V3c
Generic attacks

HASH FUNCTIONS

CRYPTO 101: Building Blocks

cryptography101.ca

Generic attacks

A generic attack on hash functions $H : \{0,1\}^* \longrightarrow \{0,1\}^n$ does not exploit any properties that the specific hash function might have.

- In the analysis of a generic attack, we view H as a random function in the sense that for each $x \in \{0,1\}^*$, the hash value $y = H(x)$ was defined by selecting $y \in_R \{0,1\}^n$.
- From a security point of view, a random function is an ideal hash function. However, random functions are not suitable for practical applications because they cannot be compactly described.

Generic attack for finding preimages

- **Attack**: Given $y \in_R \{0,1\}^n$, repeatedly select arbitrary $x \in \{0,1\}^*$ until $H(x) = y$.
- **Analysis**: The expected number of hash operations is 2^n.

- This generic attack is infeasible if $n \geq 128$.
- Note: It has been proven that this generic attack for finding preimages is optimal, i.e., no faster generic attack exists. Of course, for a specific hash function, there might exist a faster preimage finding algorithm.

Generic attack for finding collisions

- **Attack**: Select arbitrary $x \in \{0,1\}^*$ and store $(H(x), x)$ in a table sorted by first entry. Repeat until a collision is found.
- **Analysis**: By the birthday paradox, the expected number of hash operations is $\sqrt{\pi 2^n/2} \approx \sqrt{2^n}$.

- This generic attack is infeasible if $n \geq 256$.
- Note: It has been proven that this generic attack for finding collisions is optimal, i.e., no faster generic attack exists.
- Expected space required: $\sqrt{\pi 2^n/2} \approx \sqrt{2^n}$.
- **Example**: If $n = 128$, the expected running time is 2^{64} (feasible), whereas the expected space required is 5×10^8 Tbytes (infeasible).

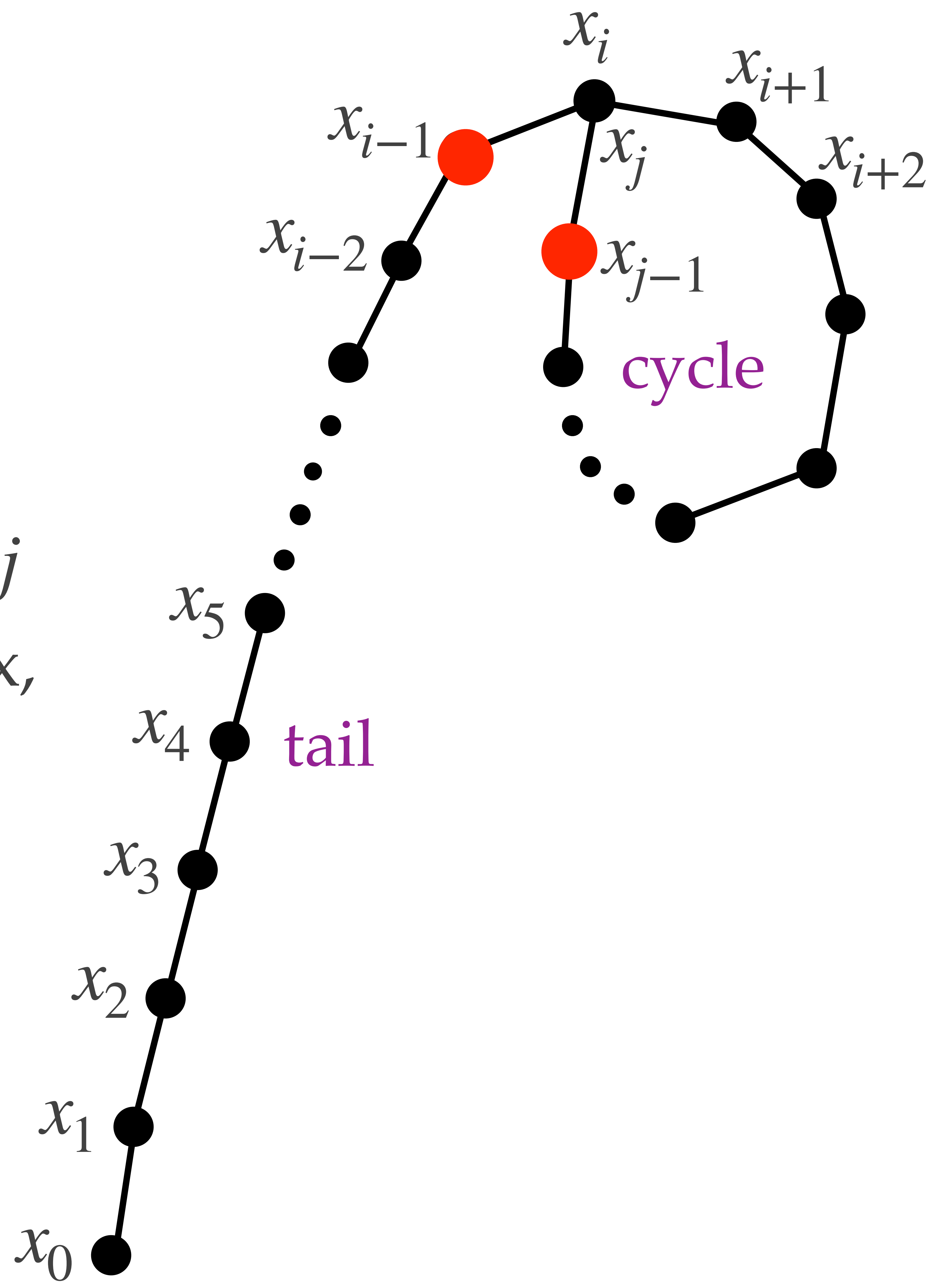
VW parallel collision search

- VW: van Oorschot & Wiener (1993)
- Expected number of hash operations: $\approx \sqrt{2^n}$.
- Expected space required: negligible.
- Easy to parallelize — m-fold speedup with m processors.
- The VW collision-finding algorithm can easily be modified to find "meaningful" collisions. (See Optional Readings at cryptography101.ca.)
- **Conclusion**: If collision resistance is desired, then use an n-bit hash function with $n \geq 256$.

Parallel collision search (VW method)

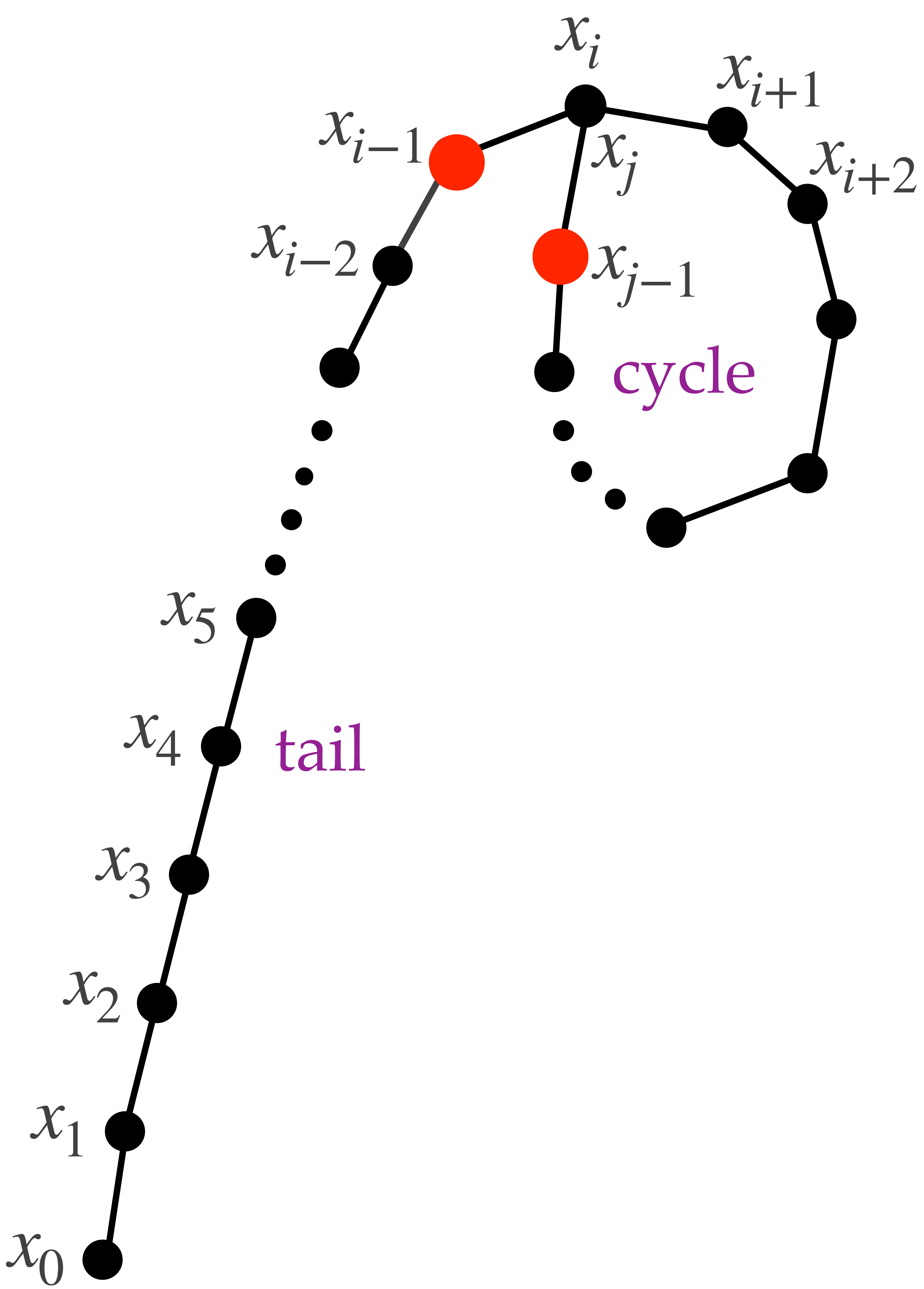
- **Problem**: Find a collision for $H : \{0,1\}^* \longrightarrow \{0,1\}^n$.
- **Assumption**: H is a random function.
- **Notation**: Let $N = 2^n$.
  Define a sequence $\{x_i\}_{i \ge 0}$ by $x_0 \in_R \{0,1\}^n$, $x_i = H(x_{i-1})$ for $i \ge 1$.

  Let j be the smallest index for which $x_j = x_i$ for some $i < j$; such a j must exist. Then $x_{j+\ell} = x_{i+\ell}$ for all $\ell \ge 1$. By the birthday paradox, $E[j] \approx \sqrt{\pi N/2} \approx \sqrt{N}$. In fact, $E[i] \approx \frac{1}{2}\sqrt{N}$ and $E[j-i] \approx \frac{1}{2}\sqrt{N}$.
- Now, $i \ne 0$ with overwhelming probability, in which event (x_{i-1}, x_{j-1}) is a collision for H.
- **Question**: How to find (x_{i-1}, x_{j-1}) without using much storage?

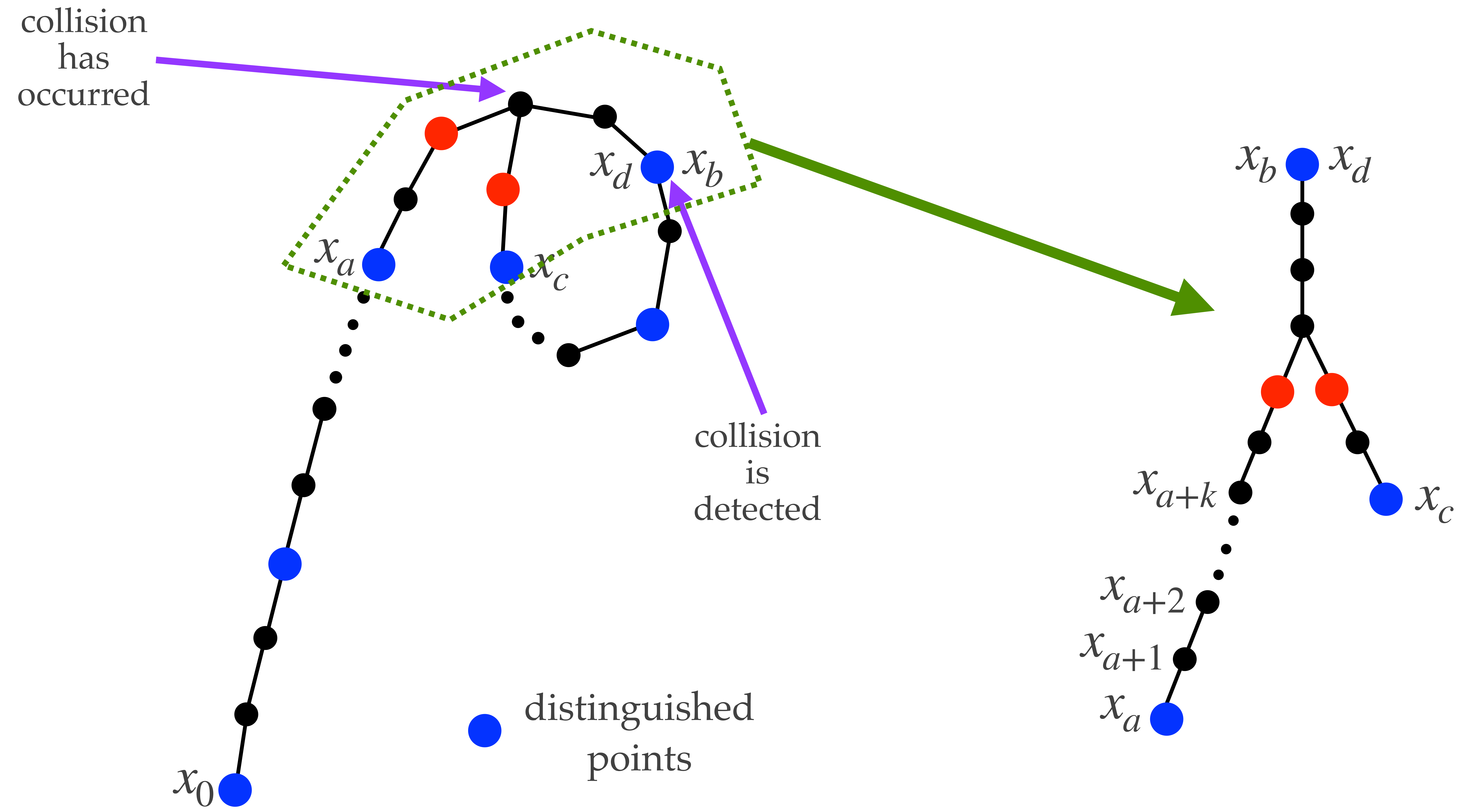


Distinguished points

- **Answer**: Only store distinguished points.
- Distinguished points: Select an easily-testable distinguishing property for elements of $\{0,1\}^n$, e.g. leading 32 bits are all 0.
  Let θ be the proportion of elements of $\{0,1\}^n$ that are distinguished.
- VW method: *Compute the sequence $x_0, x_1, x_2, x_3, \ldots$ and only store the points that are distinguished.*



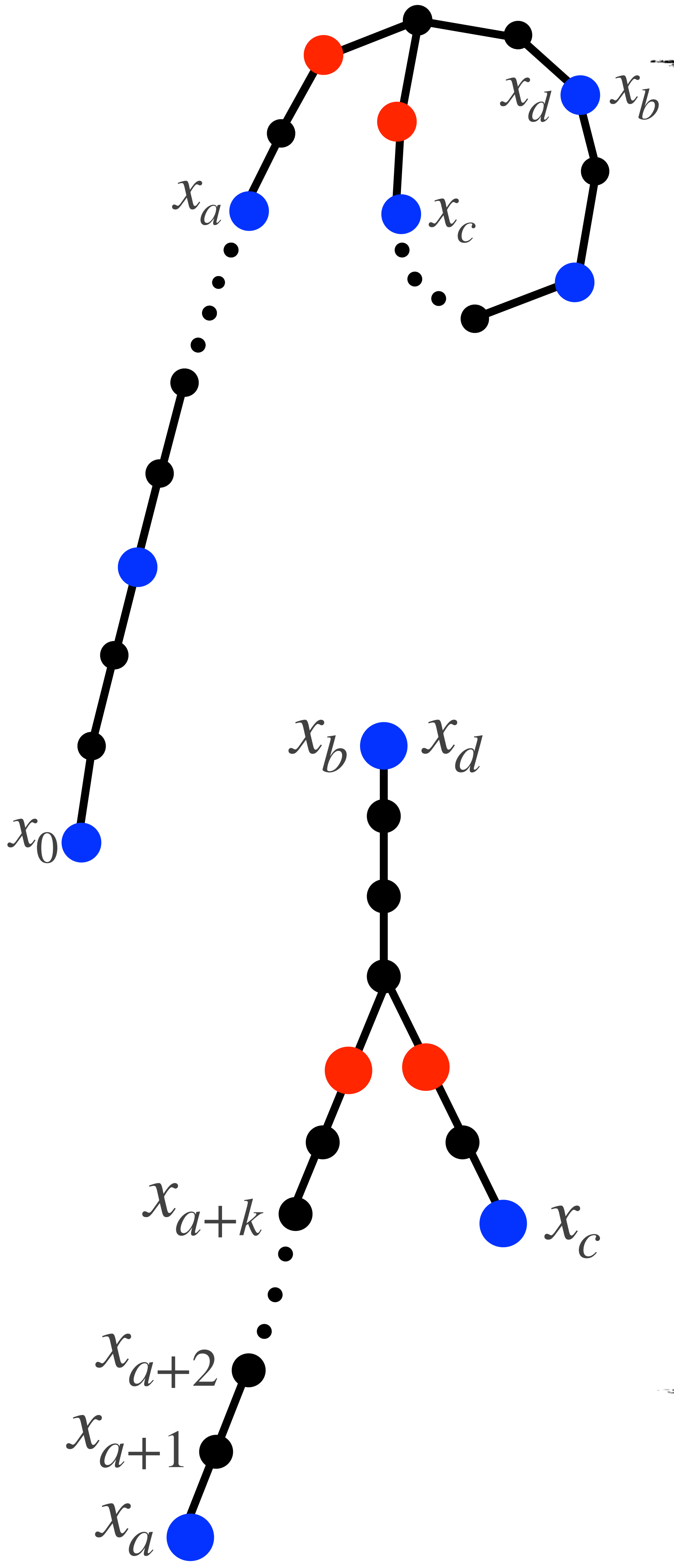
VW collision finding



VW collision finding

Stage 1: Detecting a collision

1. Select $x_0 \in_R \{0,1\}^n$.
2. Store $(x_0, 0, -)$ in a sorted table.
3. LP $\leftarrow x_0$. (LP= last point stored)
4. For $d = 1,2,3,\ldots$ do:
    a. Compute $x_d = H(x_{d-1})$.
    b. If x_d is distinguished then
        i. If x_d is already in the table, say $x_d = x_b$ where $b < d$, then go to Stage 2.
        ii. Store (x_d, d, LP) in the table.
        iii. LP $\leftarrow x_d$.



Stage 2: Finding a collision

1. Set $\ell_1 \leftarrow b - a, \quad \ell_2 \leftarrow d - c$.
2. Suppose $\ell_1 \geq \ell_2$, and set $k \leftarrow \ell_1 - \ell_2$.
3. Compute $x_{a+1}, x_{a+2}, \ldots, x_{a+k}$.
4. For $m = 1,2,3,\ldots$ do:
    a) Compute (x_{a+k+m}, x_{c+m}).
5. Until $x_{a+k+m} = x_{c+m}$.
6. The collision is $(x_{a+k+m-1}, x_{c+m-1})$.

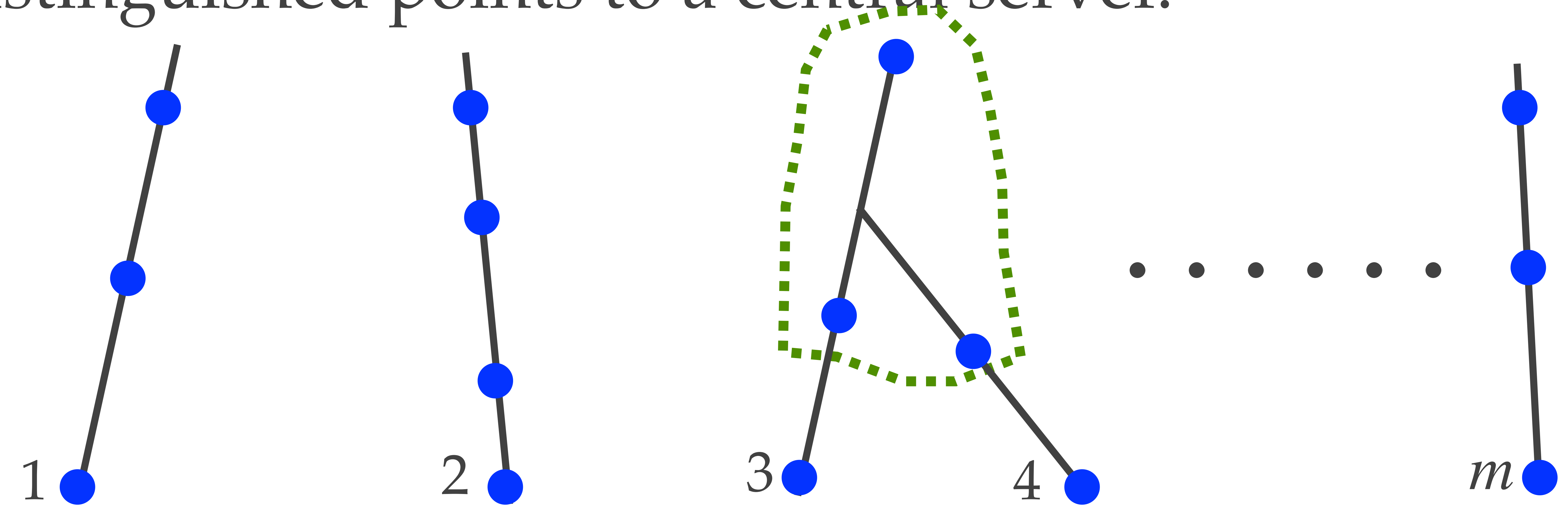
VW analysis

- Stage 1: Expected number of H-evaluations is:
$$\sqrt{\pi N/2} + \frac{1}{\theta} \quad \approx \quad \sqrt{N} + \frac{1}{\theta}.$$
- Stage 2: Expected number of H-evaluations is $\leq \frac{3}{\theta}$ (see optional readings).
- Overall expected running time: $\sqrt{N} + \frac{4}{\theta}$.
- Expected storage: $\approx 3n\theta\sqrt{N}$ bits (each table entry has bitlength $3n$).

- **Example**: Consider $n = 128$. Take $\theta = 1/2^{32}$. Then the expected run time of VW collision search is 2^{64} H-evaluations (feasible), and the expected storage is 192 Gbytes (negligible).

Parallelizing VW collision search

- Run independent copies of VW on each of m processors
- Report distinguished points to a central server.



Analysis

- Expected time $\approx \dfrac{1}{m}\sqrt{N} + \dfrac{4}{\theta}$.
- Expected storage $\approx 3n\theta\sqrt{N}$ bits.

Notes

1. Factor-m speedup.
2. No communications between processors.
3. Occasional communications with the central server.

