_{x,y} \langle \psi_x^t | \psi_y^t \rangle$ .



**Two main methods for proving quantum query lower bounds:**

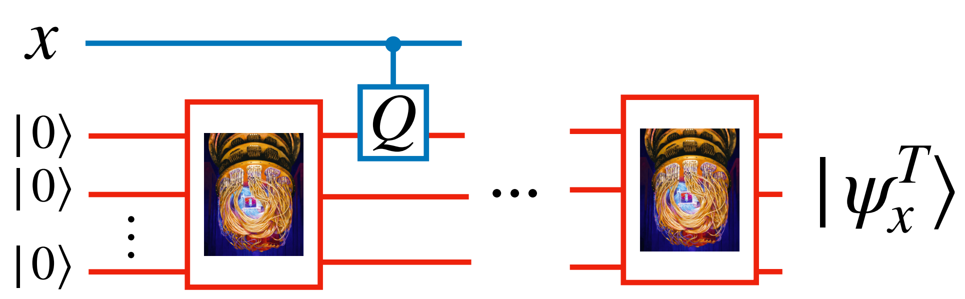
## Polynomial Method

The acceptance probability of a T-query algorithm is a polynomial in x of degree at most 2T.



## Adversary Method

Bound the progress $W^t = \sum_{x,y} w_{x,y} \langle \psi_x^t | \psi_y^t \rangle$ .




**Both methods are often difficult to use in practice:**

## K-Search in [Klauck et al.'07]

Coppersmith-Rivlin's bound + Extremal properties of Chebyshev polynomials.

## K-Search in [Ambainis'10]

Analysis of the eigenspaces of the Johnson Association Scheme.

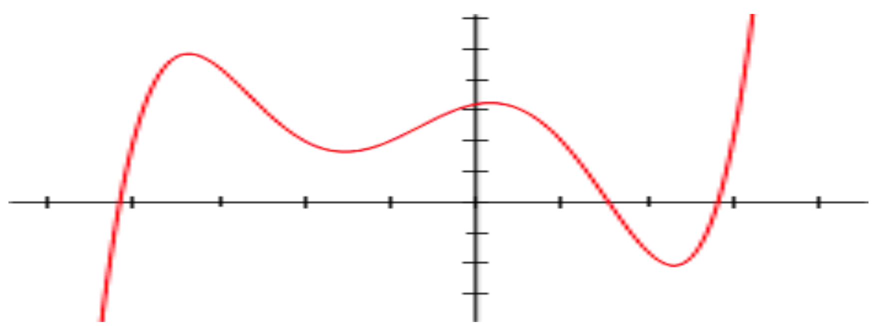
# Time Lower Bounds



**Two main methods for proving quantum query lower bounds:**

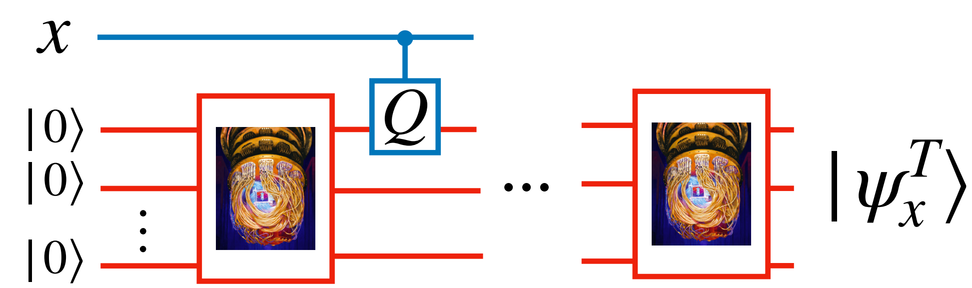
## Polynomial Method

The acceptance probability of a T-query algorithm is a polynomial in x of degree at most 2T.



## Adversary Method

Bound the progress $W^t = \sum_{x,y} w_{x,y} \langle \psi_x^t | \psi_y^t \rangle$ .




**Both methods are often difficult to use in practice:**

## K-Search in [Klauck et al.'07]

Coppersmith-Rivlin's bound + Extremal properties of Chebyshev polynomials.

## K-Search in [Ambainis'10]

Analysis of the eigenspaces of the Johnson Association Scheme.

**A simpler and more intuitive method?**

# Classical Lower Bound for K-Search



**Input:** $x = (x_1, \ldots, x_N)$ where $x_i = 1$ with probability K/N.

# Classical Lower Bound for K-Search



**Input:** $x = (x_1, \ldots, x_N)$ where $x_i$ = 1 with probability K/N.

**Strategy:** sample each entry only when it is queried, and record its value.

# Classical Lower Bound for K-Search



**Input:** $x = (x_1, \ldots, x_N)$ where $x_i = 1$ with probability K/N.

**Strategy:** sample each entry only when it is queried, and record its value.

**Algorithm**

**Input**

$$x = (\perp, \perp, \perp, \perp)$$

# Classical Lower Bound for K-Search



**Input:** $x = (x_1, \ldots, x_N)$ where $x_i = 1$ with probability K/N.

**Strategy:** sample each entry only when it is queried, and record its value.

**Algorithm** **Input**

$x = ( \perp , \perp , \perp , \perp )$

$\xrightarrow{x_2?}$

# Classical Lower Bound for K-Search

**Input:** $x = (x_1, \ldots, x_N)$ where $x_i$ = 1 with probability K/N.

**Strategy:** sample each entry only when it is queried, and record its value.

**Algorithm**

$\xrightarrow{x_2?}$

**Input**

$x = ( \perp , \perp , \perp , \perp )$

$x = ( \perp , 0 , \perp , \perp )$

# Classical Lower Bound for K-Search



**Input:** $x = (x_1, \ldots, x_N)$ where $x_i = 1$ with probability K/N.

**Strategy:** sample each entry only when it is queried, and record its value.

**Algorithm** | **Input**

$$x = (\perp, \perp, \perp, \perp)$$

$$\xrightarrow{x_2?}$$

$$\xleftarrow{x_2 = 0}$$

$$x = (\perp, 0, \perp, \perp)$$

# Classical Lower Bound for K-Search



**Input:** $x = (x_1, \ldots, x_N)$ where $x_i$ = 1 with probability K/N.

**Strategy:** sample each entry only when it is queried, and record its value.

**Algorithm** **Input**

$x = ( \perp , \perp , \perp , \perp )$

$\xrightarrow{x_2 ?}$

$\xleftarrow{x_2 = 0}$

$x = ( \perp , 0 , \perp , \perp )$

$\xrightarrow{x_1 ?}$

# Classical Lower Bound for K-Search



**Input:** $x = (x_1, \ldots, x_N)$ where $x_i = 1$ with probability K/N.

**Strategy:** sample each entry only when it is queried, and record its value.

**Algorithm**

**Input**

$x = ( \perp , \perp , \perp , \perp )$

$\xrightarrow{x_2 ?}$

$\xleftarrow{x_2 = 0}$

$x = ( \perp , 0 , \perp , \perp )$

$\xrightarrow{x_1 ?}$

$x = ( 1 , 0 , \perp , \perp )$

# Classical Lower Bound for K-Search



**Input:** $x = (x_1, \ldots, x_N)$ where $x_i$ = 1 with probability K/N.

**Strategy:** sample each entry only when it is queried, and record its value.

**Algorithm** **Input**

$x = (\perp, \perp, \perp, \perp)$

$\xrightarrow{x_2?}$

$\xleftarrow{x_2 = 0}$ $x = (\perp, 0, \perp, \perp)$

$\xrightarrow{x_1?}$

$\xleftarrow{x_1 = 1}$ $x = (1, 0, \perp, \perp)$

# Classical Lower Bound for K-Search

**Input:** $x = (x_1, \ldots, x_N)$ where $x_i$ = 1 with probability K/N.

**Strategy:** sample each entry only when it is queried, and record its value.

**Algorithm** | **Input**

$x = (\perp, \perp, \perp, \perp)$

$\xrightarrow{x_2?}$

$\xleftarrow{x_2 = 0}$ $\quad x = (\perp, 0, \perp, \perp)$

$\xrightarrow{x_1?}$

$\xleftarrow{x_1 = 1}$ $\quad x = (1, 0, \perp, \perp)$

$\xrightarrow{x_2?}$

$\xleftarrow{x_2 = 0}$ $\quad x = (1, 0, \perp, \perp)$

$\vdots$

# Classical Lower Bound for K-Search

**Input:** $x = (x_1, \ldots, x_N)$ where $x_i = 1$ with probability K/N.

**Strategy:** sample each entry only when it is queried, and record its value.

**Algorithm** **Input**

$$x = (\perp, \perp, \perp, \perp)$$

$$\xrightarrow{x_2?}$$

$$\xleftarrow{x_2 = 0} \qquad x = (\perp, 0, \perp, \perp)$$

$$\xrightarrow{x_1?}$$

$$\xleftarrow{x_1 = 1} \qquad x = (1, 0, \perp, \perp)$$

$$\xrightarrow{x_2?}$$

$$\xleftarrow{x_2 = 0} \qquad x = (1, 0, \perp, \perp)$$

$$\vdots$$

Probability to have recorded at least K/2 ones after T queries $\leq \binom{T}{K/2}\left(\frac{K}{N}\right)^{K/2}$

# Classical Lower Bound for K-Search



**Input:** $x = (x_1, \ldots, x_N)$ where $x_i = 1$ with probability K/N.

**Strategy:** sample each entry only when it is queried, and record its value.

**Algorithm** | **Input**

$x = (\perp, \perp, \perp, \perp)$

$\xrightarrow{x_2?}$

$\xleftarrow{x_2 = 0}$ $\quad x = (\perp, 0, \perp, \perp)$

$\xrightarrow{x_1?}$

$\xleftarrow{x_1 = 1}$ $\quad x = (1, 0, \perp, \perp)$

$\xrightarrow{x_2?}$

$\xleftarrow{x_2 = 0}$ $\quad x = (1, 0, \perp, \perp)$

$\vdots$

Probability to have recorded at least K/2 ones after T queries $\leq \binom{T}{K/2}\left(\frac{K}{N}\right)^{K/2}$

$\leq 2^{-\Omega(K)}$ when $T \leq O(N)$

# Classical Lower Bound for K-Search



**Input:** $x = (x_1, \ldots, x_N)$ where $x_i$ = 1 with probability K/N.

**Strategy:** sample each entry only when it is queried, and record its value.

**Algorithm** | **Input**

$x = (\perp, \perp, \perp, \perp)$

$\xrightarrow{x_2?}$

$\xleftarrow{x_2 = 0}$ $\quad x = (\perp, 0, \perp, \perp)$

$\xrightarrow{x_1?}$

$\xleftarrow{x_1 = 1}$ $\quad x = (1, 0, \perp, \perp)$

$\xrightarrow{x_2?}$

$\xleftarrow{x_2 = 0}$ $\quad x = (1, 0, \perp, \perp)$

$\vdots$

Probability to have recorded at least K/2 ones after T queries $\leq \binom{T}{K/2}\left(\frac{K}{N}\right)^{K/2}$

$\leq 2^{-\Omega(K)}$ when $T \leq O(N)$

(The un-recorded positions can only be guessed, with success $\leq (K/N)^{K/2} \leq 2^{-\Omega(K)}$).

# Recording of Quantum Queries



**Can we record quantum queries similarly?**

**Can we record quantum queries similarly?**

**[Zhandry'19]:**

- A quantum "recording technique" that works when the input $x_1, \ldots, x_N$ is sampled from the **uniform distribution** on $[M]^N$.
- Motivations: security proofs in the quantum random oracle model.

# Recording of Quantum Queries 

**Can we record quantum queries similarly?**

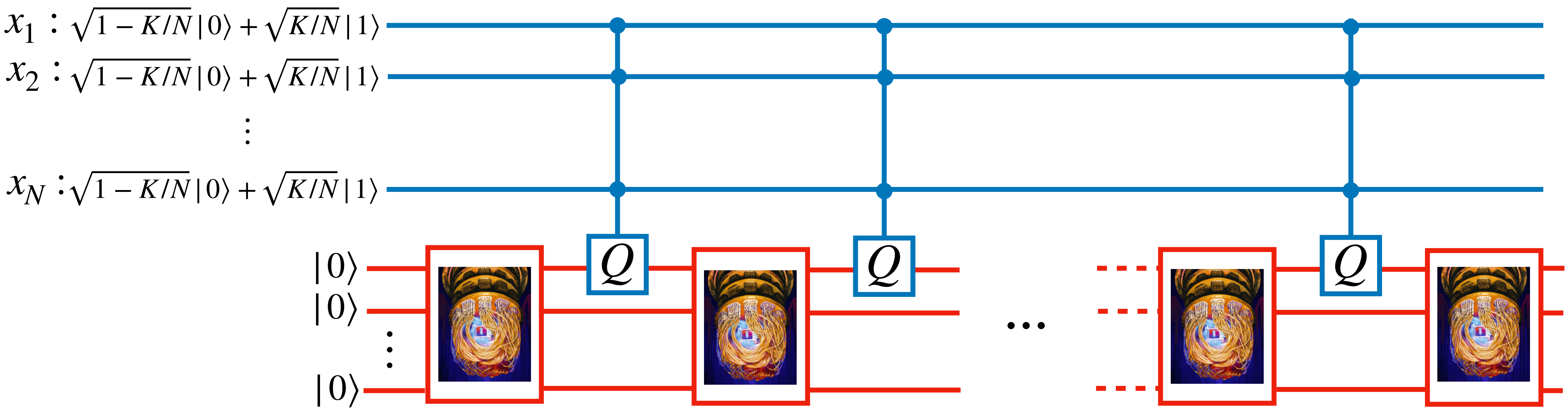
**[Zhandry'19]:**

- A quantum "recording technique" that works when the input $x_1, \ldots, x_N$ is sampled from the **uniform distribution** on $[M]^N$.
- Motivations: security proofs in the quantum random oracle model.

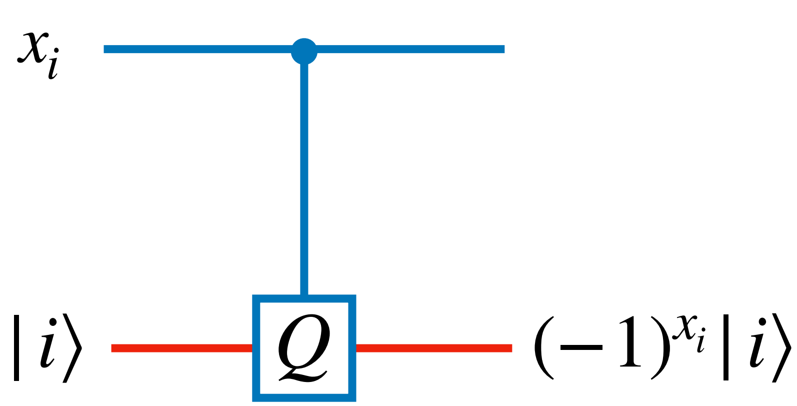
**Our contribution:**

- We generalize Zhandry's technique to the case where $x_1, \ldots, x_N$ is sampled from any **product distribution $D_1 \otimes \ldots \otimes D_N$** on $[M]^N$.
- We simplify the framework and the analysis of the method.

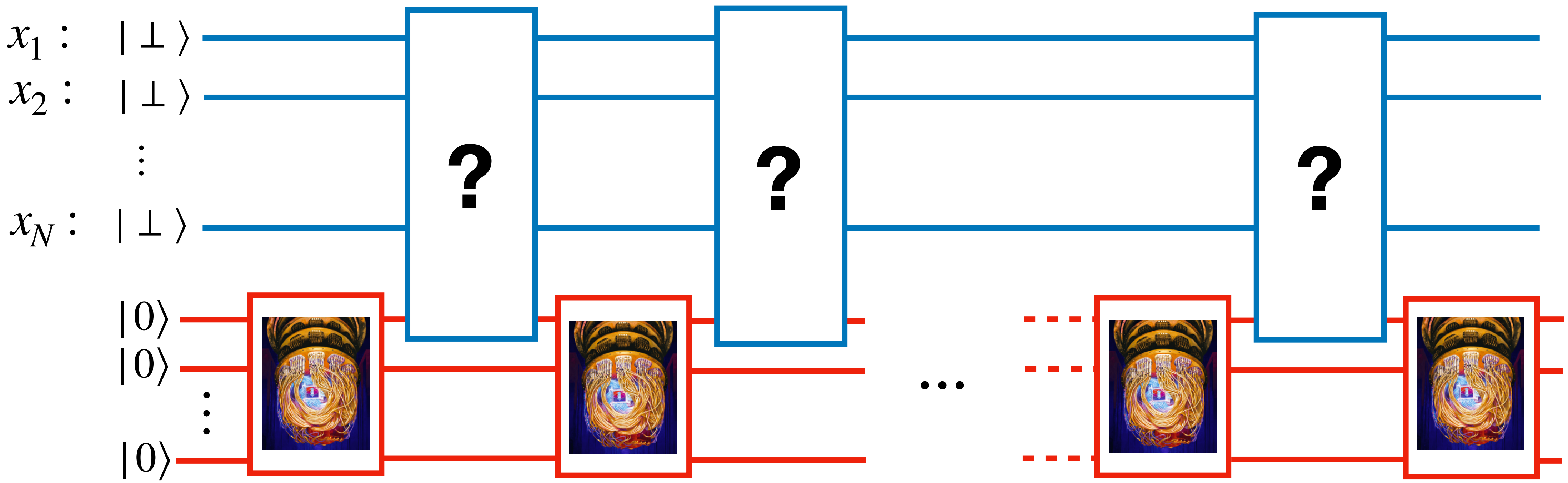
# Quantum Recording for K-Search



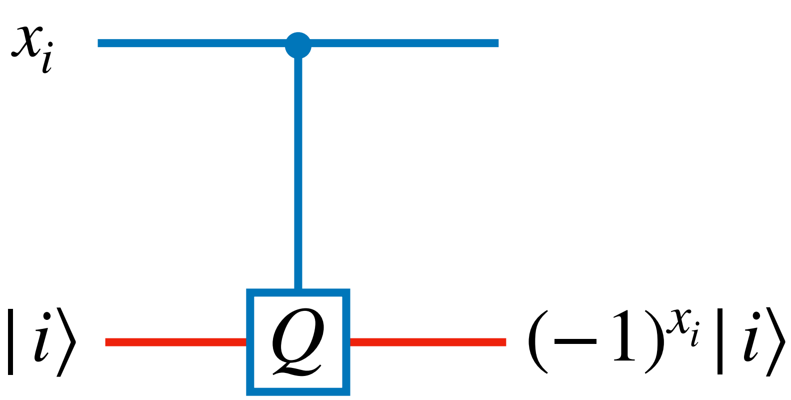

## Query Operator



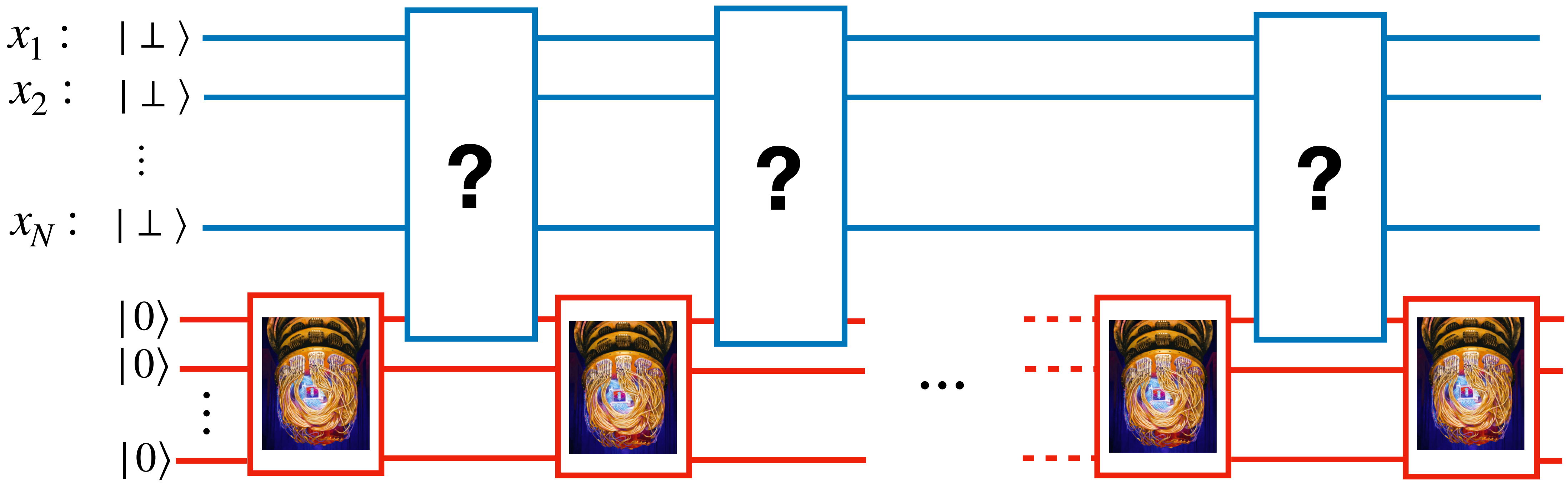
# Quantum Recording for K-Search




## Query Operator

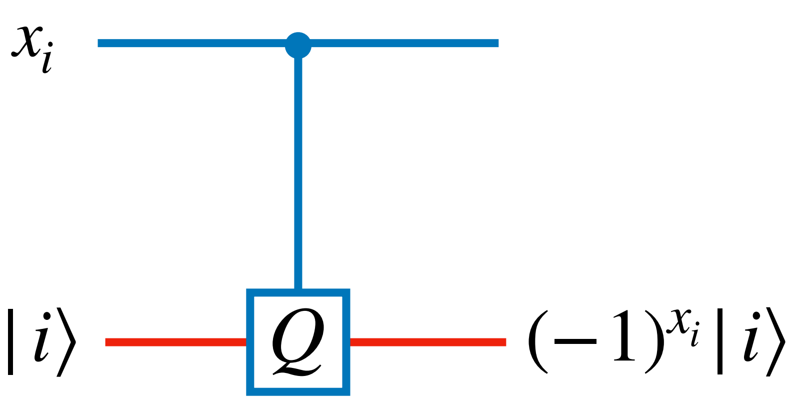


# Quantum Recording for K-Search

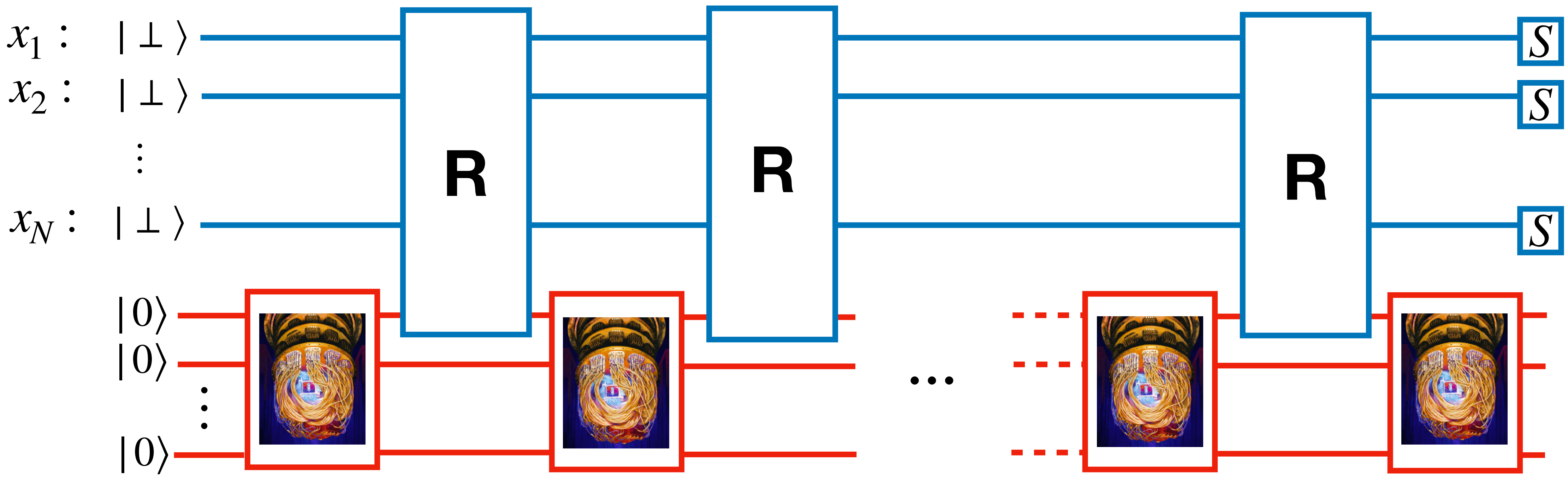



$$S|\perp\rangle = \sqrt{1-K/N}\,|0\rangle + \sqrt{K/N}\,|1\rangle$$

## Query Operator

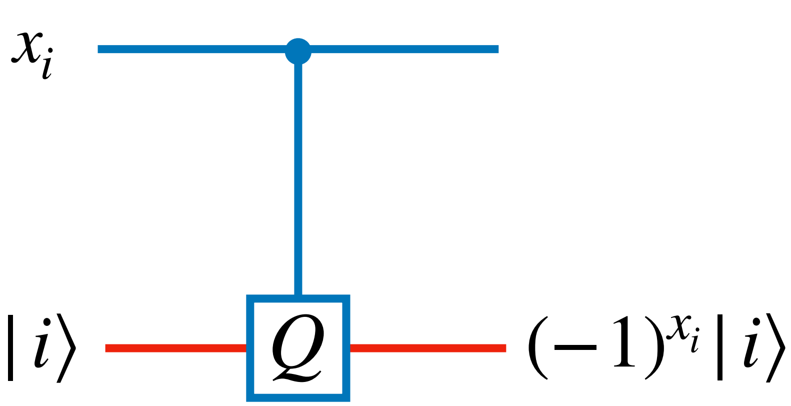


# Quantum Recording for K-Search

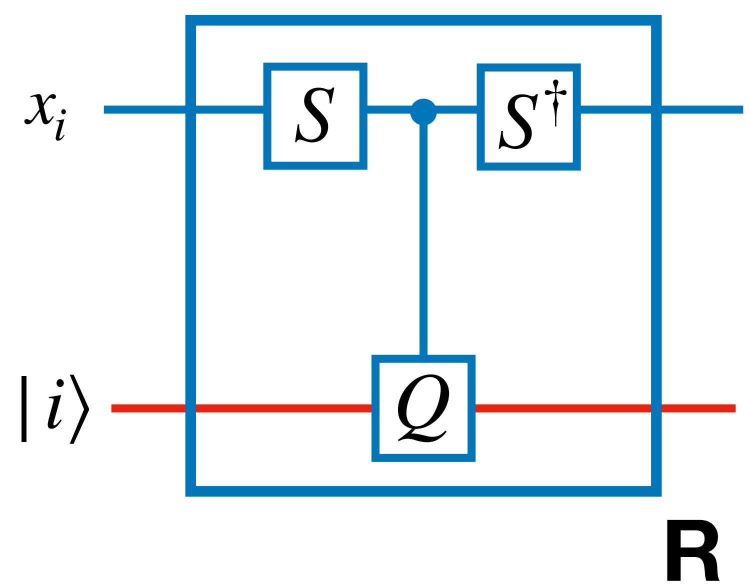



$$S|\perp\rangle = \sqrt{1-K/N}\,|0\rangle + \sqrt{K/N}\,|1\rangle$$

## Query Operator



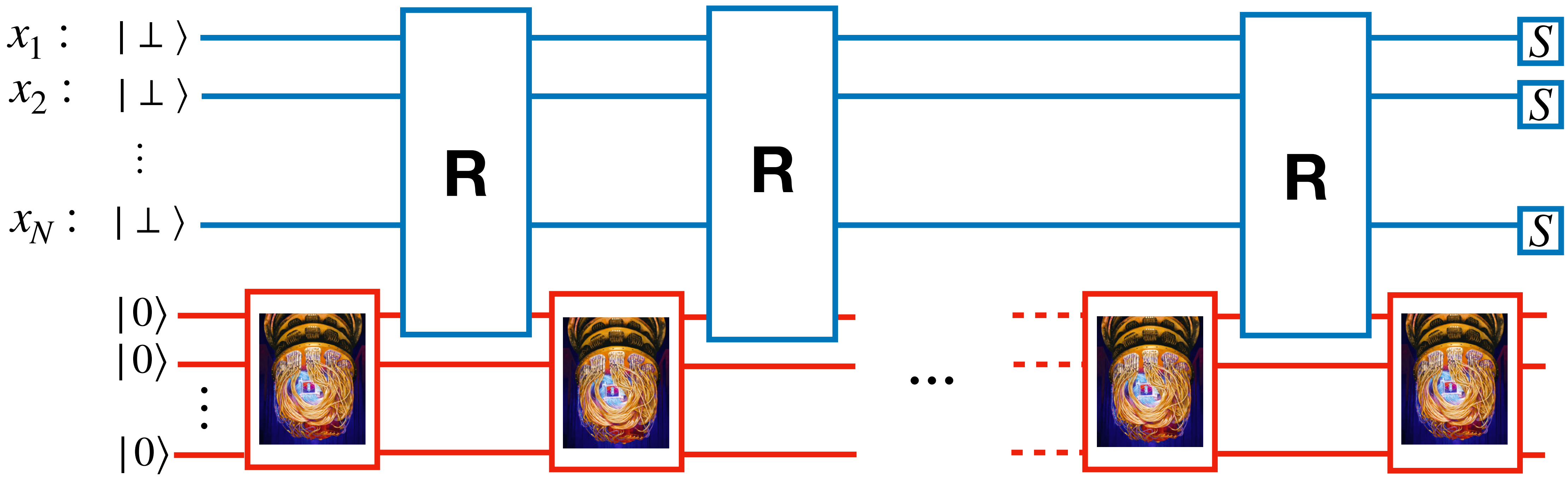

## Recording Query Operator




(where $(-1)^\perp = 1$)

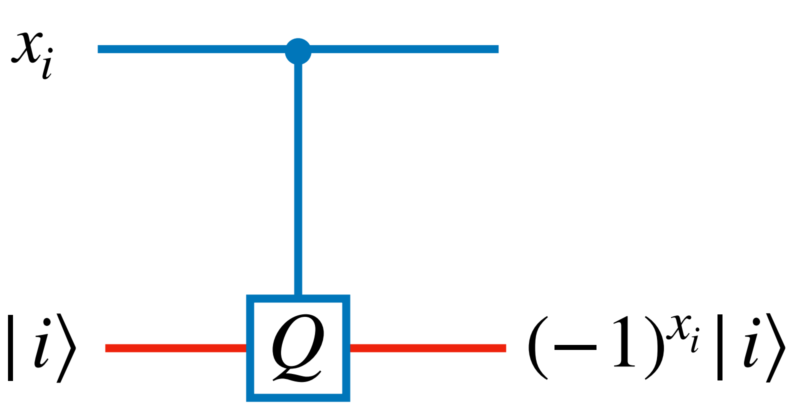
# Quantum Recording for K-Search



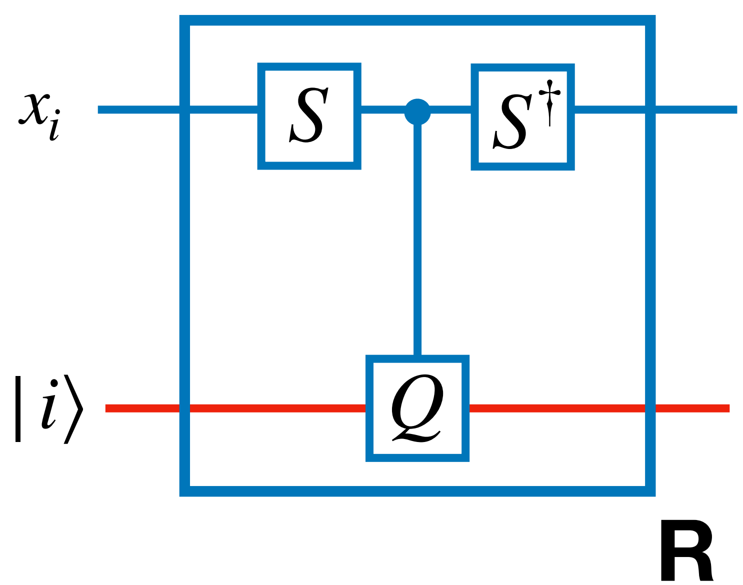



$$S|\perp\rangle = \sqrt{1 - K/N}\,|0\rangle + \sqrt{K/N}\,|1\rangle$$

## Query Operator



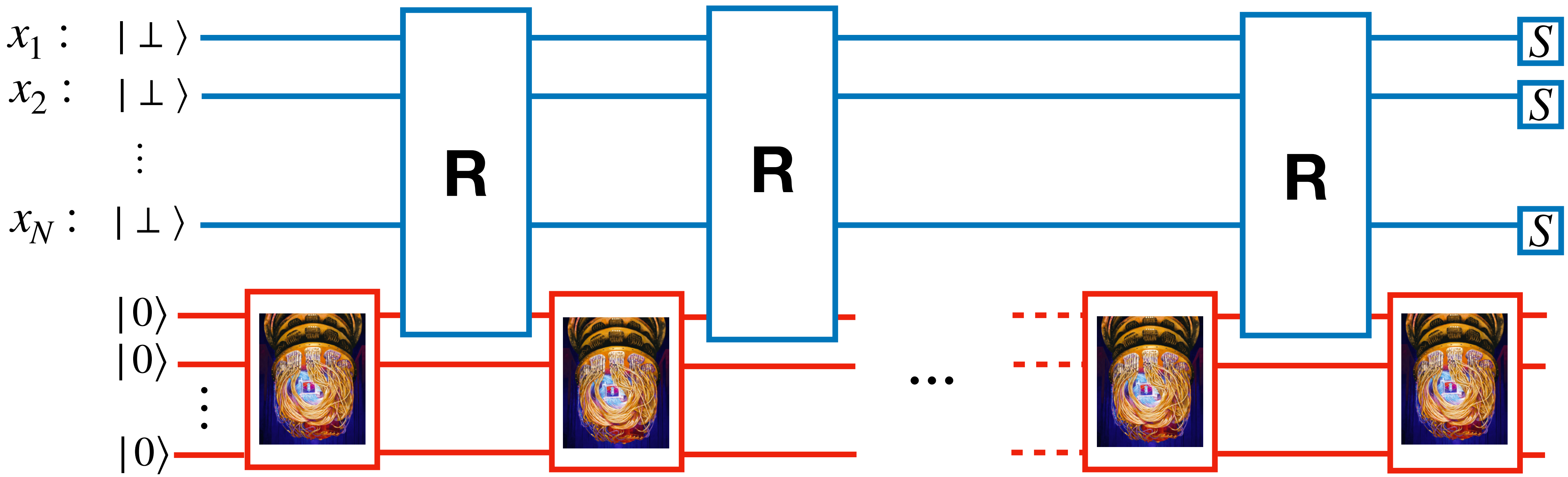

## Recording Query Operator




(where $(-1)^{\perp} = 1$)

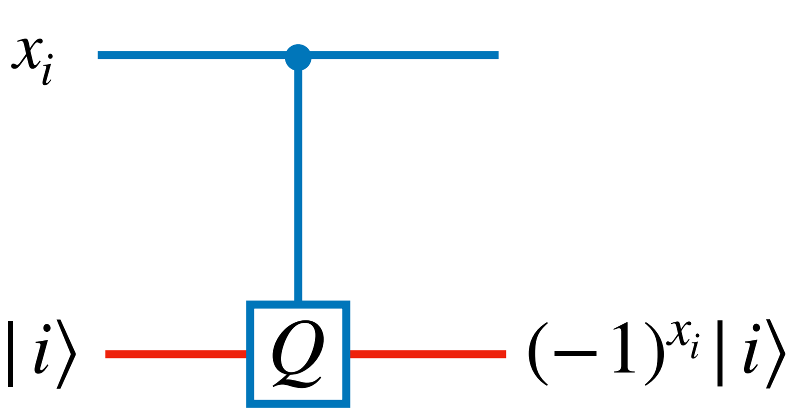
✓ We show that it makes no difference for the algorithm.

# Quantum Recording for K-Search

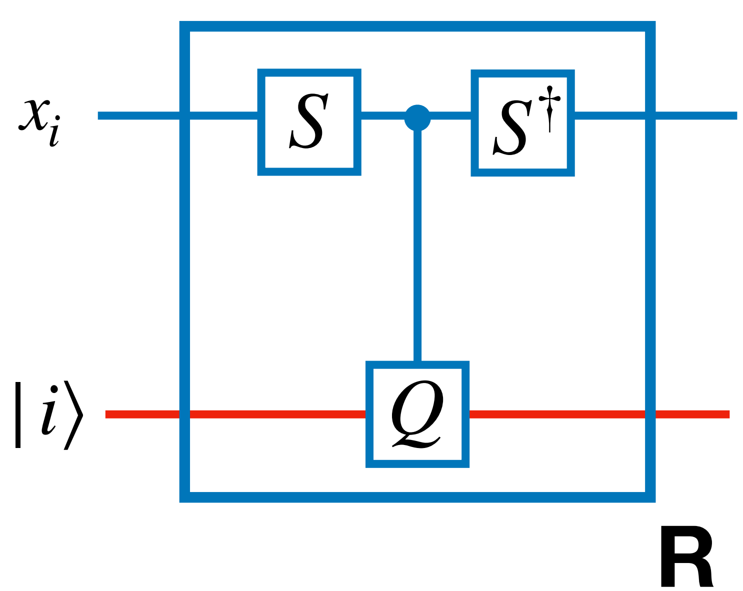



$$S|\perp\rangle = \sqrt{1-K/N}\,|0\rangle + \sqrt{K/N}\,|1\rangle$$

## Query Operator



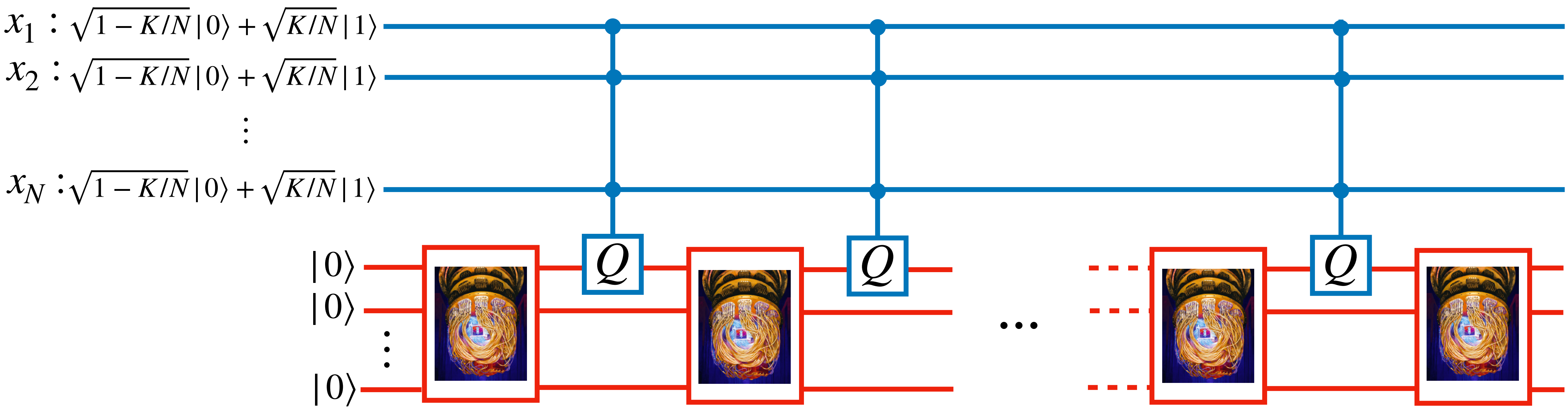

## Recording Query Operator




(where $(-1)^{\perp} = 1$)

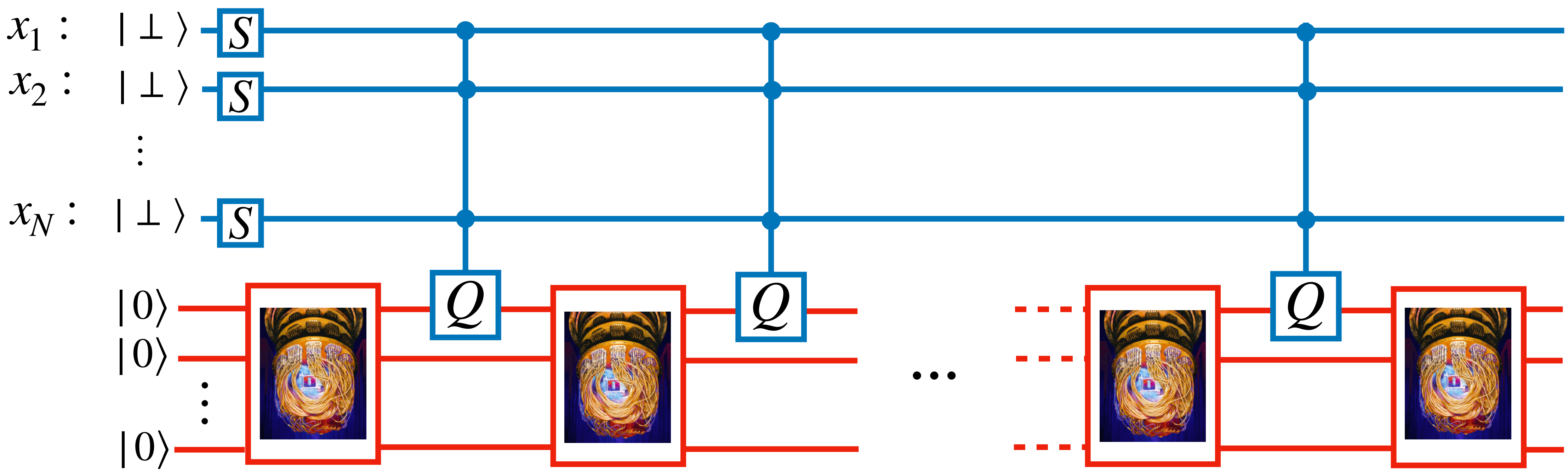
- ✓ We show that it makes no difference for the algorithm.
- ✓ We show that it "records" the 1's.

# Quantum Recording for K-Search




$$S|\perp\rangle = \sqrt{1-K/N}|0\rangle + \sqrt{K/N}|1\rangle$$

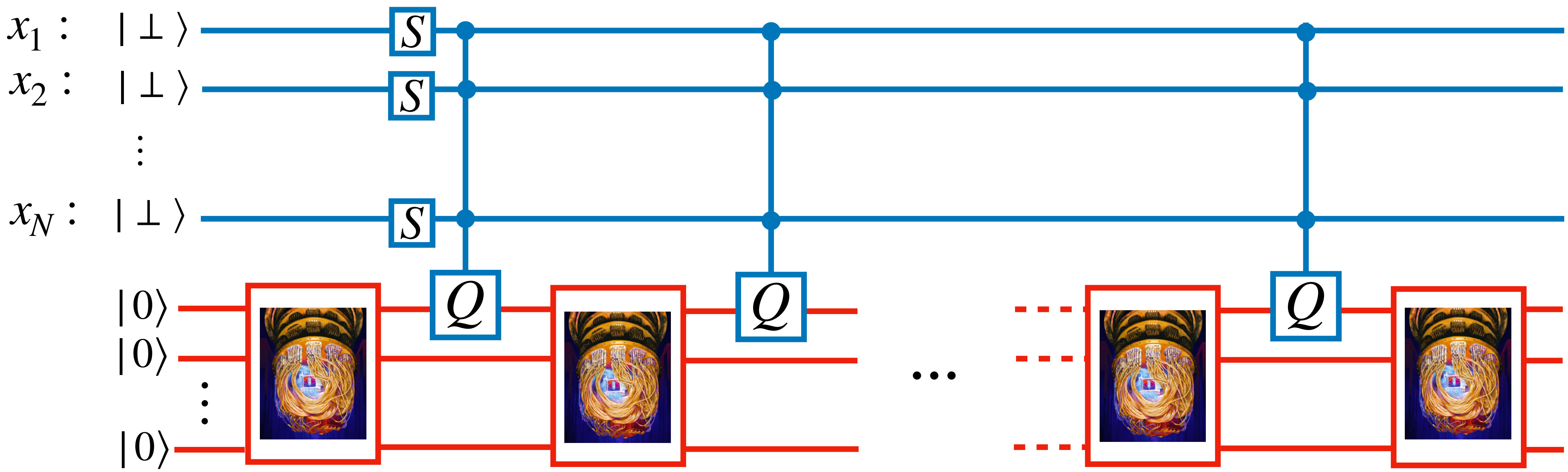
# Quantum Recording for K-Search




$$S|\perp\rangle = \sqrt{1-K/N}\,|0\rangle + \sqrt{K/N}\,|1\rangle$$

# Quantum Recording for K-Search

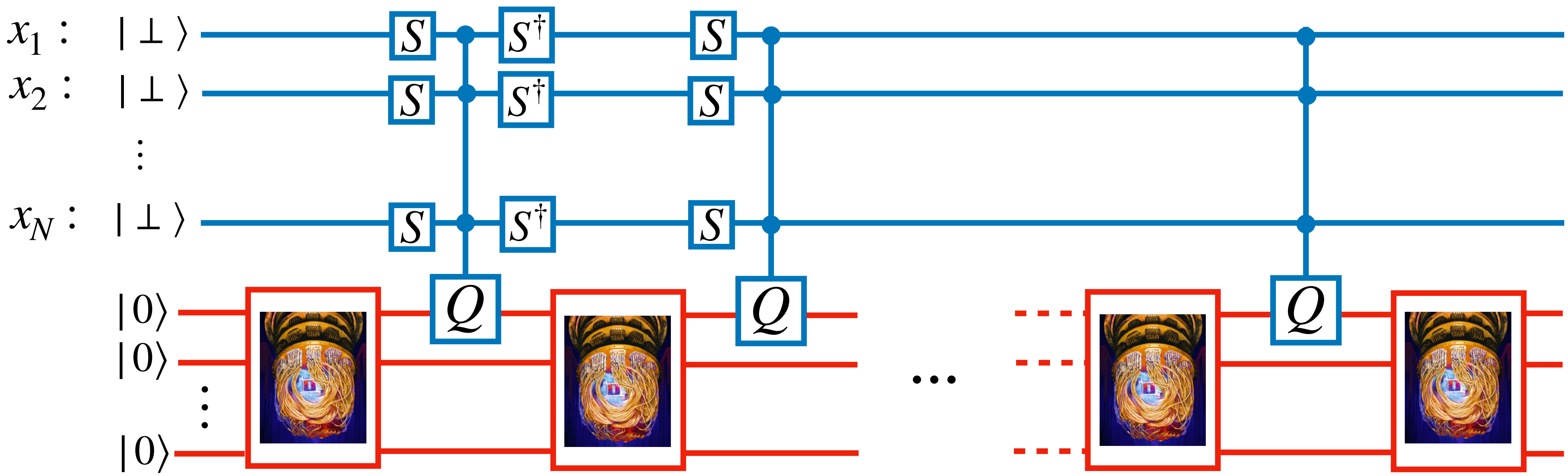





$$S|\perp\rangle = \sqrt{1 - K/N}\,|0\rangle + \sqrt{K/N}\,|1\rangle$$

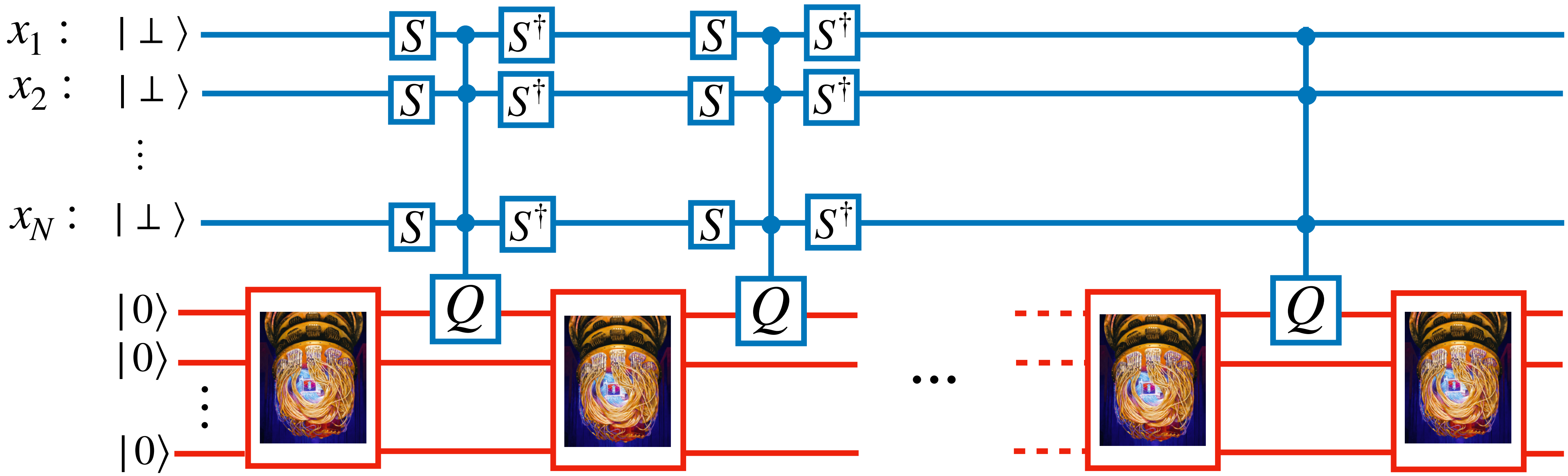
# Quantum Recording for K-Search






$$S|\perp\rangle = \sqrt{1 - K/N}\,|0\rangle + \sqrt{K/N}\,|1\rangle$$

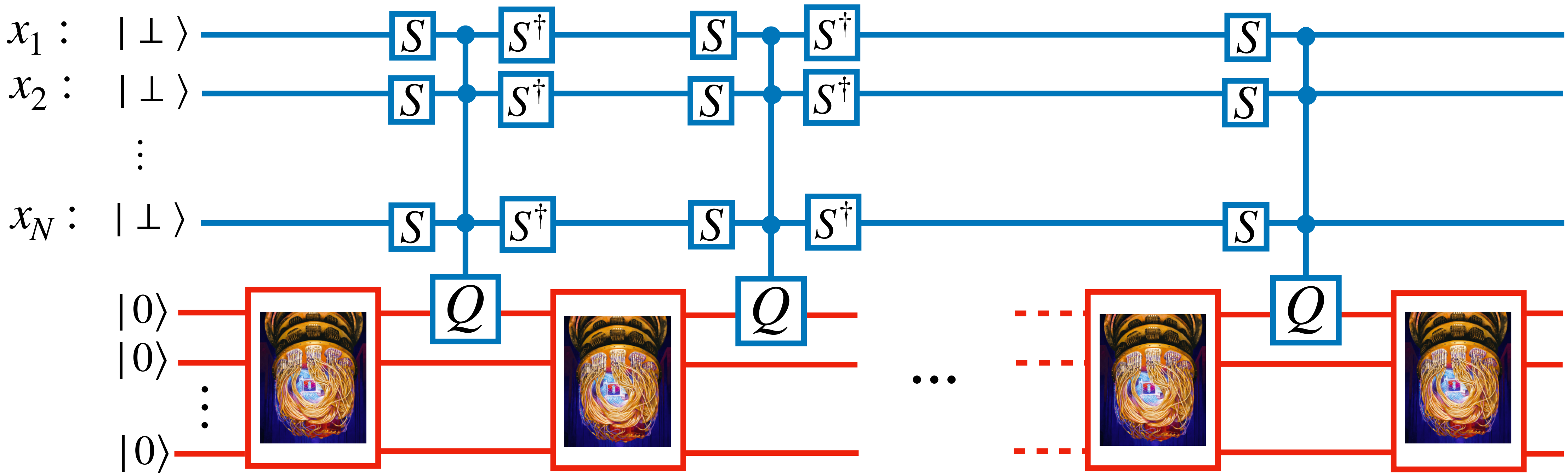
# Quantum Recording for K-Search




$$S|\perp\rangle = \sqrt{1 - K/N}\,|0\rangle + \sqrt{K/N}\,|1\rangle$$

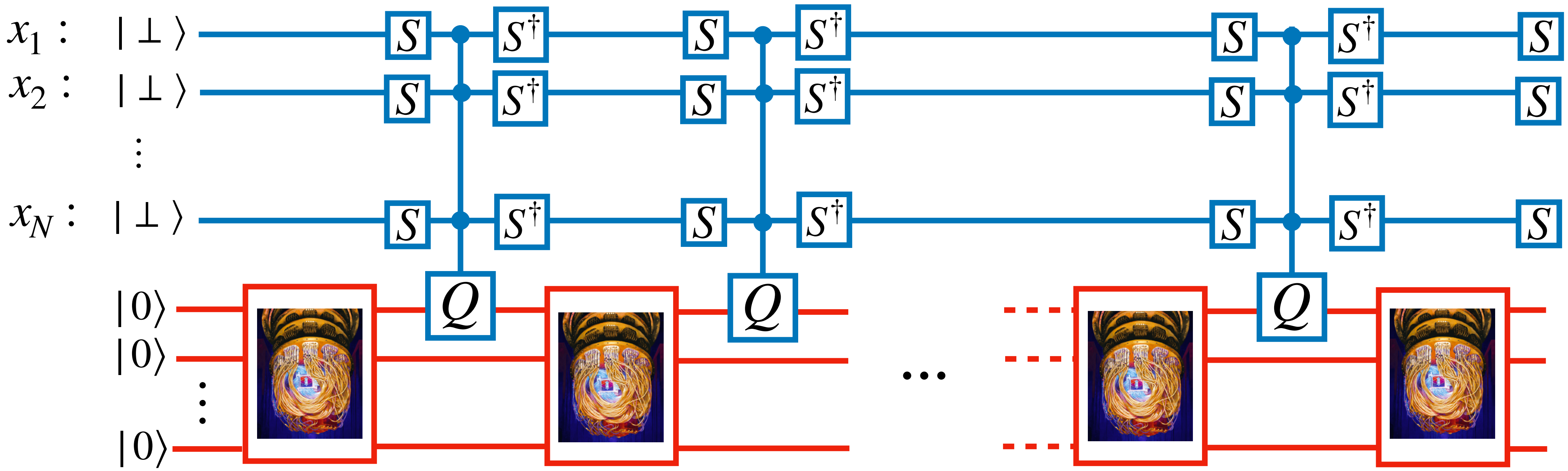
# Quantum Recording for K-Search





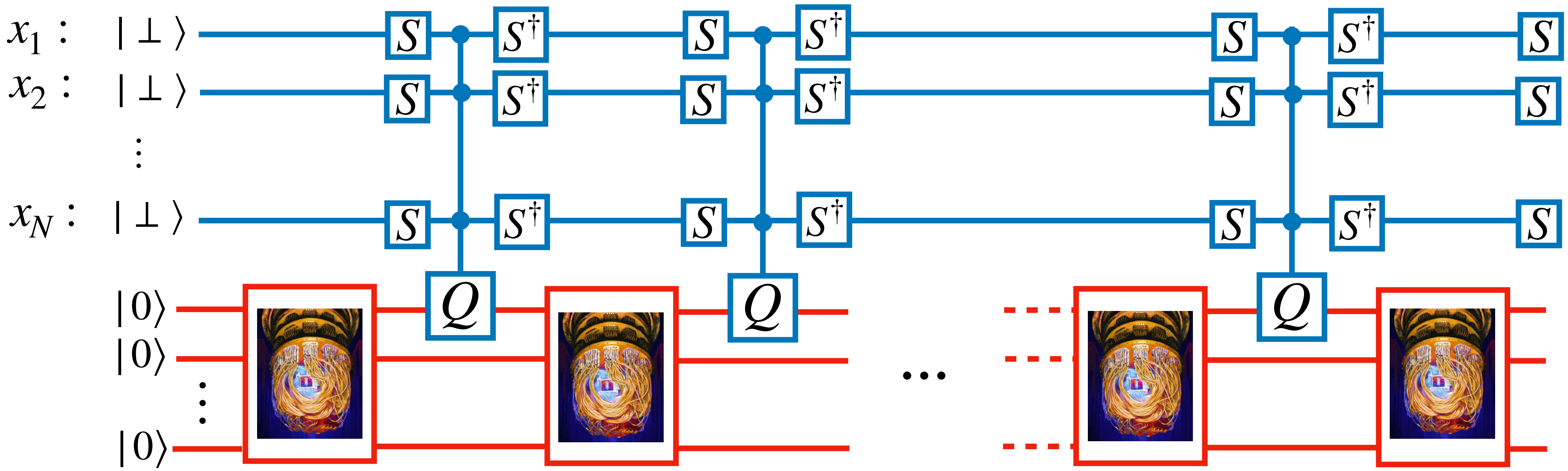

$$S|\perp\rangle = \sqrt{1 - K/N}\,|0\rangle + \sqrt{K/N}\,|1\rangle$$

# Quantum Recording for K-Search



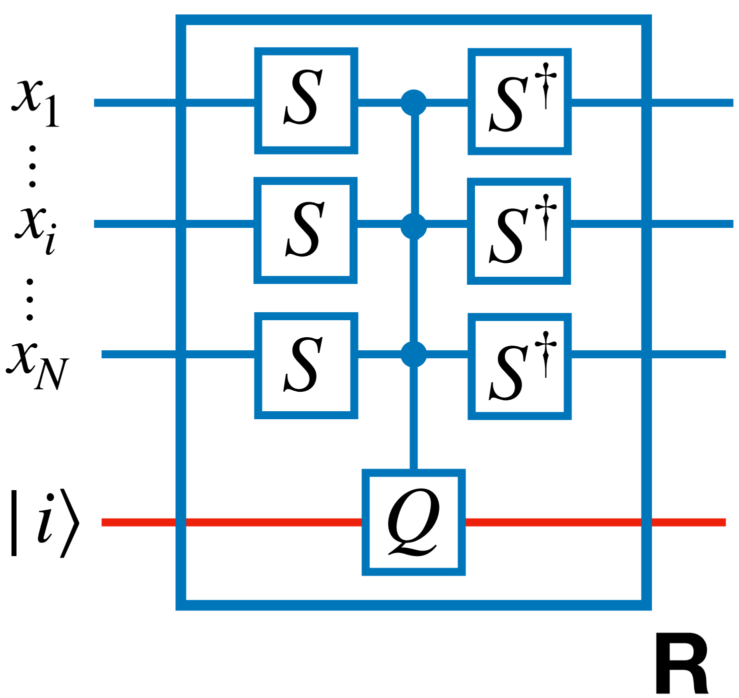

$$S|\perp\rangle = \sqrt{1-K/N}\,|0\rangle + \sqrt{K/N}\,|1\rangle$$

# Quantum Recording for K-Search



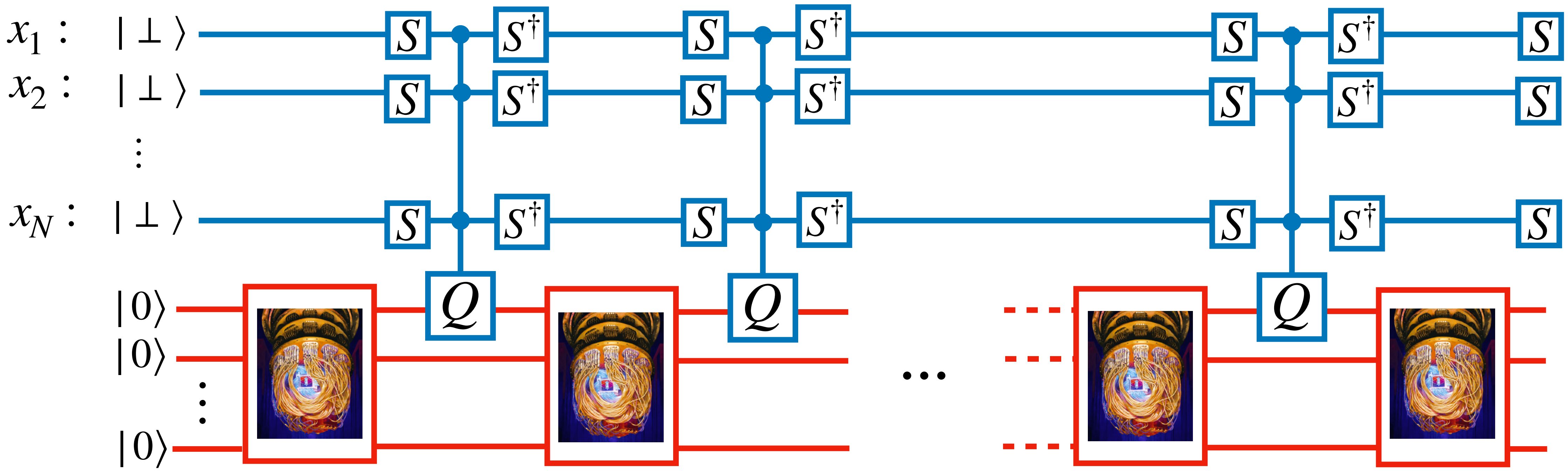

$$S|\perp\rangle = \sqrt{1 - K/N}\,|0\rangle + \sqrt{K/N}\,|1\rangle$$

## Recording Query Operator



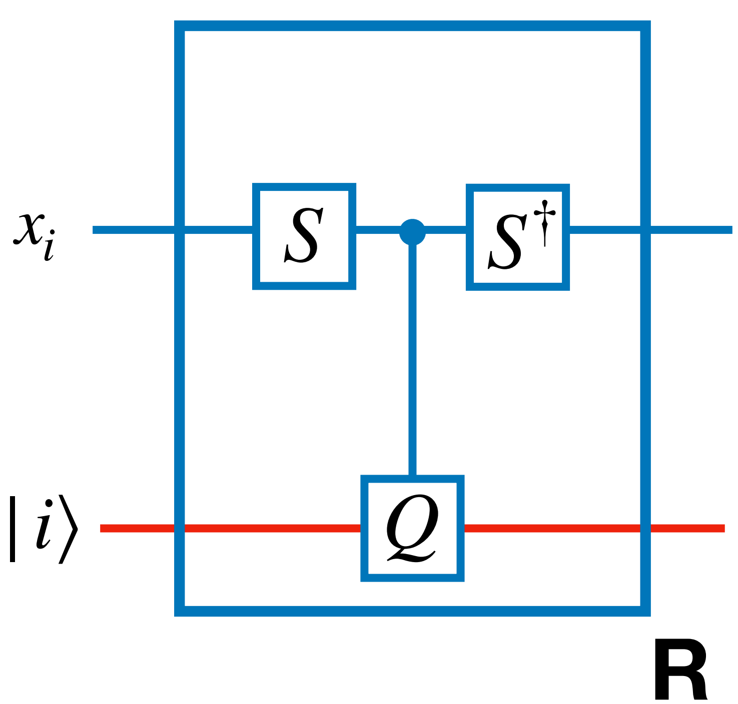
# Quantum Recording for K-Search





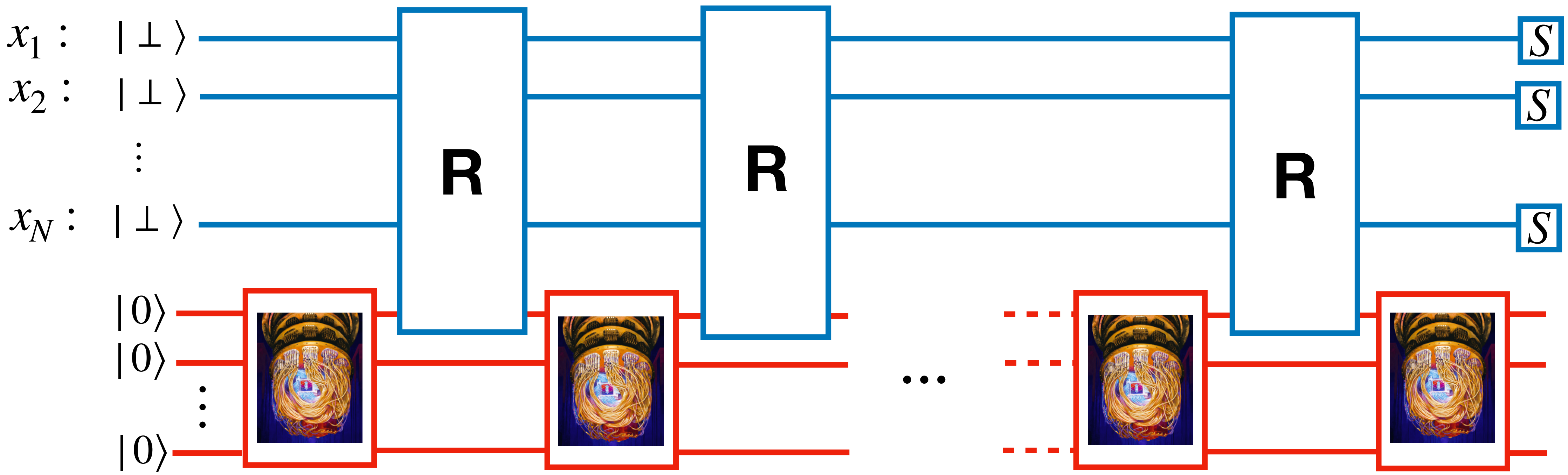

$$S|\perp\rangle = \sqrt{1 - K/N}\,|0\rangle + \sqrt{K/N}\,|1\rangle$$

## Recording Query Operator



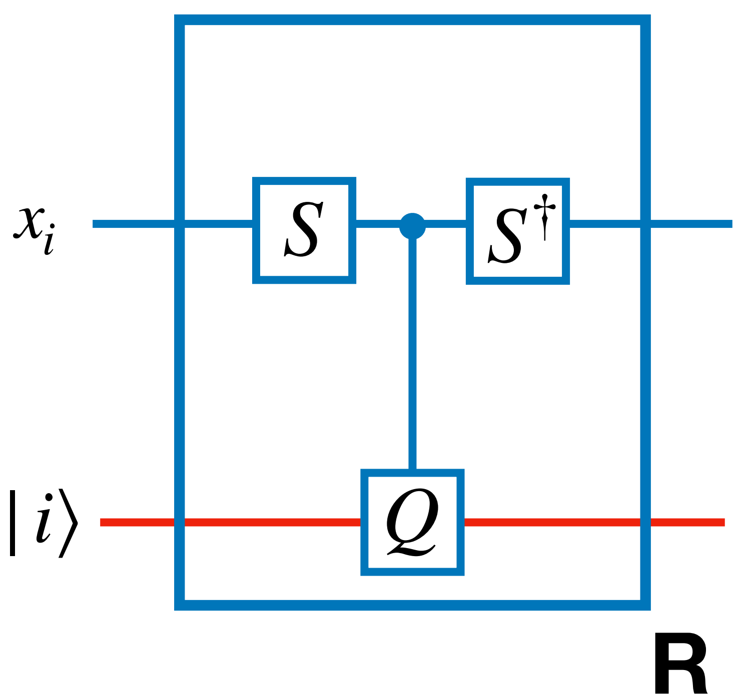

✓ If $x_j$ is not queried
it stays unchanged.

# Quantum Recording for K-Search



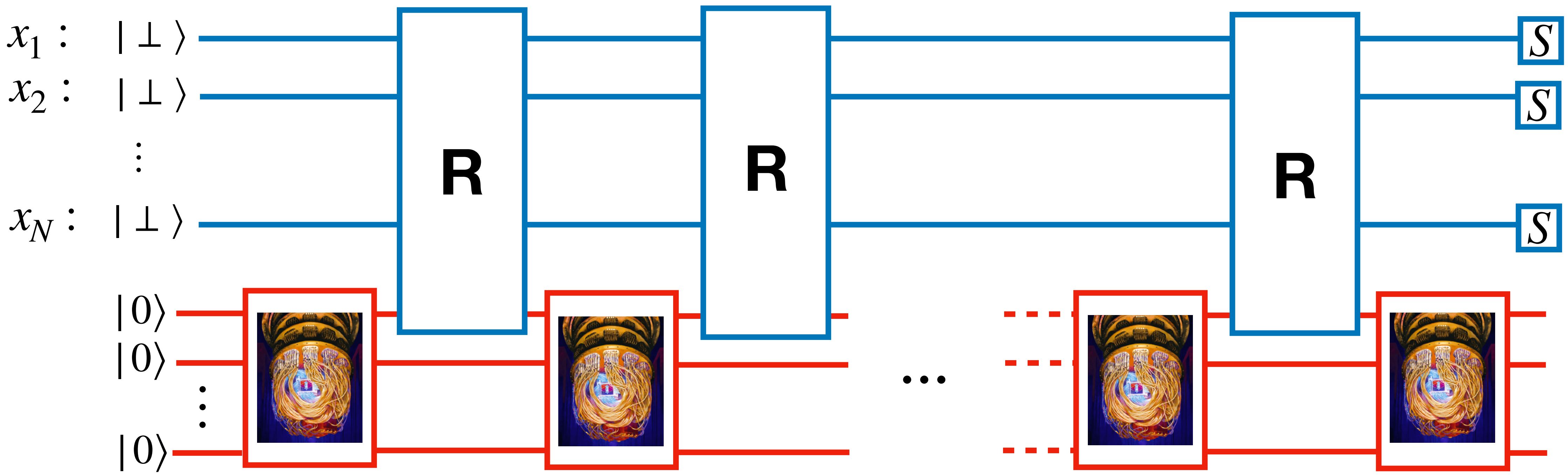

$$S|\perp\rangle = \sqrt{1 - K/N}\,|0\rangle + \sqrt{K/N}\,|1\rangle$$

## Recording Query Operator



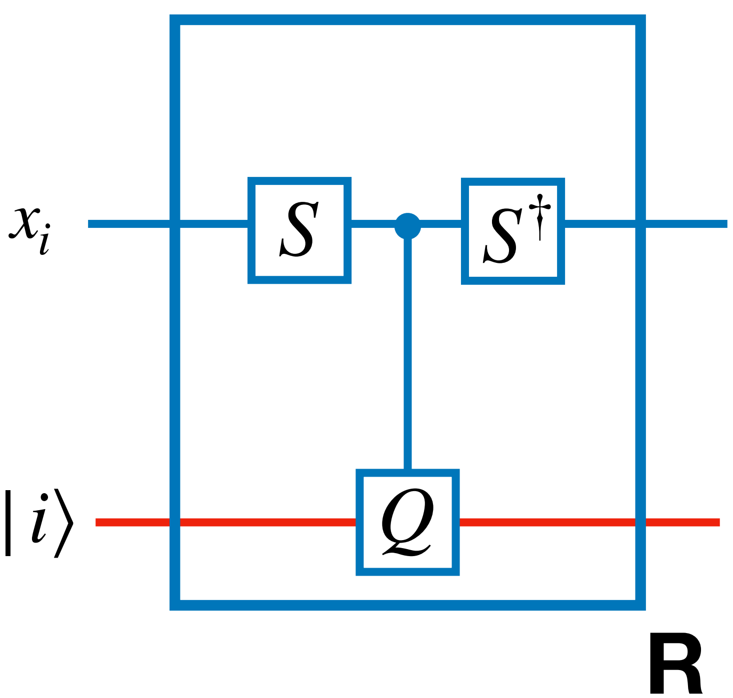

✓ If $x_j$ is not queried
it stays unchanged.

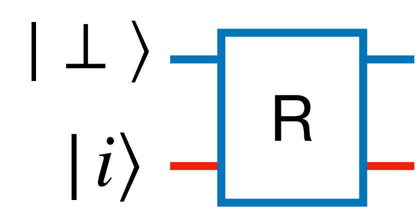
# Quantum Recording for K-Search

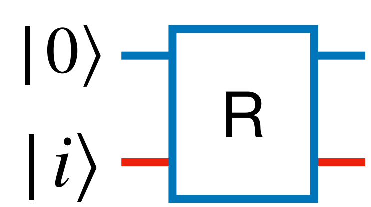


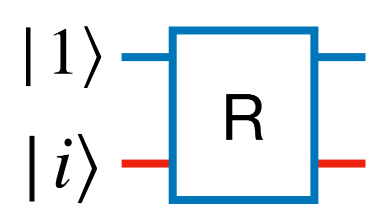

$$S|\perp\rangle = \sqrt{1-K/N}\,|0\rangle + \sqrt{K/N}\,|1\rangle$$

## Recording Query Operator



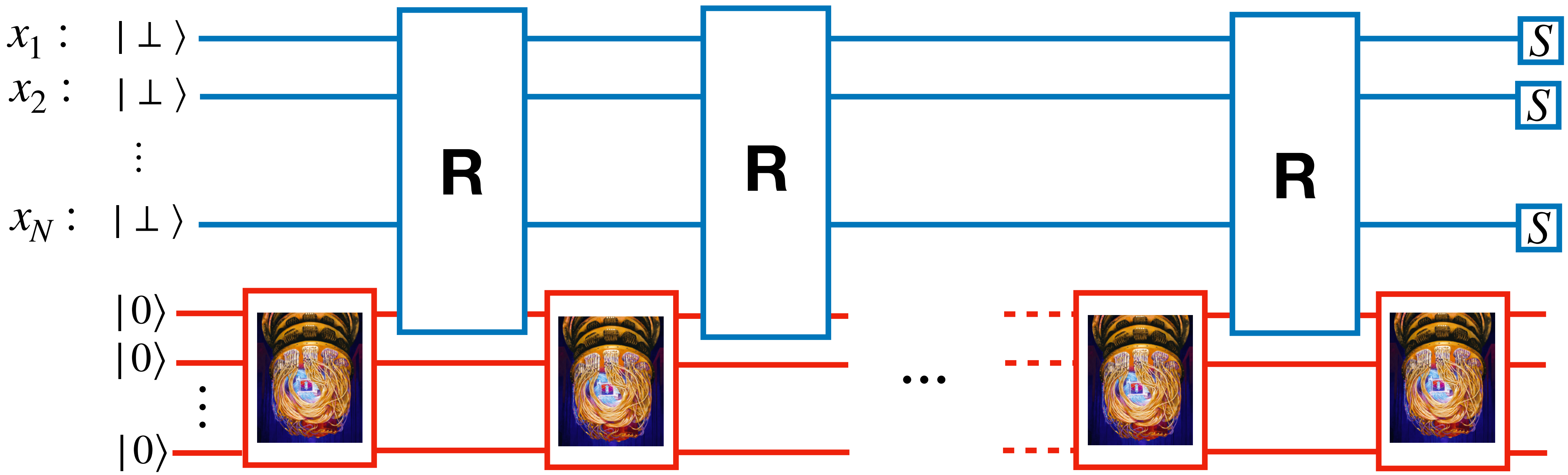



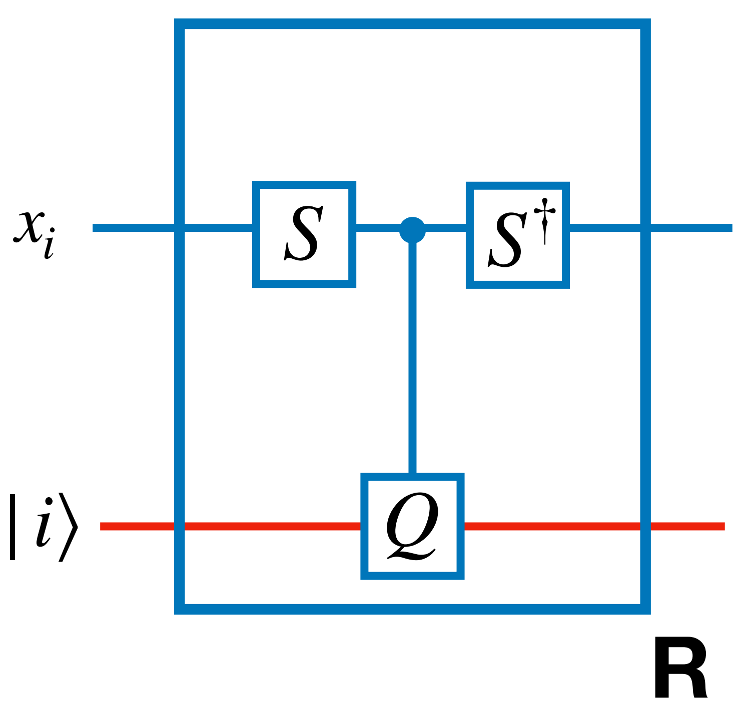
# Quantum Recording for K-Search

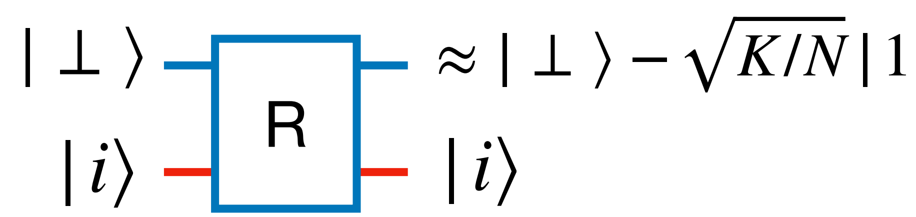


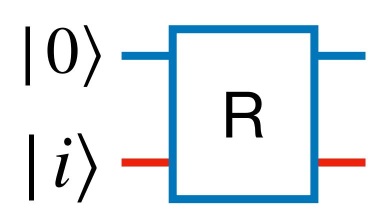


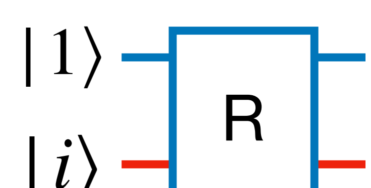

$$S|\perp\rangle = \sqrt{1 - K/N}\,|0\rangle + \sqrt{K/N}\,|1\rangle$$

## Recording Query Operator

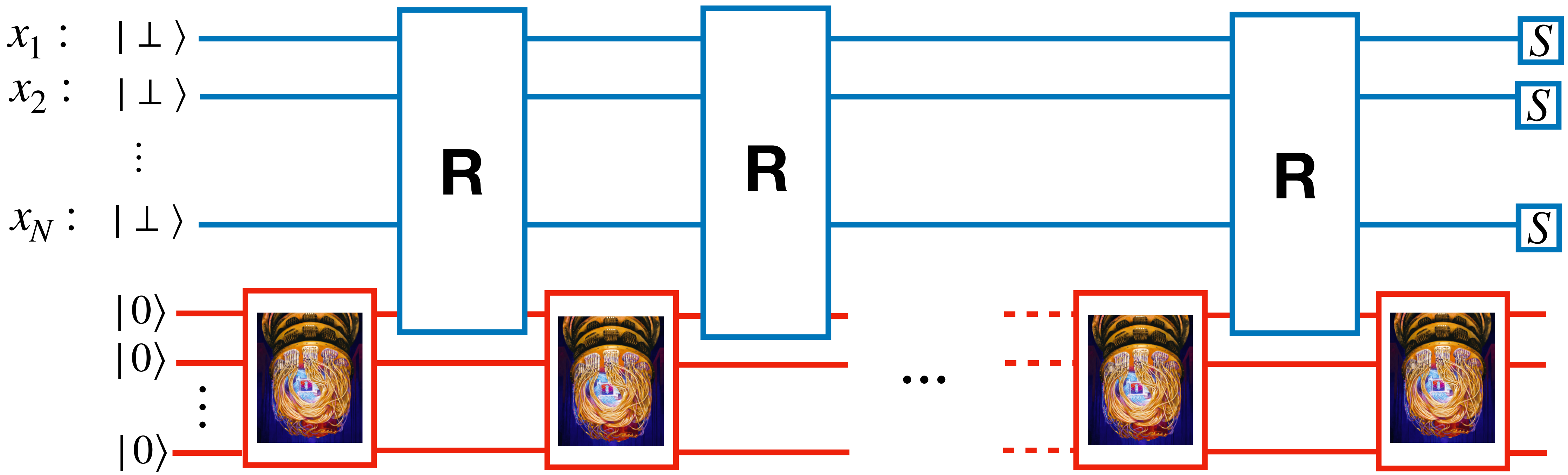


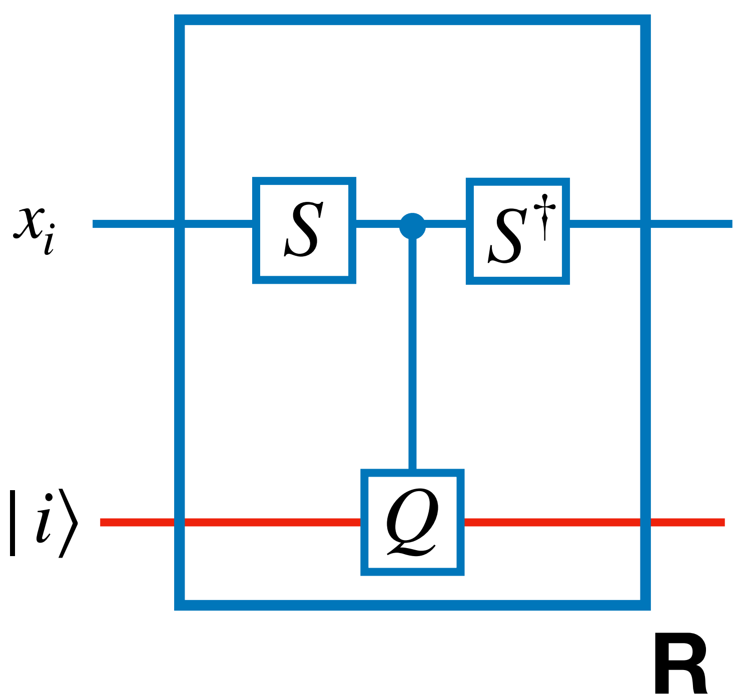



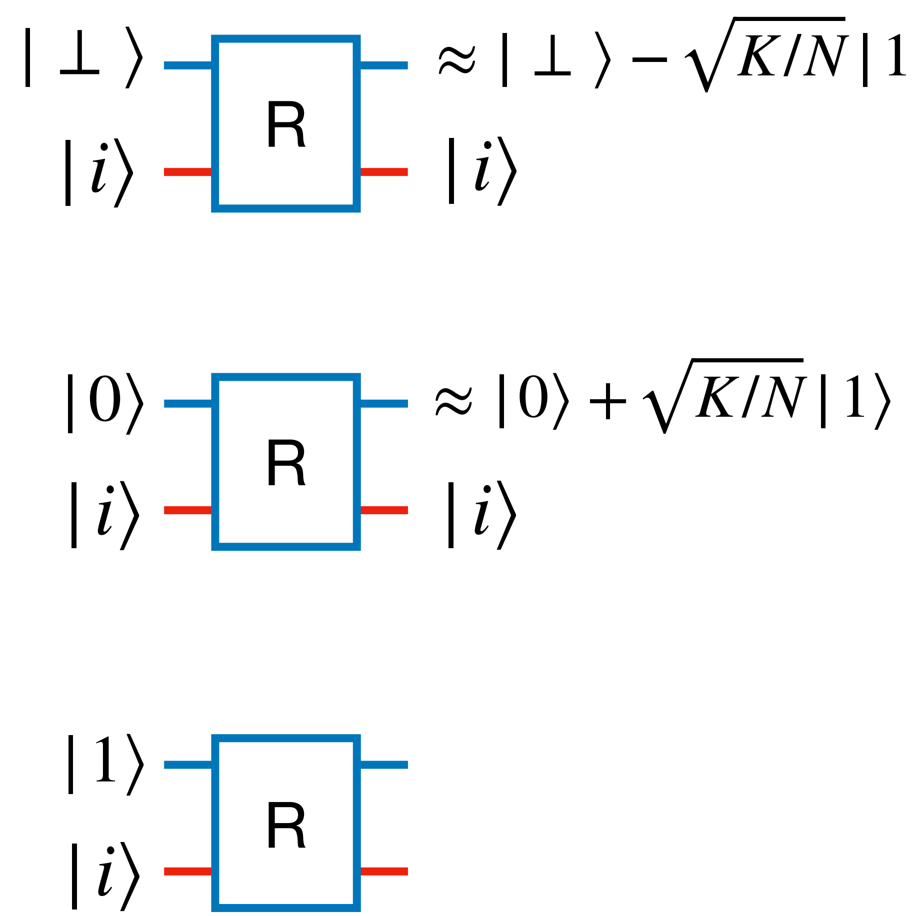
# Quantum Recording for K-Search



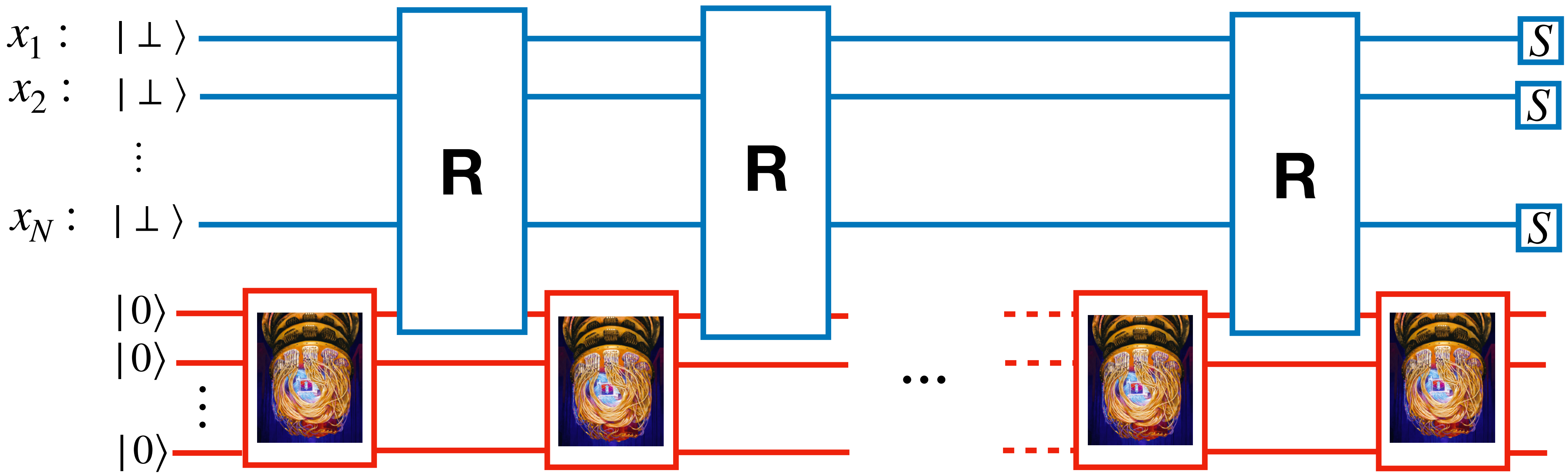

$$S|\perp\rangle = \sqrt{1 - K/N}\,|0\rangle + \sqrt{K/N}\,|1\rangle$$

## Recording Query Operator



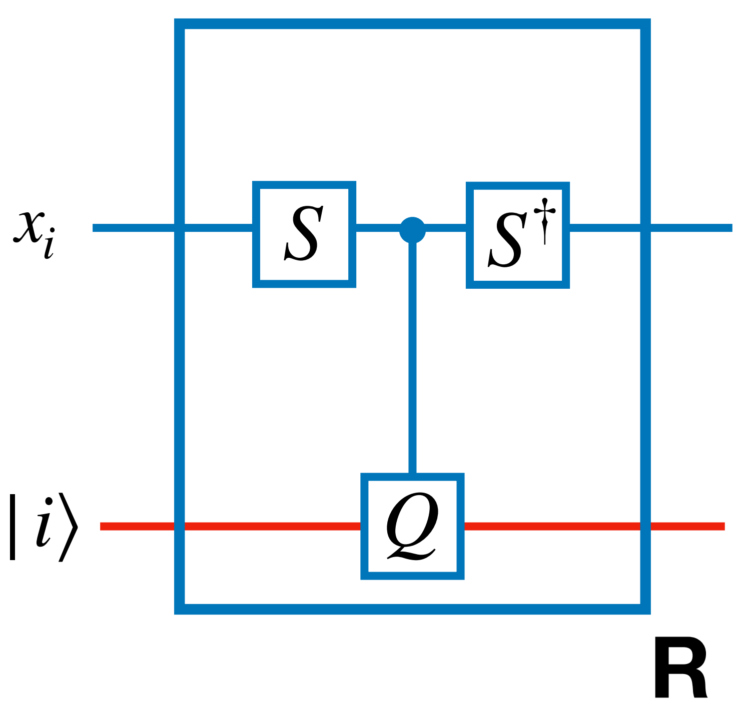



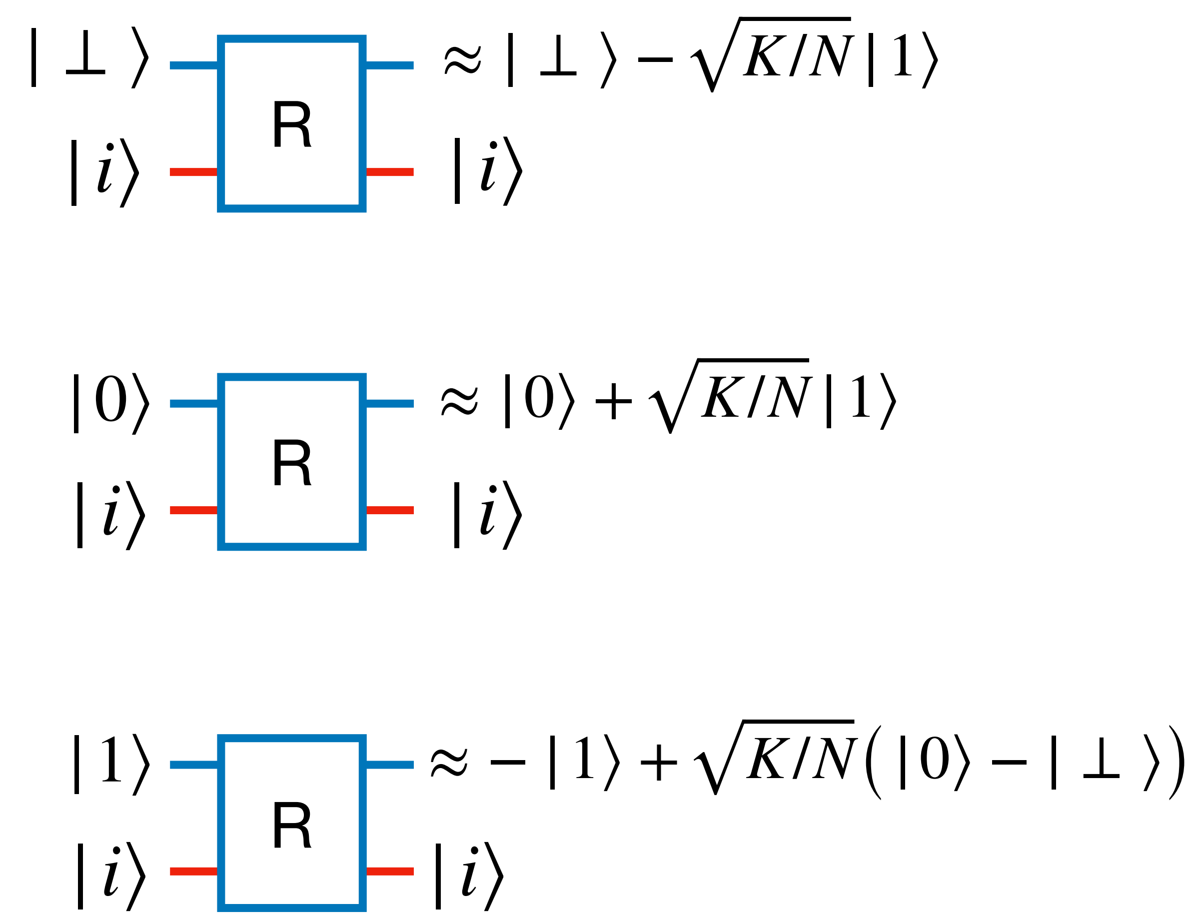
# Quantum Recording for K-Search



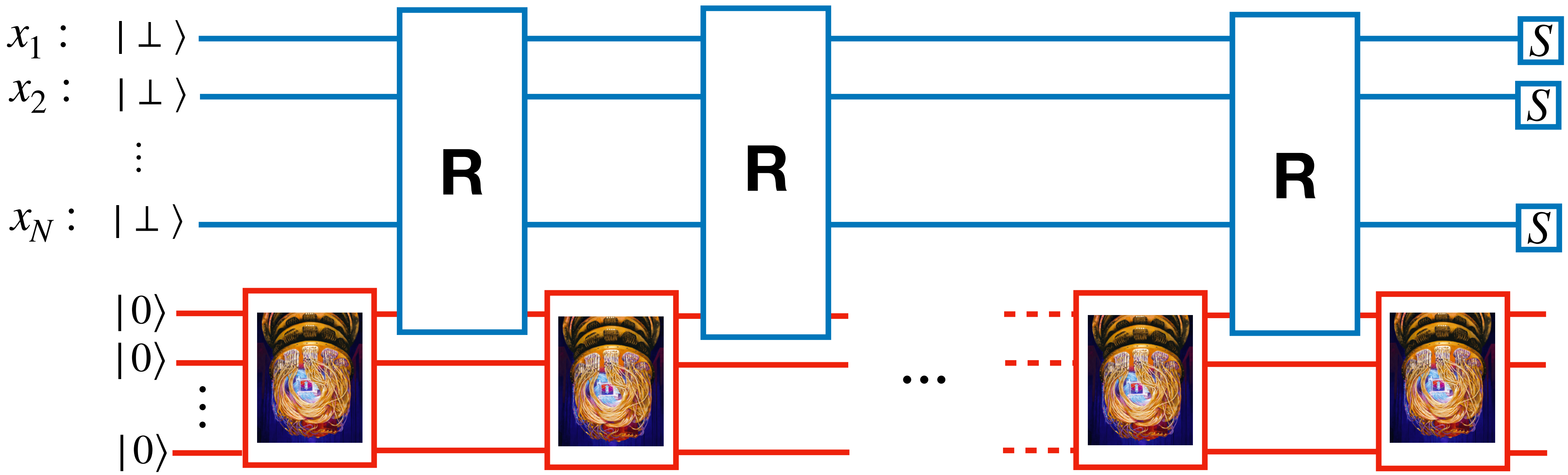

$$S|\perp\rangle = \sqrt{1-K/N}\,|0\rangle + \sqrt{K/N}\,|1\rangle$$

## Recording Query Operator



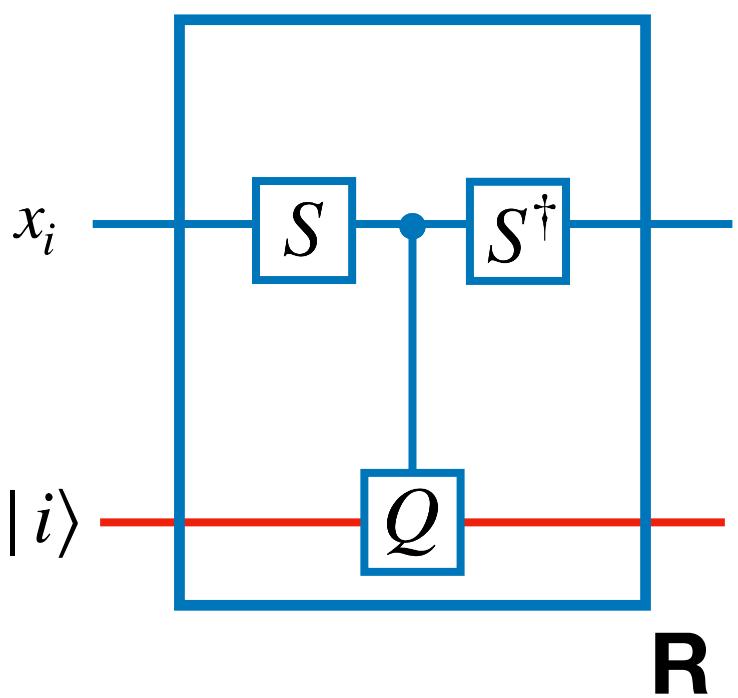



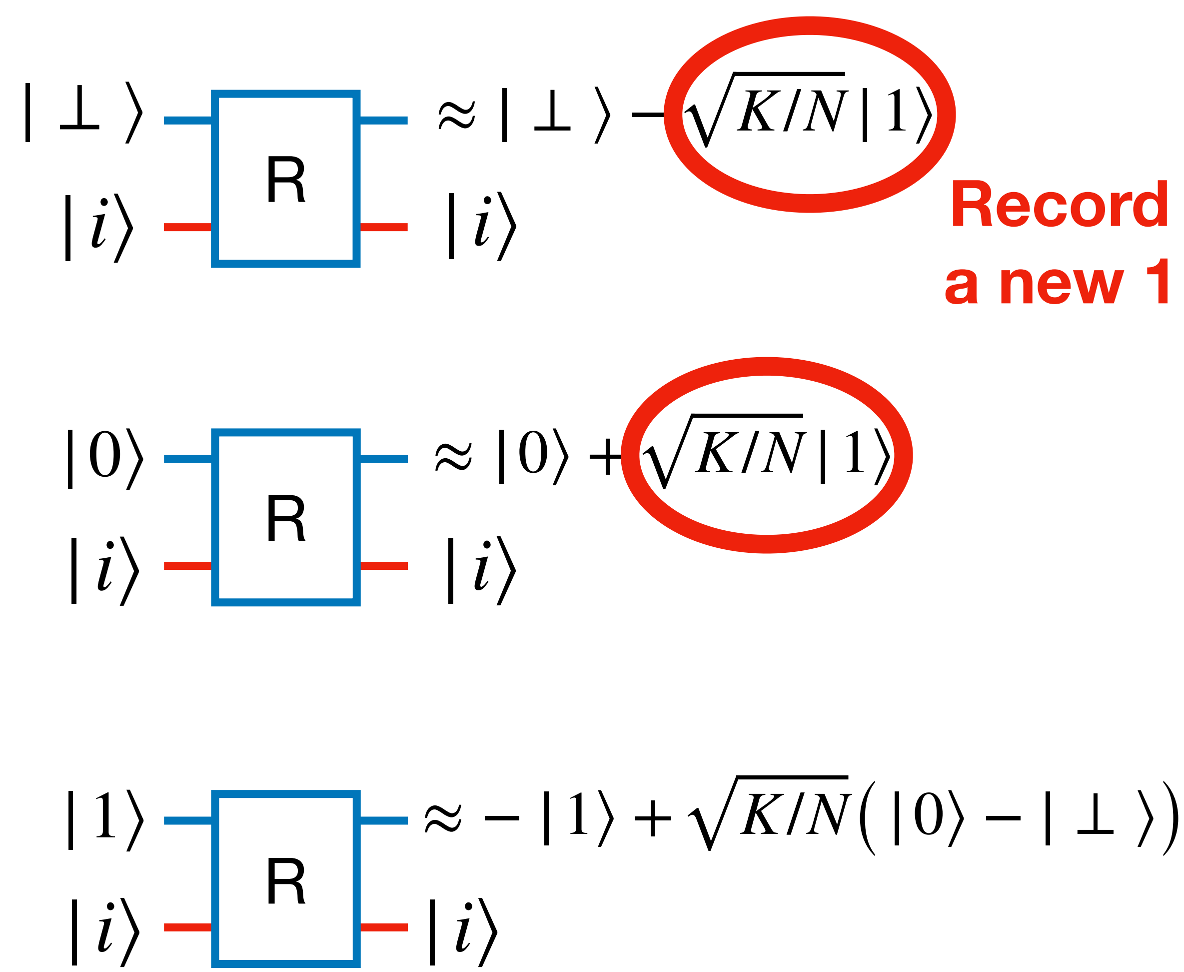
# Quantum Recording for K-Search




$$S|\perp\rangle = \sqrt{1-K/N}|0\rangle + \sqrt{K/N}|1\rangle$$

## Recording Query Operator



| | Classical Recording | Quantum Recording |
|---|---|---|
| **K-Search** | Probability to have recorded at least K/2 ones $\leq \binom{T}{K/2}\left(\frac{K}{N}\right)^{K/2}$ | |

| | Classical Recording | Quantum Recording |
| --- | --- | --- |
| **K-Search** | **Probability** to have recorded at least K/2 ones $\leq \binom{T}{K/2}\left(\frac{K}{N}\right)^{K/2}$ | **Amplitude** of the states that have recorded at least K/2 ones $\leq \binom{T}{K/2}\left(4\sqrt{\frac{K}{N}}\right)^{K/2}$ |

# Quantum Recording



| | Classical Recording | Quantum Recording |
|---|---|---|
| **K-Search** | **Probability** to have recorded at least K/2 ones $\leq \binom{T}{K/2}\left(\frac{K}{N}\right)^{K/2}$ | **Amplitude** of the states that have recorded at least K/2 ones $\leq \binom{T}{K/2}\left(4\sqrt{\frac{K}{N}}\right)^{K/2}$ |
| **K-Collision Pairs** | **Probability** to have recorded at least K/2 (disjoint) collisions $\leq \binom{T}{K/2}\left(\frac{T}{N}\right)^{K/2}$ | |

| | Classical Recording | Quantum Recording |
|---|---|---|
| **K-Search** | **Probability** to have recorded at least K/2 ones $\leq \binom{T}{K/2}\left(\frac{K}{N}\right)^{K/2}$ | **Amplitude** of the states that have recorded at least K/2 ones $\leq \binom{T}{K/2}\left(4\sqrt{\frac{K}{N}}\right)^{K/2}$ |
| **K-Collision Pairs** | **Probability** to have recorded at least K/2 (disjoint) collisions $\leq \binom{T}{K/2}\left(\frac{T}{N}\right)^{K/2}$ | **Amplitude** of the states that have recorded at least K/2 (disjoint) collisions $\leq \binom{T}{K/2}\left(4\sqrt{\frac{T}{N}}\right)^{K/2}$ |

# Quantum Recording



| | Classical Recording | Quantum Recording |
|---|---|---|
| **K-Search** | **Probability** to have recorded at least K/2 ones<br><br>$\leq 2^{-\Omega(K)}$ when $T \leq O(N)$ | **Amplitude** of the states that have recorded at least K/2 ones<br><br>$\leq 2^{-\Omega(K)}$ when $T \leq O\left(\sqrt{NK}\right)$ |
| **K-Collision Pairs** | **Probability** to have recorded at least K/2 (disjoint) collisions<br><br>$\leq 2^{-\Omega(K)}$ when $T \leq O\left(\sqrt{NK}\right)$ | **Amplitude** of the states that have recorded at least K/2 (disjoint) collisions<br><br>$\leq 2^{-\Omega(K)}$ when $T \leq O\left(K^{2/3}N^{1/3}\right)$ |

# Conclusion

# Open Problems:

- Extend the quantum recording technique to **non-product** distributions?

  Example: uniform distribution over the symmetric group.

- Improve the tradeoff for finding $\tilde{\Theta}(N)$ Collision Pairs to $T^2S \geq \Omega(N^3)$,

  or find a quantum algorithm with $T^3S \leq O(N^4)$?

- New lower bounds by recording queries? **Triangles Finding**?

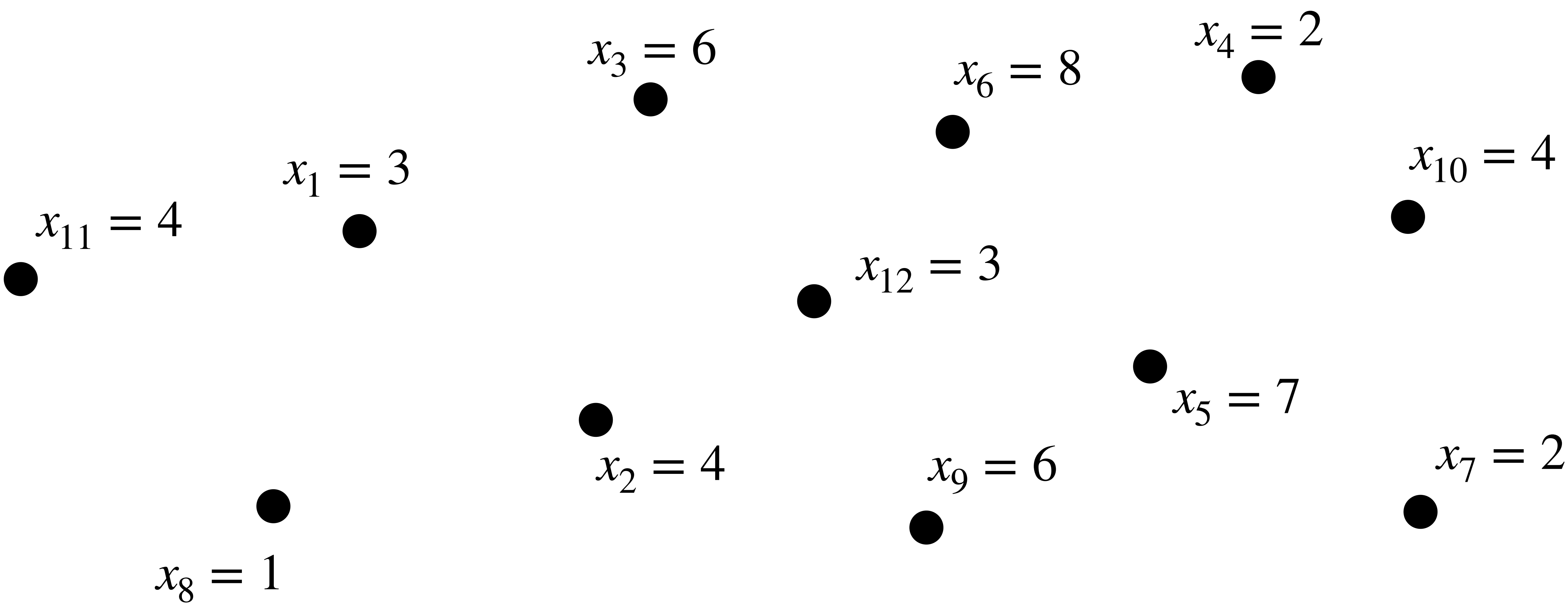
**arXiv: 2002.08944**

# Supplementary slides

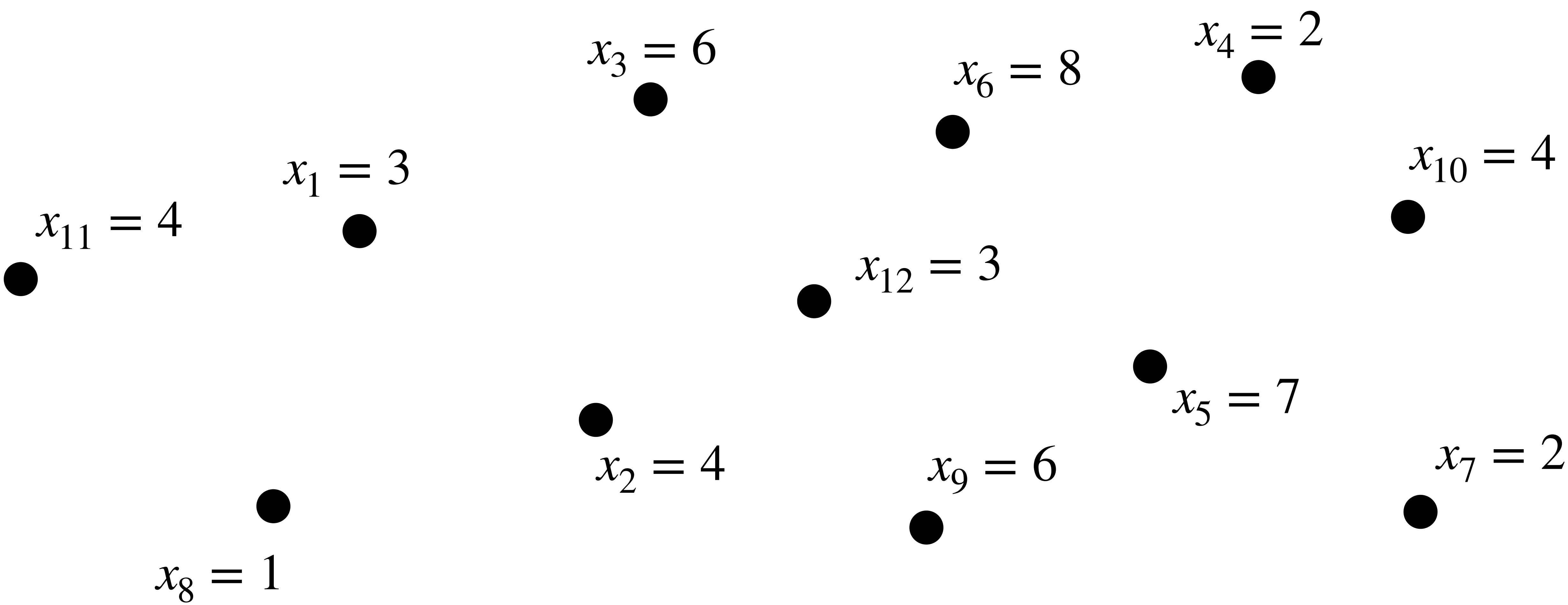
# Reducing Θ̃(N)-Collision Pairs to Element Distinctness



How to find **Θ̃(N) Collision Pairs** by using an algorithm for **Element Distinctness**



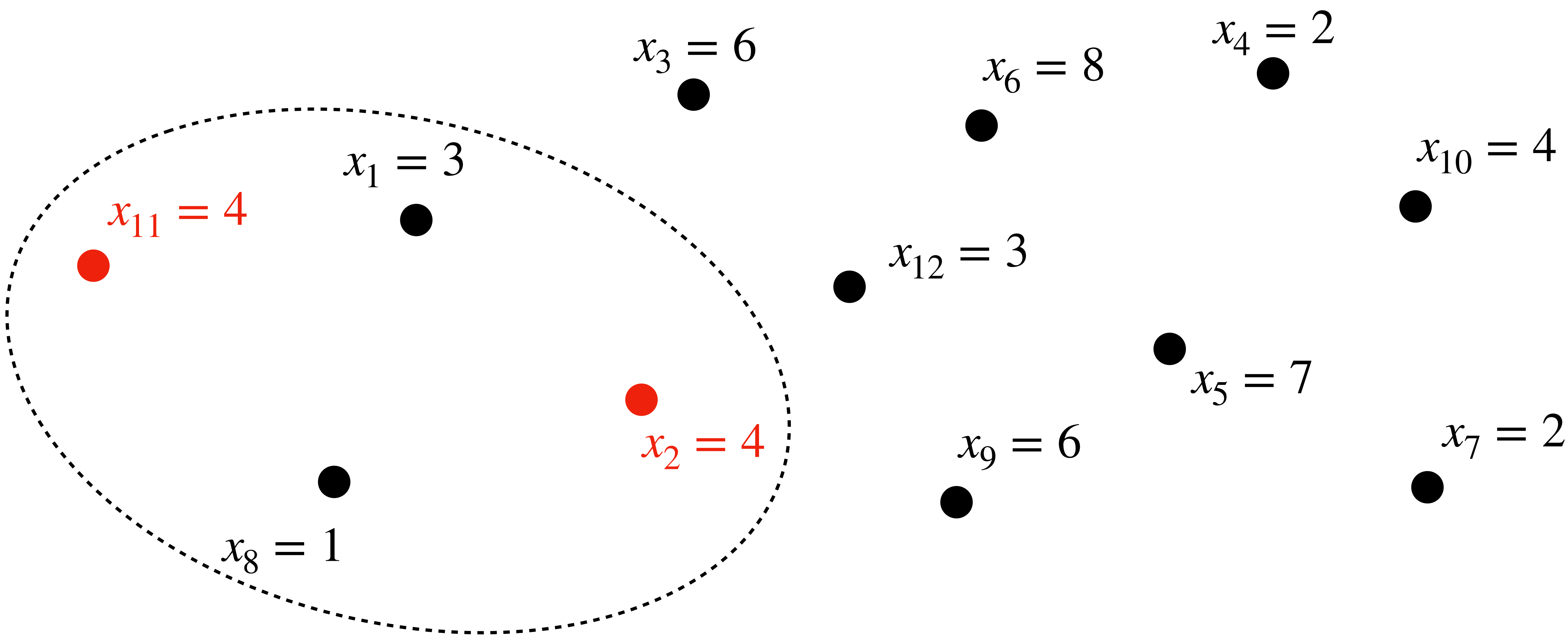
How to find **Θ̃(N) Collision Pairs** by using an algorithm for **Element Distinctness**




**Repeat O(N) times:** sample $\sqrt{N}$ elements and find a collision among them with ED.

# Reducing Θ̃(N)-Collision Pairs to Element Distinctness 

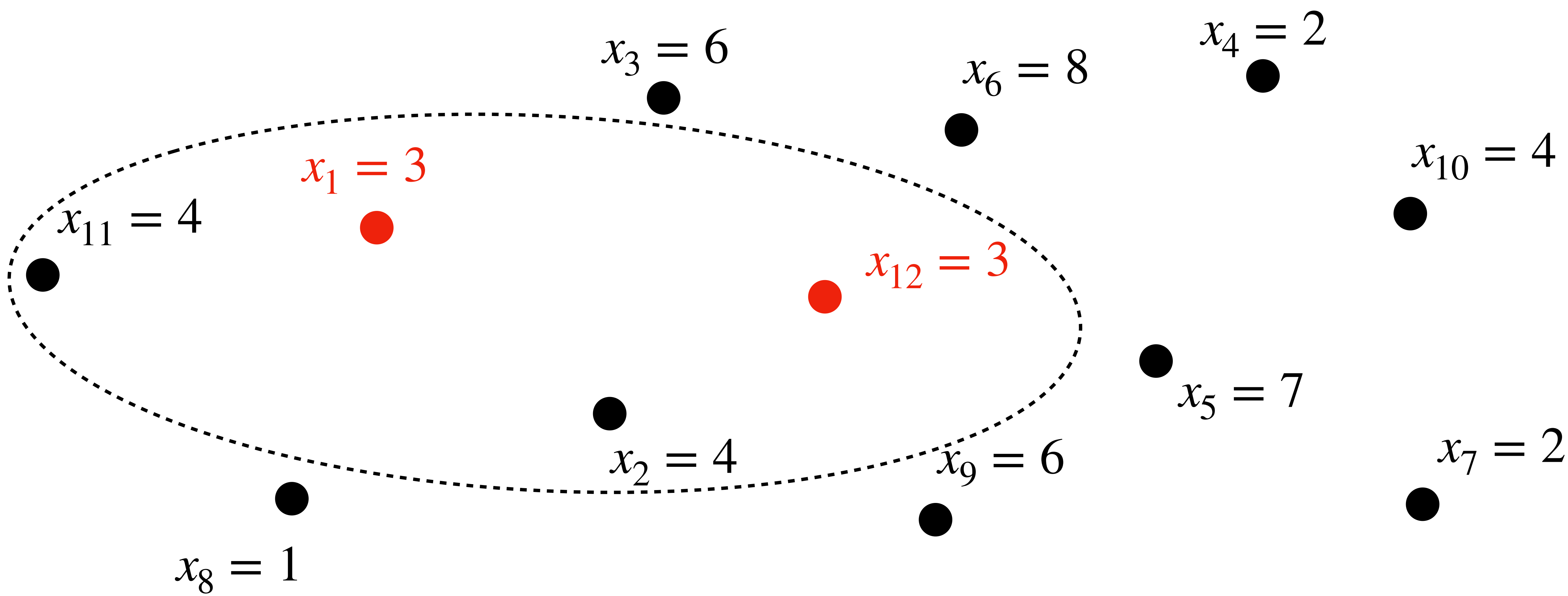
How to find **Θ̃(N) Collision Pairs** by using an algorithm for **Element Distinctness**




**Repeat O(N) times:** sample $\sqrt{N}$ elements and find a collision among them with ED.

# Reducing $\tilde{\Theta}$(N)-Collision Pairs to Element Distinctness 

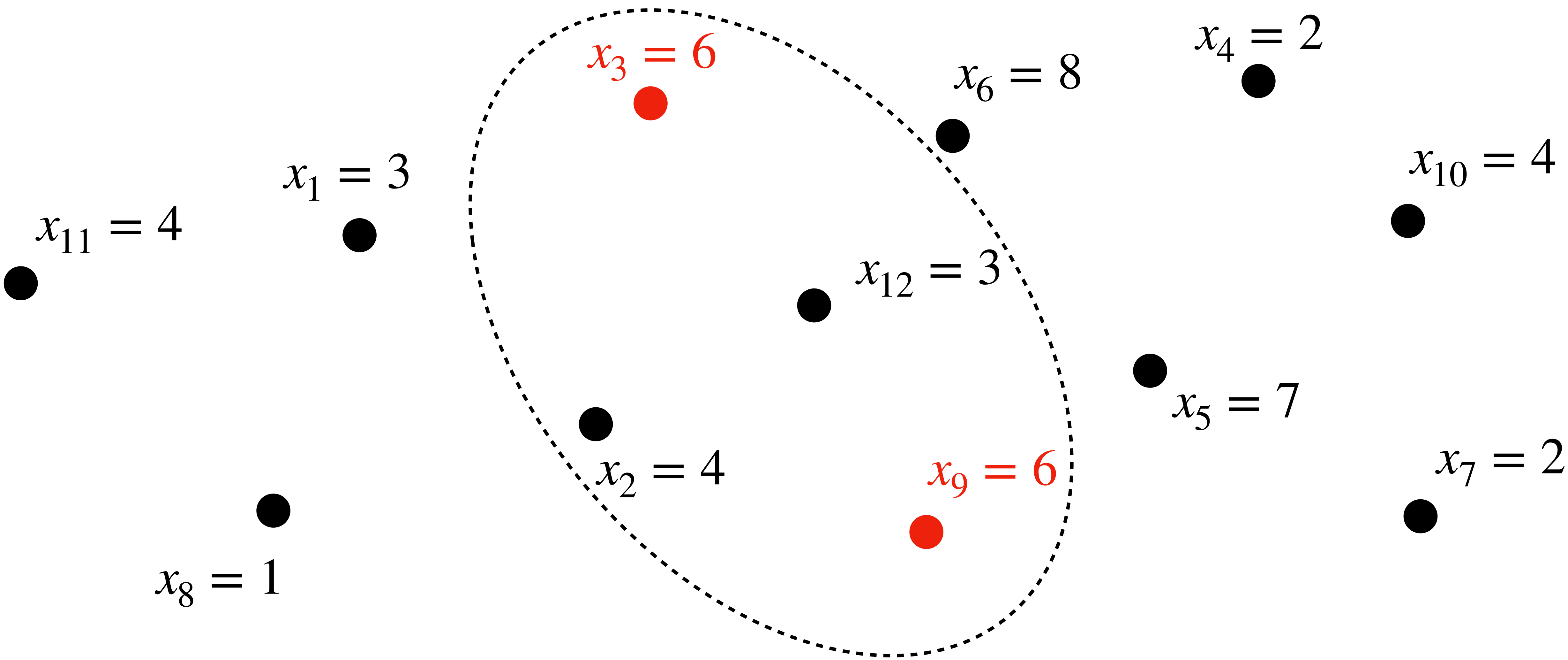
How to find **$\tilde{\Theta}$(N) Collision Pairs** by using an algorithm for **Element Distinctness**




**Repeat O(N) times:** sample $\sqrt{N}$ elements and find a collision among them with ED.

# Reducing Θ̃(N)-Collision Pairs to Element Distinctness

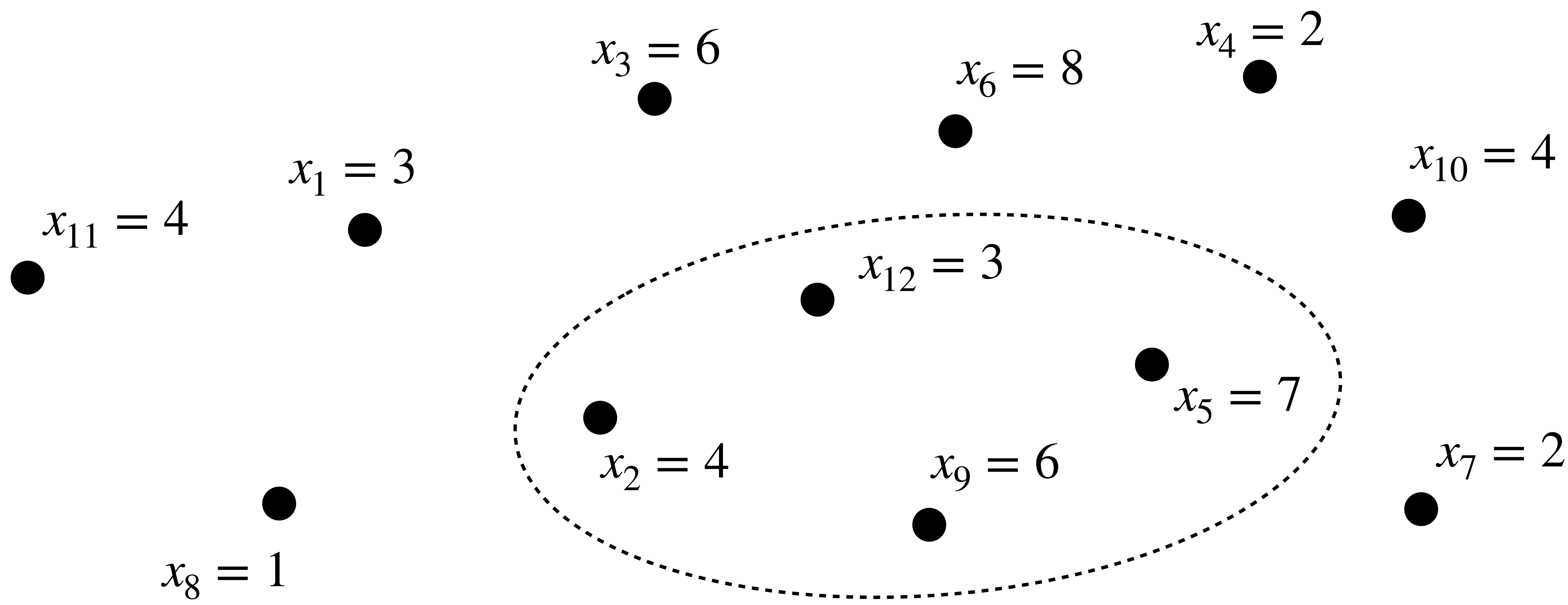
How to find **Θ̃(N) Collision Pairs** by using an algorithm for **Element Distinctness**




**Repeat O(N) times:** sample $\sqrt{N}$ elements and find a collision among them with ED.

# Reducing Θ̃(N)-Collision Pairs to Element Distinctness 

How to find **Θ̃(N) Collision Pairs** by using an algorithm for **Element Distinctness**

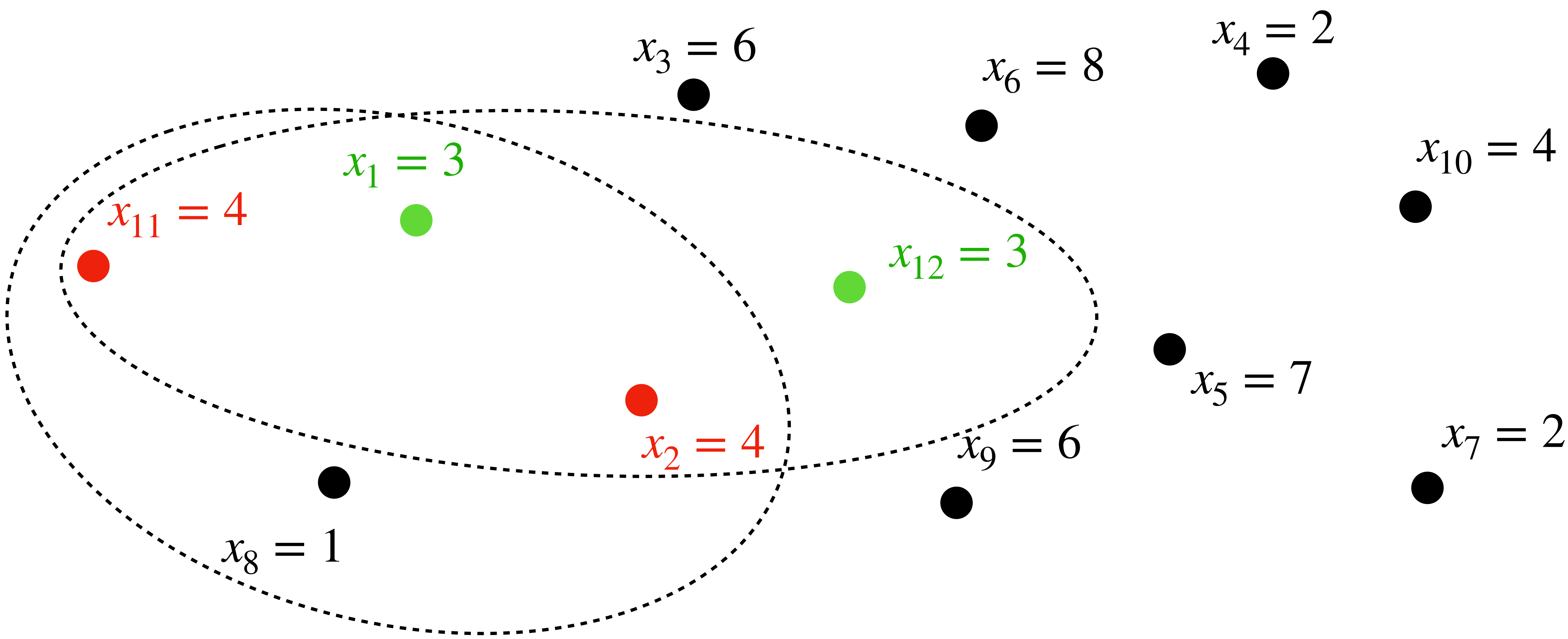



**Repeat O(N) times:** sample $\sqrt{N}$ elements and find a collision among them with ED.

→ Sometimes, there is no collision to find.

# Reducing Θ̃(N)-Collision Pairs to Element Distinctness

How to find **Θ̃(N) Collision Pairs** by using an algorithm for **Element Distinctness**



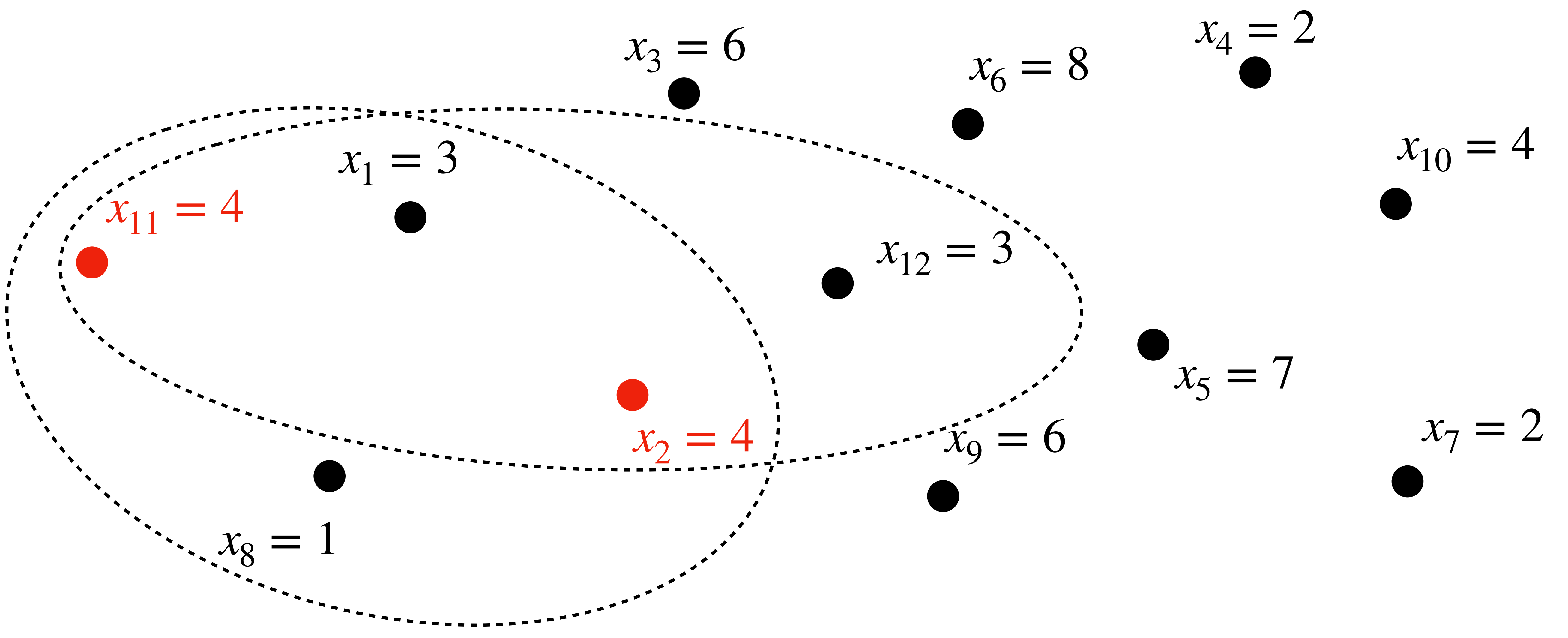

**Repeat O(N) times:** sample $\sqrt{N}$ elements and find a collision among them with ED.

→ Sometimes, there is no collision to find.

→ We cannot control which collision ED is going to output.

# Reducing $\tilde{\Theta}(N)$-Collision Pairs to Element Distinctness

How to find **$\tilde{\Theta}(N)$ Collision Pairs** by using an algorithm for **Element Distinctness**




**Repeat O(N) times:** sample $\sqrt{N}$ elements and find a collision among them with ED.

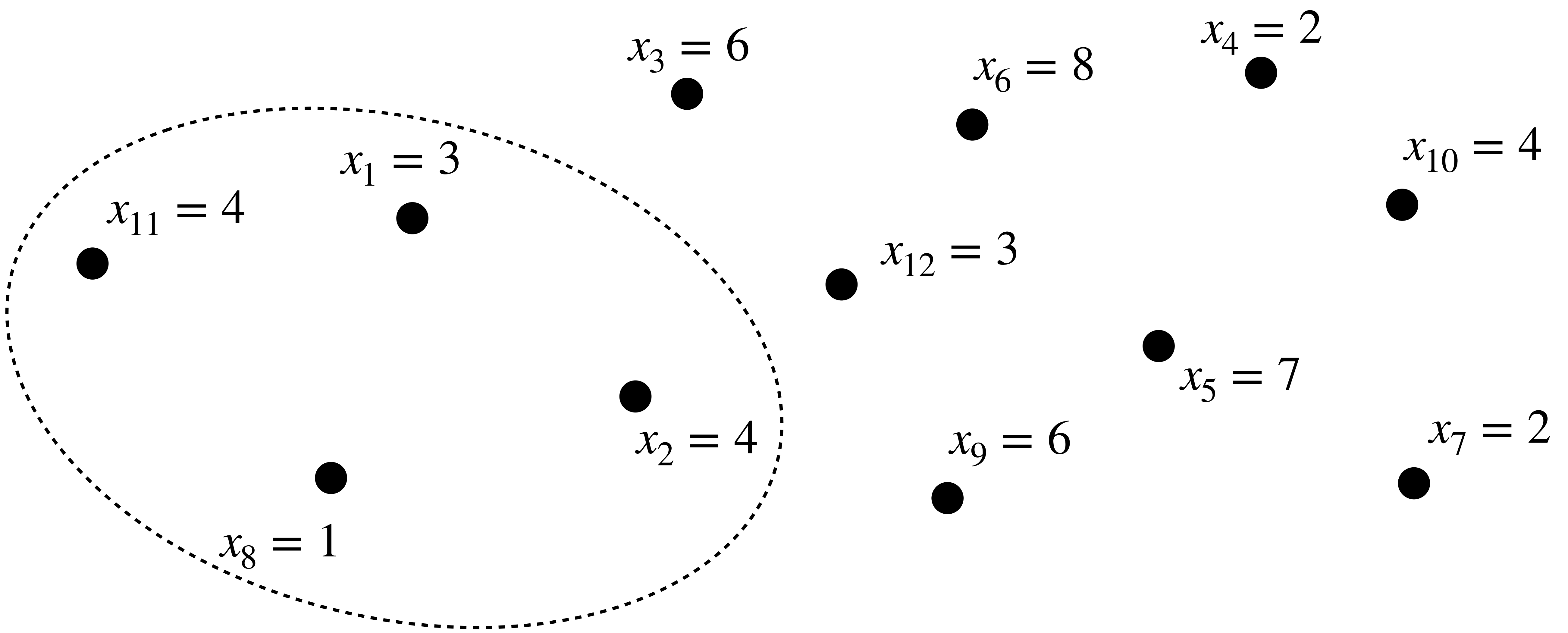
→ Sometimes, there is no collision to find.

→ We cannot control which collision ED is going to output.

→ We can output the same collision many times (but it only counts as one collision).

# Reducing Θ̃(N)-Collision Pairs to Element Distinctness

How to find **Θ̃(N) Collision Pairs** by using an algorithm for **Element Distinctness**




**Repeat O(N) times:** sample $\sqrt{N}$ elements and find a collision among them with ED.

→ Sometimes, there is no collision to find.

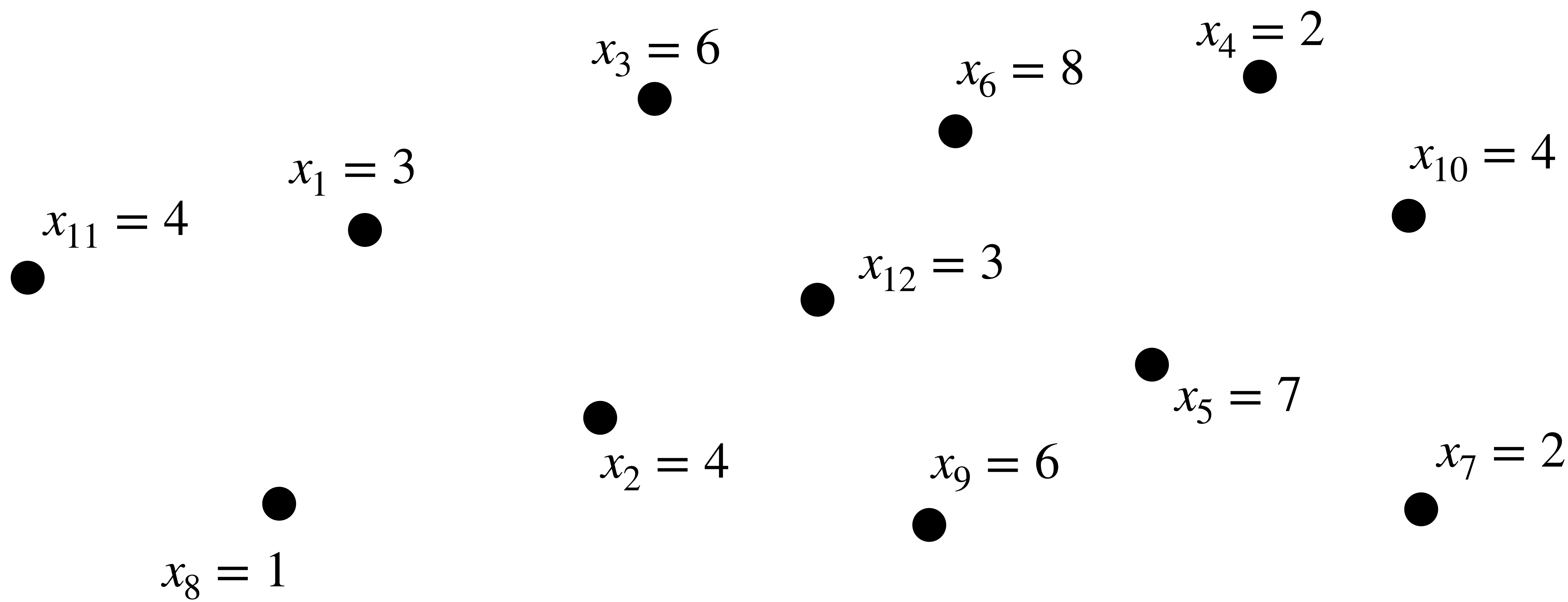
→ We cannot control which collision ED is going to output.

→ We can output the same collision many times (but it only counts as one collision).

→ We need to store the $\sqrt{N}$ sampled indices ⇒ **4-wise independent sampling**

# Reducing Θ̃(N)-Collision Pairs to Element Distinctness 

How to find **Θ̃(N) Collision Pairs** by using an algorithm for **Element Distinctness**

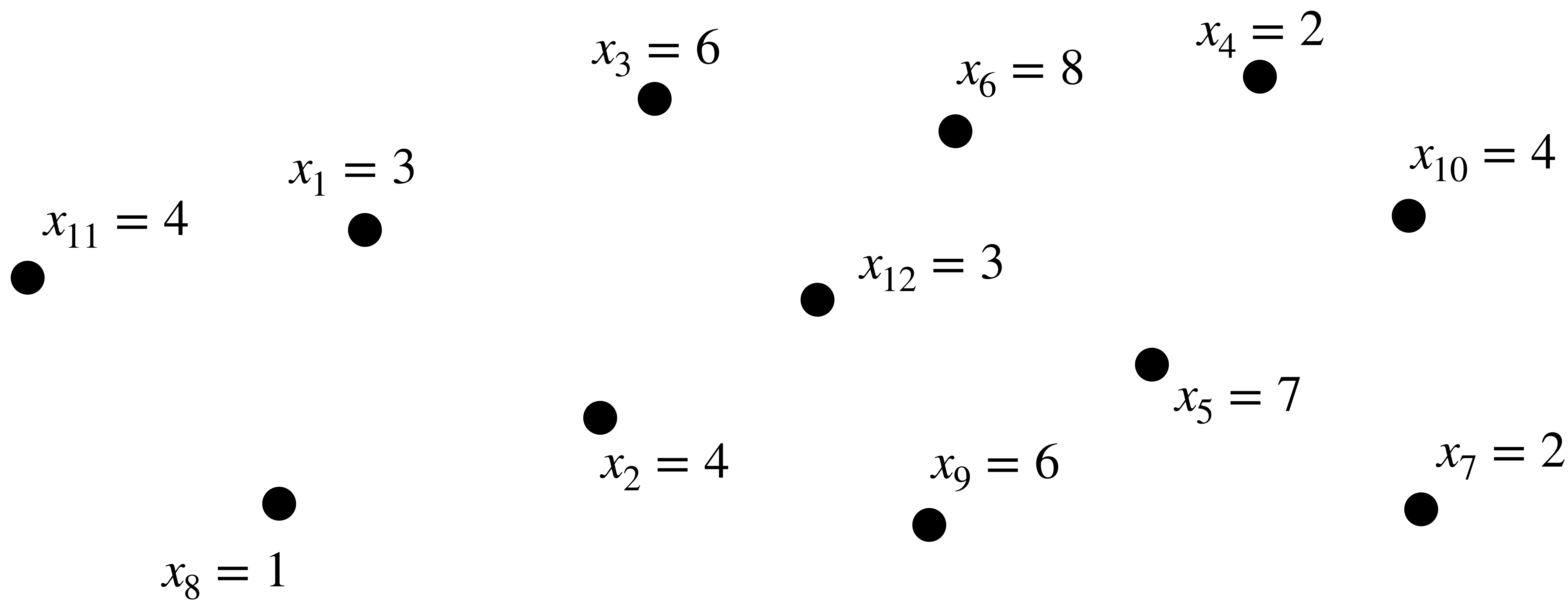



**(T,S)**-Algorithm for **Element Distinctness** on inputs of size $\sqrt{N}$ $\Longrightarrow$ **(NT,S)**-Algorithm for finding **Θ̃(N) Collision Pairs** on inputs of size $N$

# Reducing Θ̃(N)-Collision Pairs to Element Distinctness



How to find **Θ̃(N) Collision Pairs** by using an algorithm for **Element Distinctness**

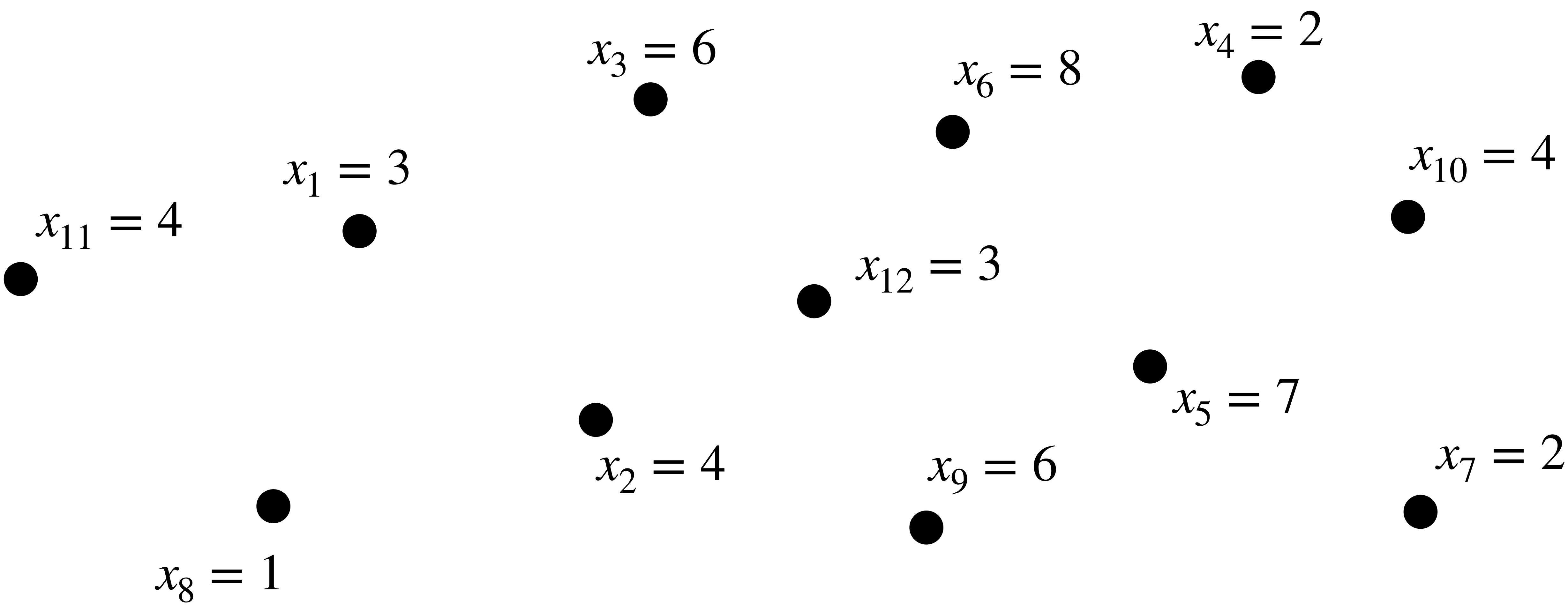



**(T,S)**-Algorithm for **Element Distinctness** on inputs of size $\sqrt{N}$ $\Longrightarrow$ **(NT,S)**-Algorithm for finding **Θ̃(N) Collision Pairs** on inputs of size $N$

$$(NT)^2 S \geq \tilde{\Omega}\left(N^3\right)$$

# Reducing Θ̃(N)-Collision Pairs to Element Distinctness

How to find **Θ̃(N) Collision Pairs** by using an algorithm for **Element Distinctness**




**(T,S)**-Algorithm for **Element Distinctness** on inputs of size $\sqrt{N}$ $\Longrightarrow$ **(NT,S)**-Algorithm for finding **Θ̃(N) Collision Pairs** on inputs of size $N$

$T^2 S \geq \tilde{\Omega}\left(\sqrt{N}^2\right)$ $\Longleftarrow$ $(NT)^2 S \geq \tilde{\Omega}\left(N^3\right)$