Cryptographic Hash Functions
Problem Set 3

Summer 2019

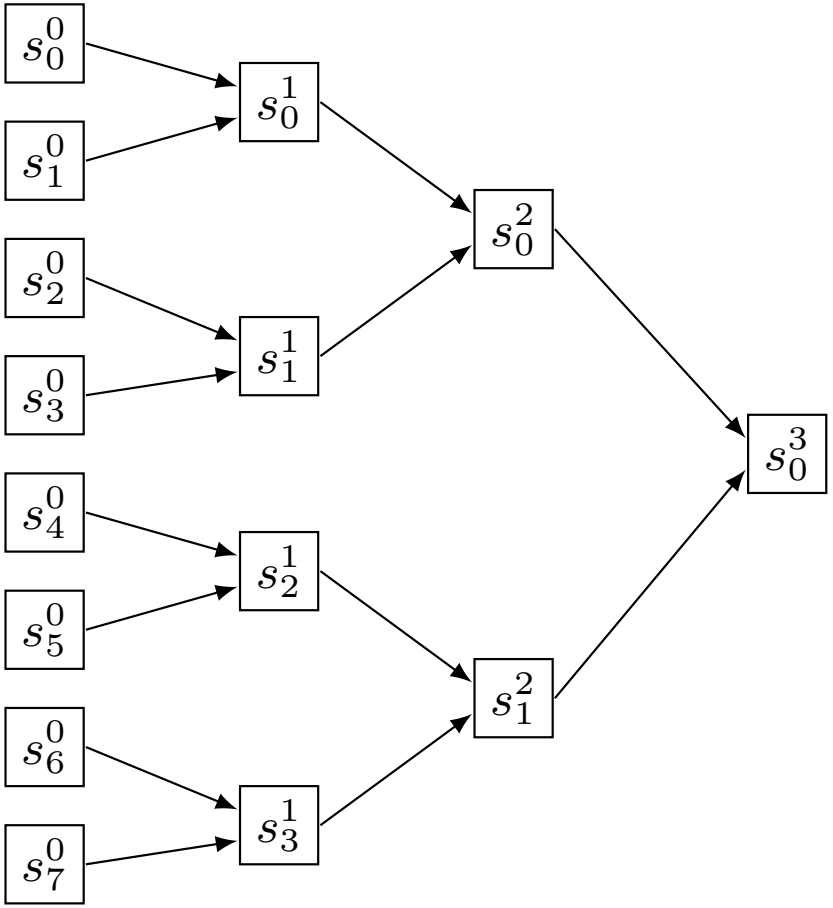
Chair of Media Security Prof. Stefan Lucks
Jannis Bossert, Nathalie Dittrich, Eik List <firstname>.<lastname>(at)uni-weimar.de

Bauhaus-Universität Weimar

May 16, 2019

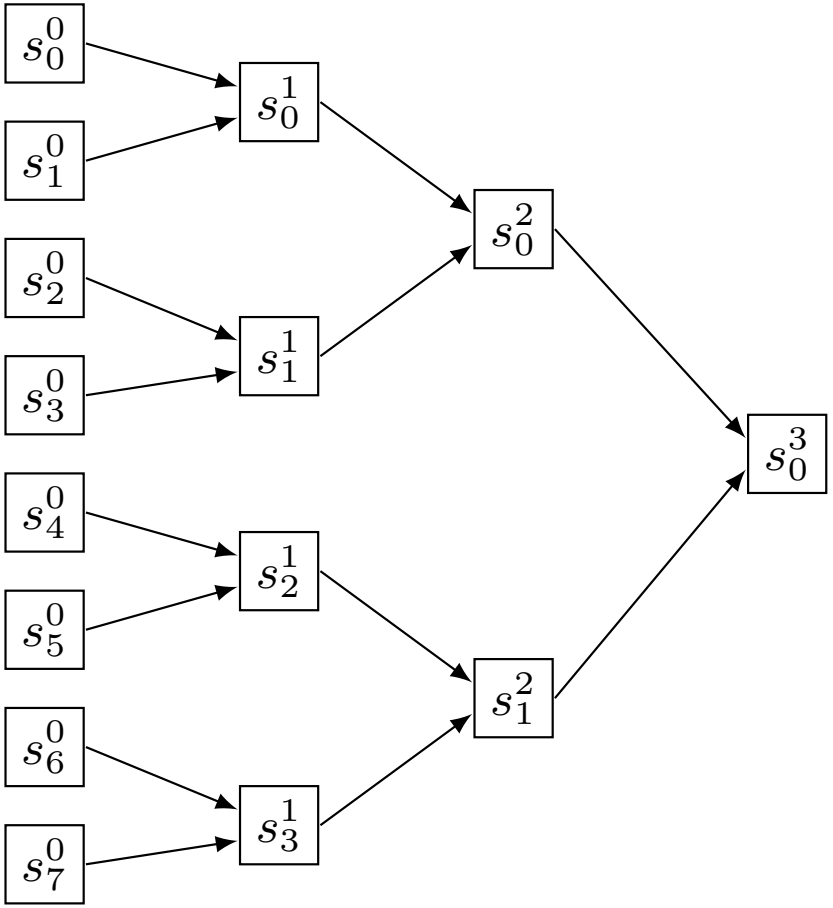
Task 1: The Nostradamus Attack

1. Precompute a diamond structure of k levels with 2^k start nodes
2. Layer i: Generate $O(2^{n/2} \cdot 2^{i/2})$ messages to find collisions; repeat for all $i \in [0, k]$ with $O(2^{2+(n+k)/2})$ messages
3. Publish hash H
4. Wait for result
5. Generate Preimage by traversing the tree, starting at correct startng node.



Task 1: The Nostradamus Attack

1. Precompute a diamond structure of k levels with 2^k start nodes
2. Layer i: Generate $O(2^{n/2} \cdot 2^{i/2})$ messages to find collisions; repeat for all $i \in [0, k]$ with $O(2^{2+(n+k)/2})$ messages
3. Publish hash H
4. Wait for result
5. Generate Preimage by traversing the tree, starting at correct startng node.



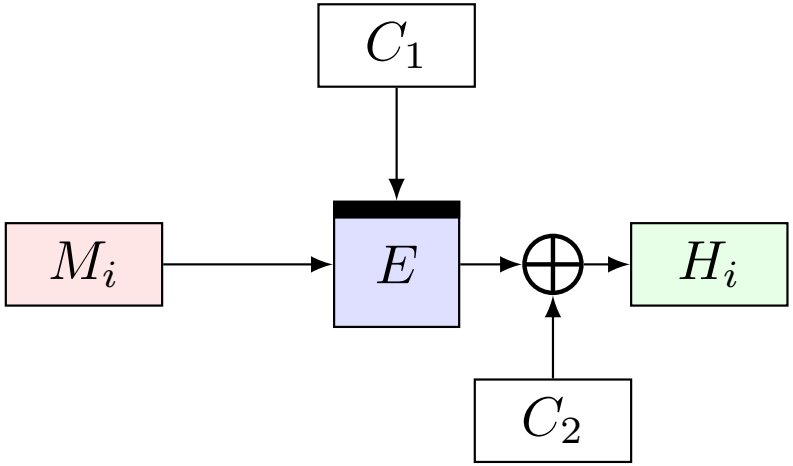
- Effort: $2^{(n+k)/2+2}$
- Minimum if all 32 competitors are known: $k = 5 \rightarrow 2^{36.5}$ queries

Task 2: Block-Cipher-Based Compression Functions

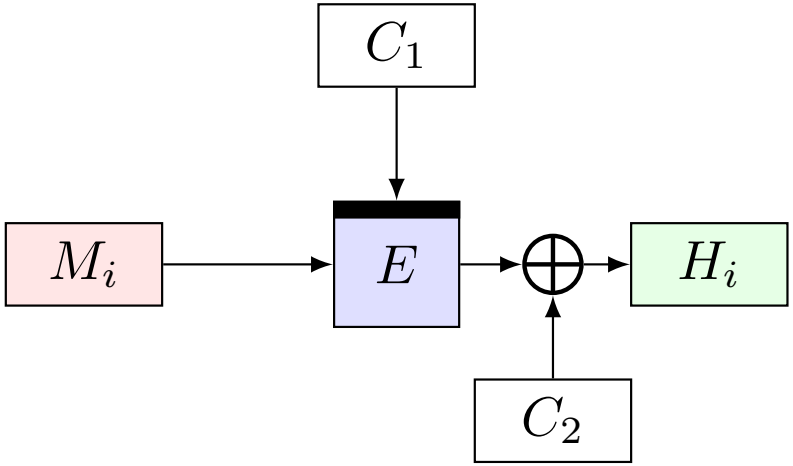
Attack Types from [PGV93]

- **Trivial**: H_i can be computed without knowing both H_{i-1} and M_i or at least $H_{i-1} \oplus M_i$.
- **Direct**: Given H_i and H_{i-1}, one can compute M_i directly.
- **Permutation**: There exists a function f with $H_i \leftarrow f(M_i) \oplus H_{i-1}$.
- **Forward**: Given H_{i-1}, H'_{i-1}, M_i, one can compute an M'_i with $\mathcal{CF}(H_{i-1}, M_i) = \mathcal{CF}(H'_{i-1}, M'_i)$.
- **Backward**: Given H_i, it is feasible to find M_i and H_{i-1} with $\mathcal{CF}(H_{i-1}, M_i) = H_i$.
- **Fixpoint**: It is feasible to find H_i and M_i with $\mathcal{CF}(H_i, M_i) = H_i$.

Task 2: Block-Cipher-Based Compression Functions (1/6)

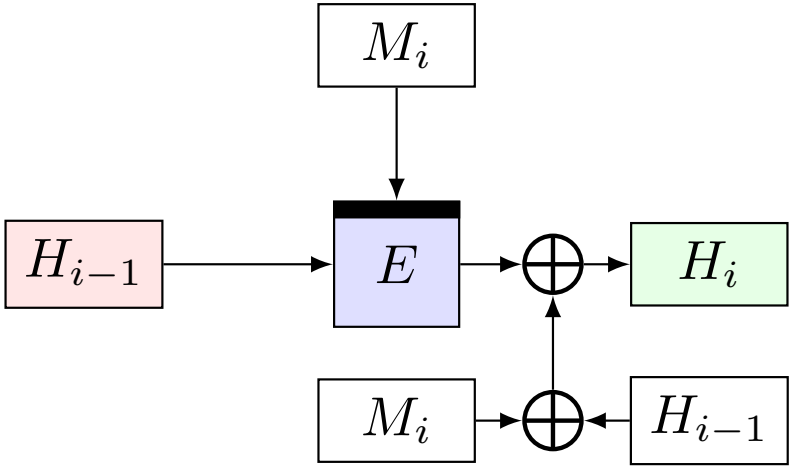


Task 2: Block-Cipher-Based Compression Functions (1/6)

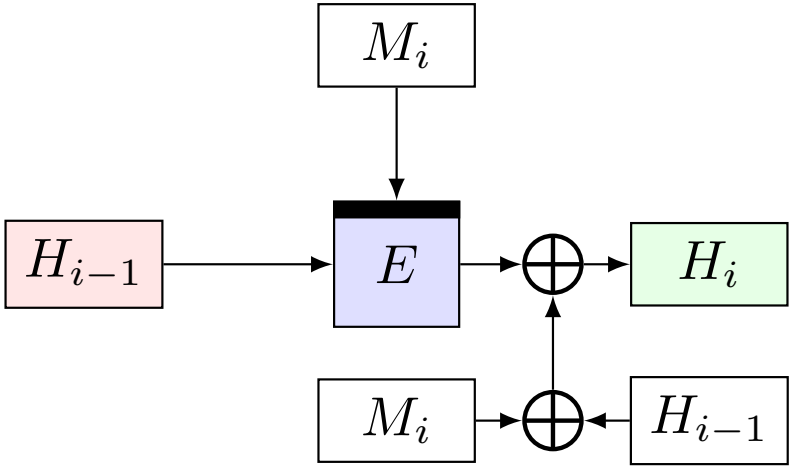


- **Trivial attack:** H_i can be computed without H_{i-1}

Task 2: Block-Cipher-Based Compression Functions (2/6)

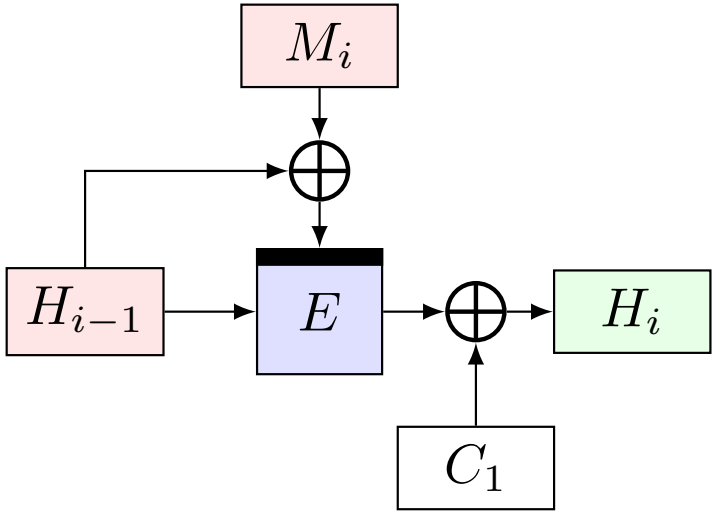


Task 2: Block-Cipher-Based Compression Functions (2/6)

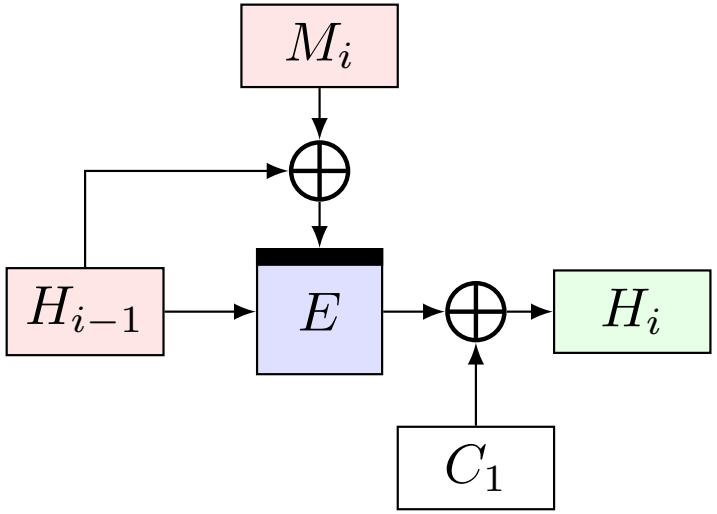


- **Fix-point attack:** Choose an arbitrary M_i and compute a fix point $H_{i-1} \leftarrow E^{-1}_{M_i}(M_i)$

Task 2: Block-Cipher-Based Compression Functions (3/6)

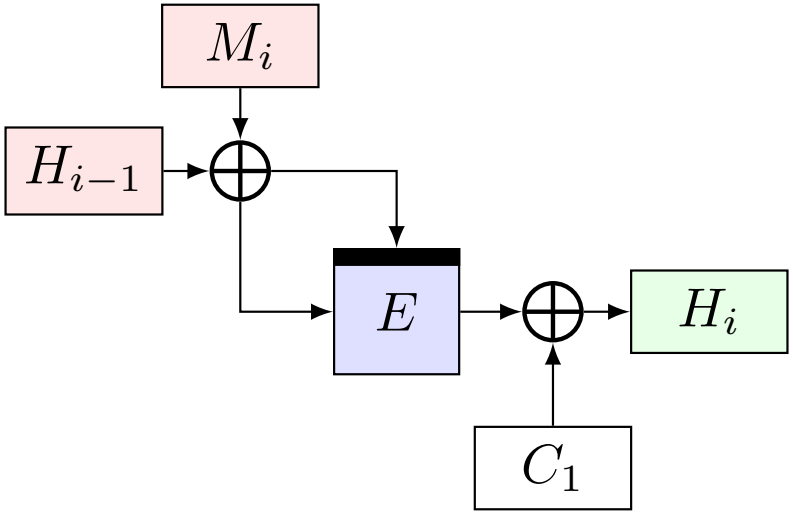


Task 2: Block-Cipher-Based Compression Functions (3/6)

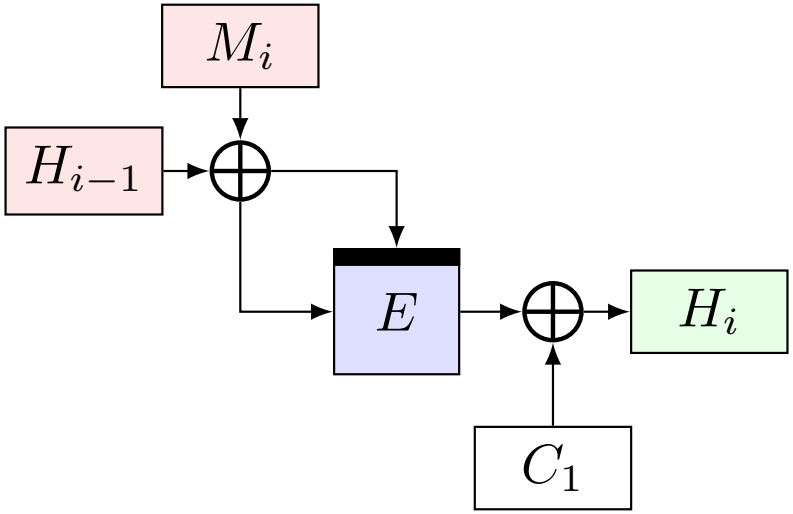


- **Backward:** Given H_i, choose any $K = M_i \oplus H_{i-1}$
- Compute $H_{i-1} \leftarrow E_K^{-1}(H_i \oplus \text{const}_1)$
- Derive $M_i \leftarrow K \oplus H_{i-1}$

Task 2: Block-Cipher-Based Compression Functions (4/6)

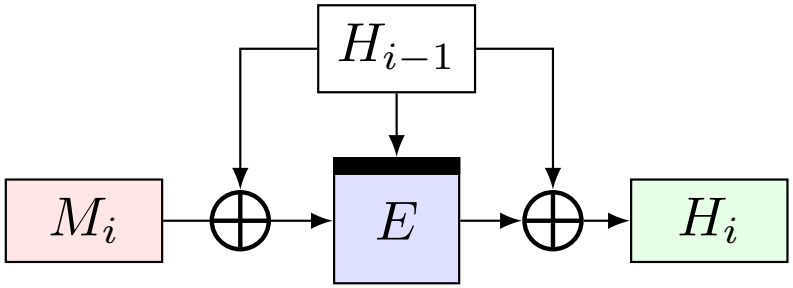


Task 2: Block-Cipher-Based Compression Functions (4/6)

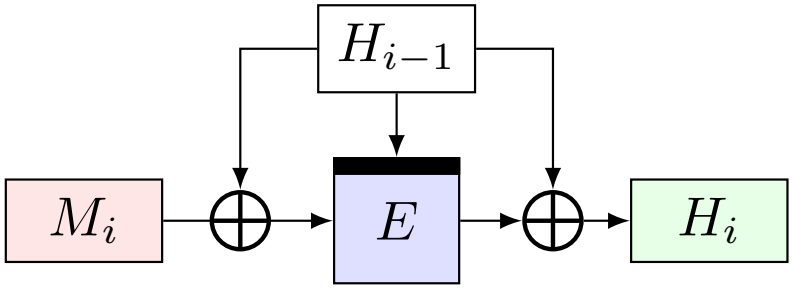


- **Forward:** Given M_i, H_{i-1}, and H'_{i-1}, derive $M'_1 \leftarrow M_i \oplus H_{i-1} \oplus H'_{i-1}$

Task 2: Block-Cipher-Based Compression Functions (5/6)

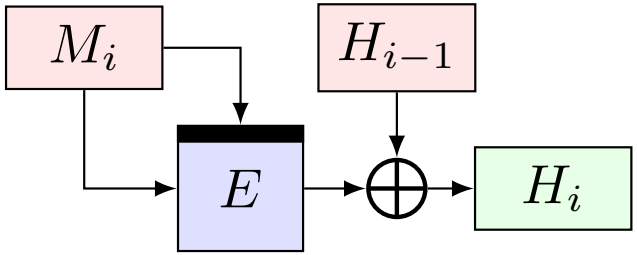


Task 2: Block-Cipher-Based Compression Functions (5/6)

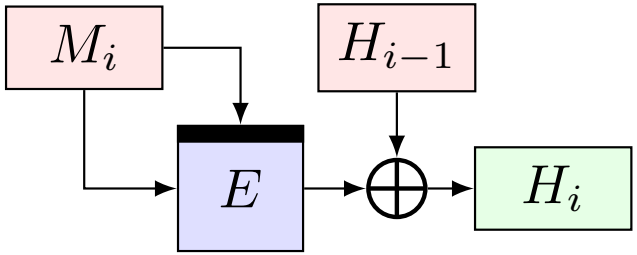


- **Direct:** Given H_{i-1} and H_i, we can derive $M_i \leftarrow E^{-1}_{H_{i-1}}(H_i \oplus H_{i-1}) \oplus H_{i-1}$ directly

Task 2: Block-Cipher-Based Compression Functions (6/6)



Task 2: Block-Cipher-Based Compression Functions (6/6)



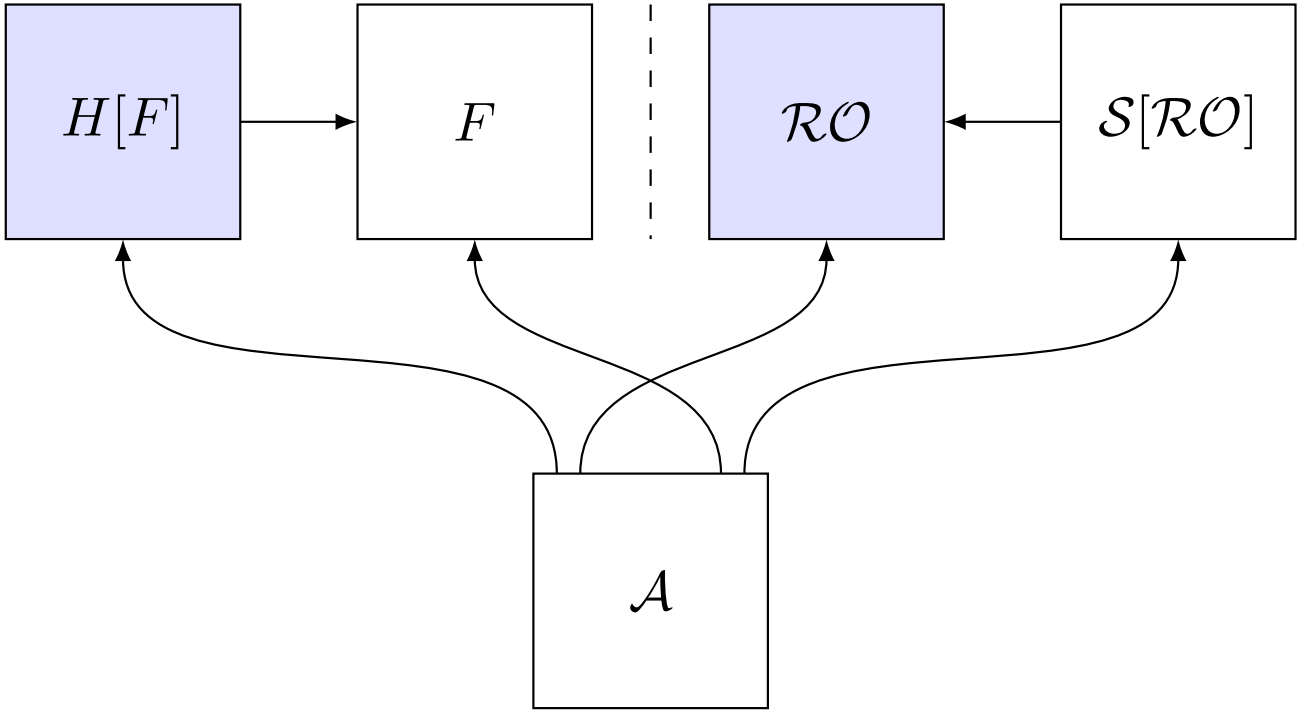
- **Permutation:** It holds that

$$\begin{aligned} H_1 &= E_{M_1}(M_1) \oplus H_0 \\ H_2 &= E_{M_2}(M_2) \oplus H_1 \\ &= E_{M_2}(M_2) \oplus E_{M_1}(M_1) \oplus H_0. \end{aligned}$$

For $(M_1', M_2') = (M_2, M_1)$, it holds that $H_2' = H_2$

Task 3: Indifferentiability

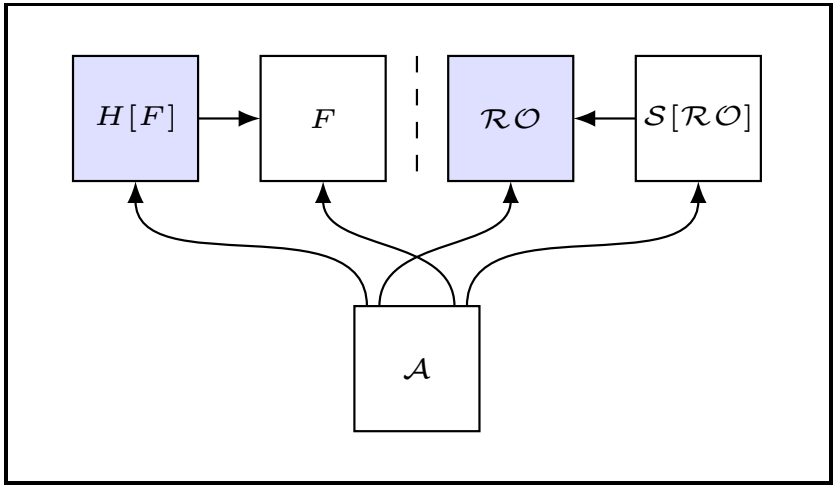
Model



- $H[F]$: Real construction; can query primitive F internally
- F: Real primitive
- $\mathcal{RO}$: Random oracle (ideal construction)
- $\mathcal{S}[\mathcal{RO}]$: Simulator, can query $\mathcal{RO}$ internally
- $\mathcal{A}$'s goal: Distinguish between both worlds

Task 3: Indifferentiability

Assume $H(x) := F(h(x))$; $h : \mathcal{X} \to \mathcal{Z}$ is a secure one-way hash function

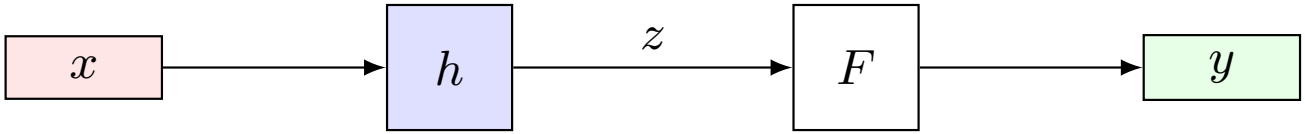


Attack:

1. $\mathcal{A}$ chooses x, asks for $y \leftarrow O_1(x)$
2. $\mathcal{A}$ computes $z \leftarrow h(x)$
3. $\mathcal{A}$ asks $y' \leftarrow O_2(z)$
4. If $y = y'$ then $\mathcal{A}$ outputs real

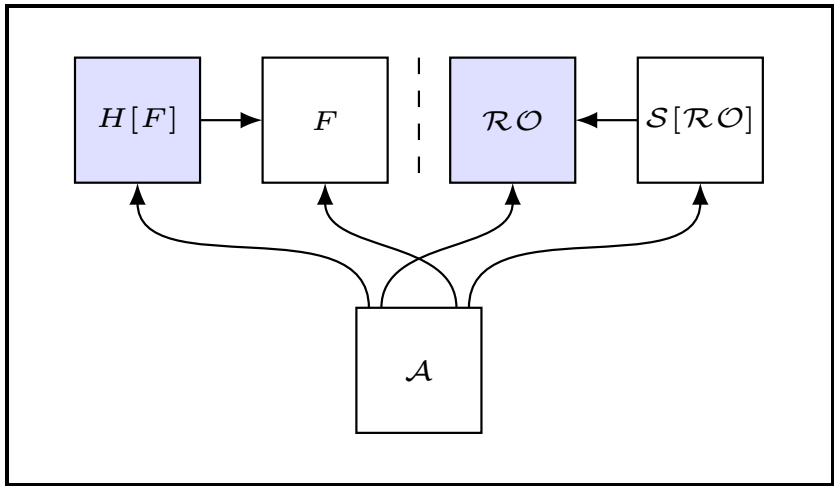
Reason:

- $\mathcal{S}$ cannot neither see y nor invert z from y
- $\implies$ $\mathcal{S}$ will most probably output $y' \neq y$



Task 3: Indifferentiability

Assume $H(x) := F(h(x))$;
$h : \mathcal{X} \to \mathcal{X}$ is public permutation

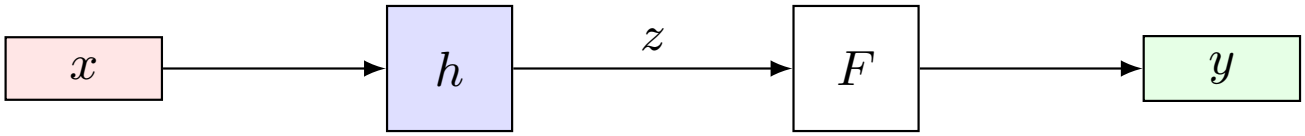


Attack:

1. $\mathcal{A}$ chooses x, asks for $y \leftarrow O_1(x)$
2. $\mathcal{A}$ computes $z \leftarrow h(x)$
3. $\mathcal{A}$ asks $y' \leftarrow O_2(z)$
4. $y = y'$ will hold in both worlds

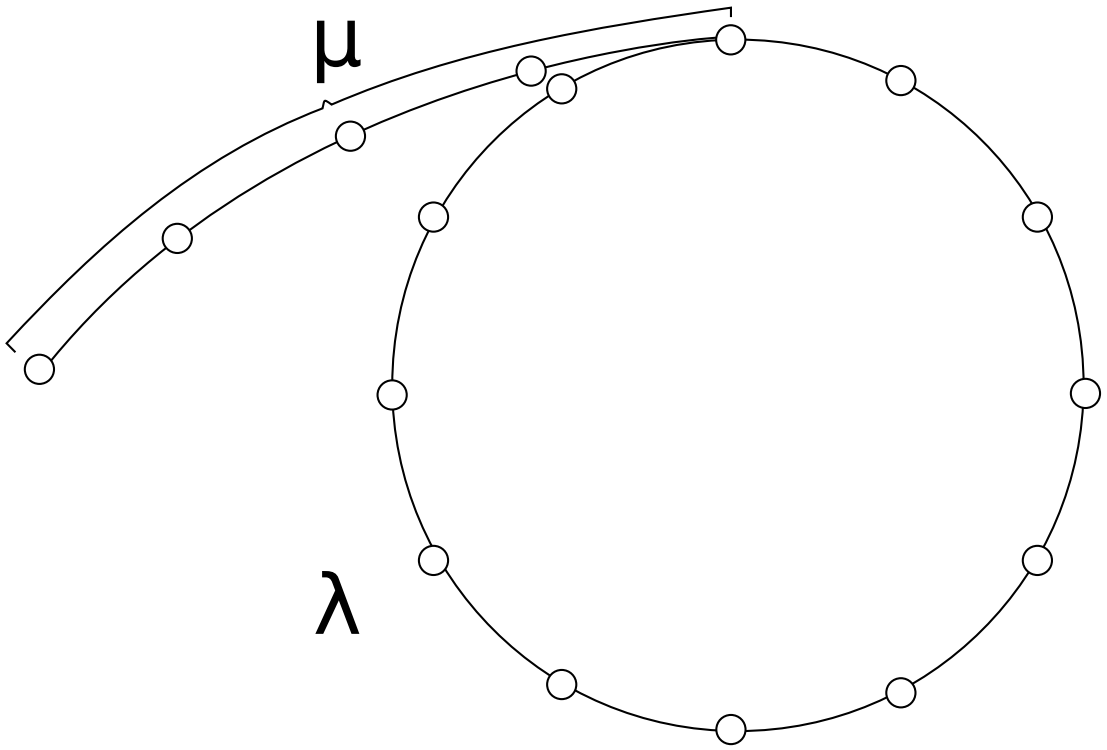
Reason:

- $\mathcal{S}$ **can** compute $x' \leftarrow h^{-1}(z)$
- $\implies$ $\mathcal{S}$ can always return $y \leftarrow \mathcal{RO}(x')$



Task 4: (Partial) Memoryless Collision Search

Common Model



Task 4: Collision Search

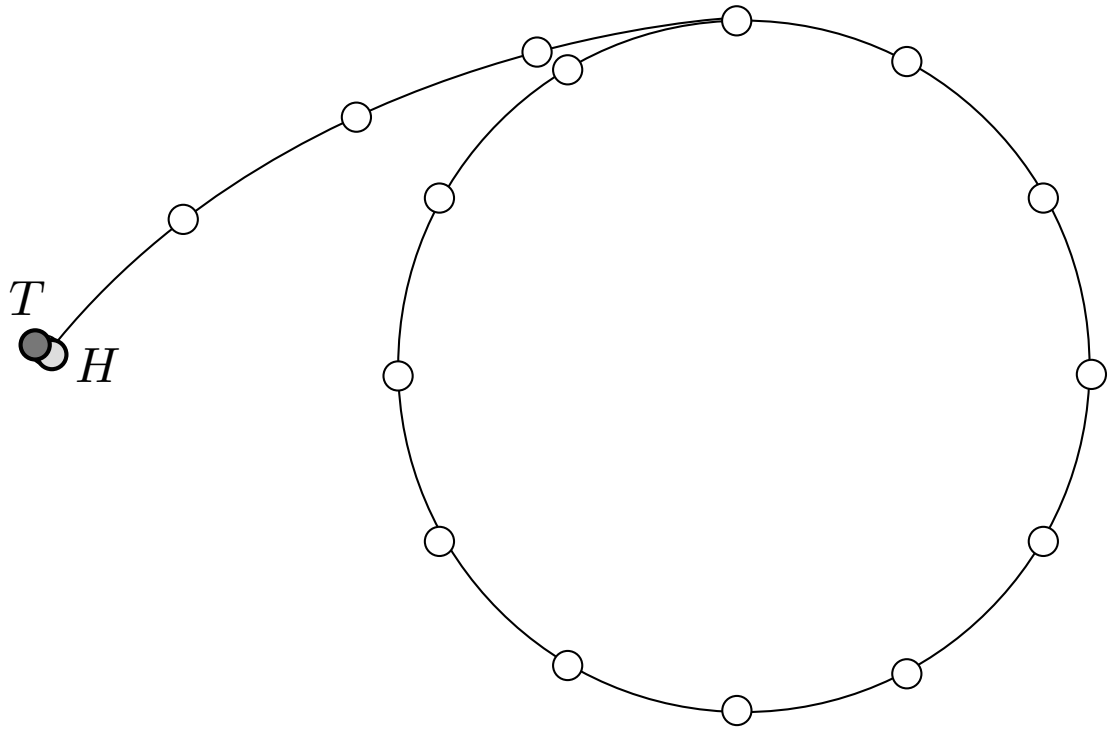
Floyd

```
function FLOYD(x_0)
    T ← F(x_0)
    H ← F(F(x_0))
    while T ≠ H do                    ▷ Loop #1: Find collision
        T ← F(T)                      ▷ Move tortoise 1 step
        H ← F(F(H))                   ▷ Move hare 2 steps
    μ ← 0
    T ← x_0                           ▷ Reset tortoise
    while T ≠ H do                    ▷ Loop #2: Find start
        T ← F(T)                      ▷ Move tortoise to collision
        H ← F(H)                      ▷ Move hare to collision
        μ ← μ + 1
    λ ← 1
    H ← F(T)
    while T ≠ H do                    ▷ Loop #3: Find cycle length
        H ← F(H)                      ▷ Let hare cycle once
        λ ← λ + 1
    return (λ, μ)
```

Task 4: Collision Search

Floyd

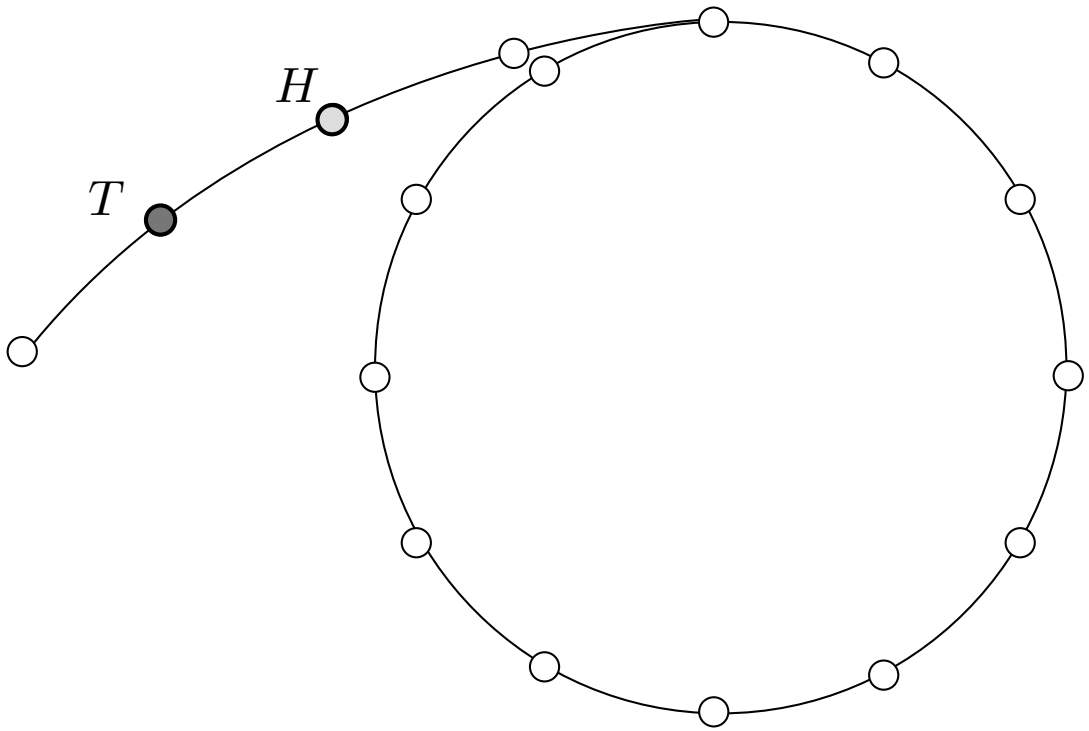
- Start at $T \leftarrow H \leftarrow x_0$



Task 4: Collision Search

Floyd

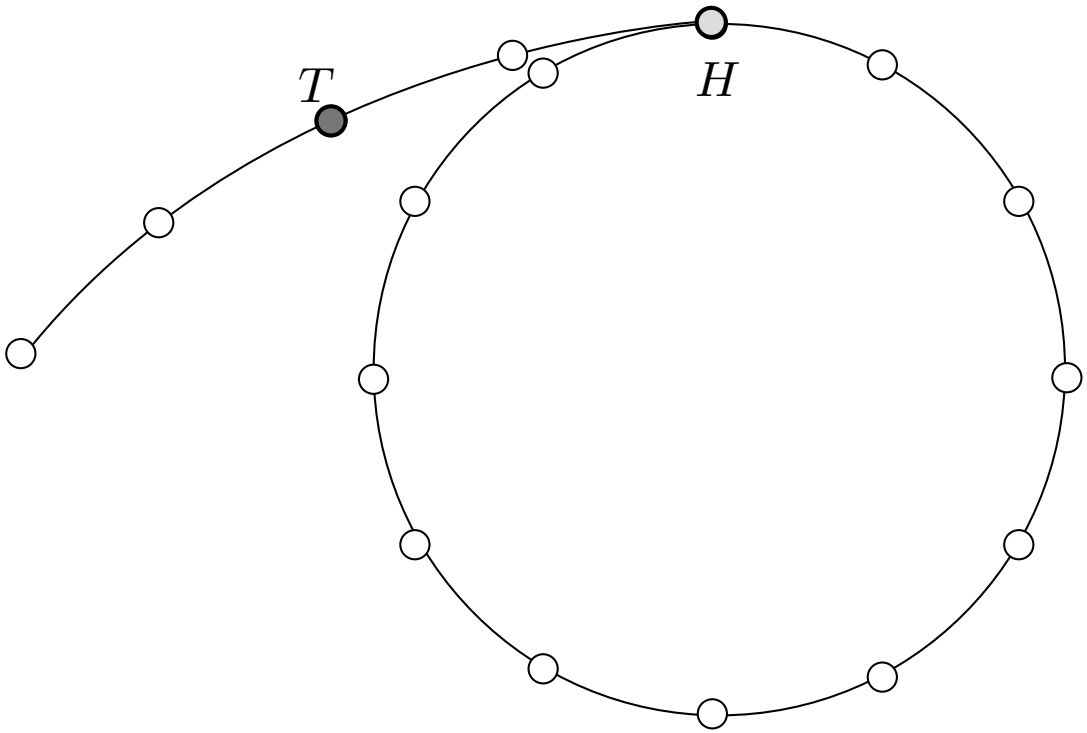
- Move tortoise 1 step; move hare 2 steps



Task 4: Collision Search

Floyd

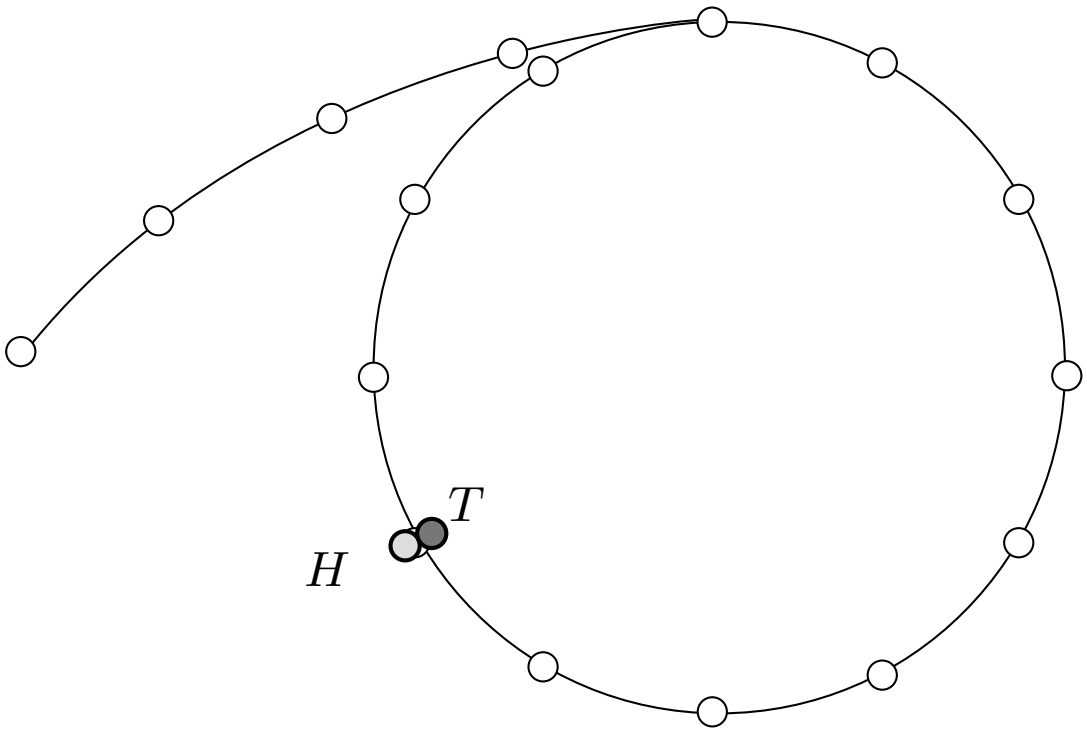
- Move tortoise 1 step; move hare 2 steps



Task 4: Collision Search

Floyd

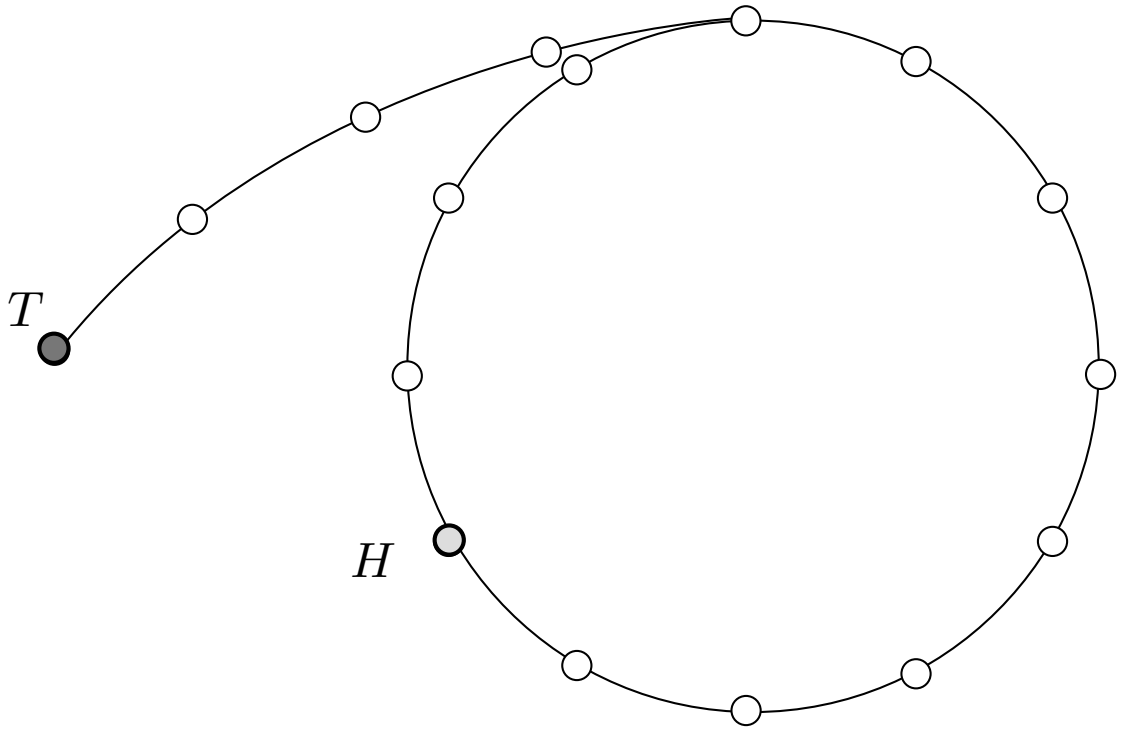
- Collide after 12 steps



Task 4: Collision Search

Floyd

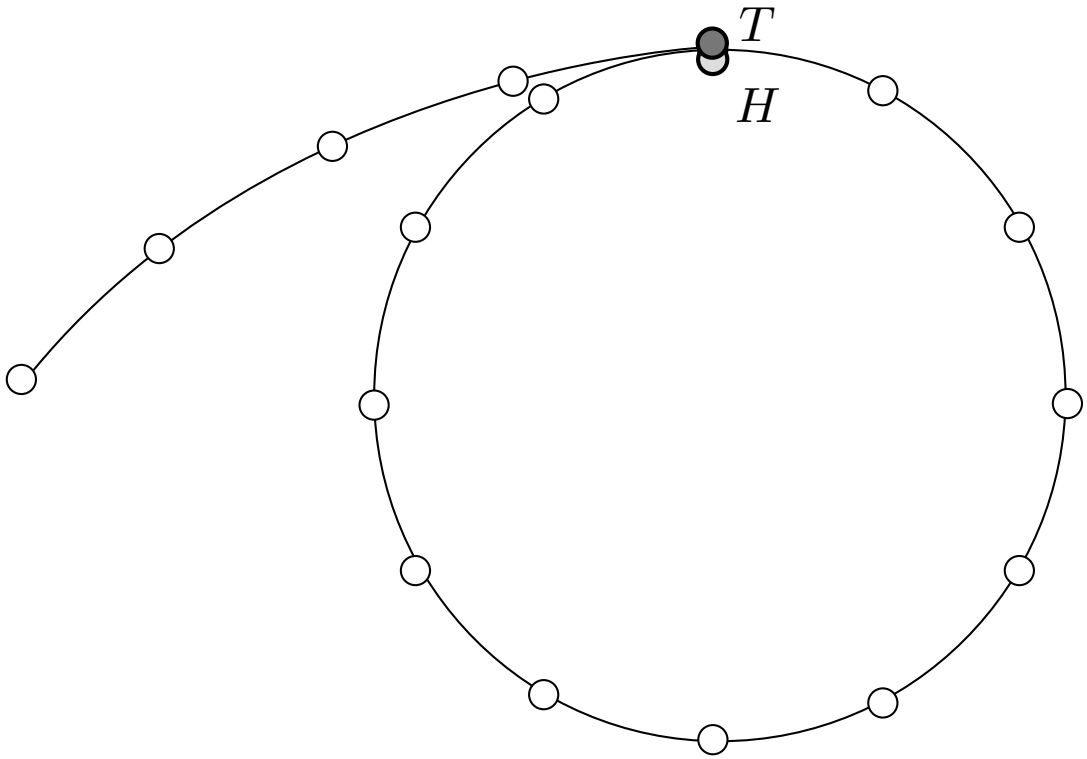
- Move tortoise to x_0



Task 4: Collision Search

Floyd

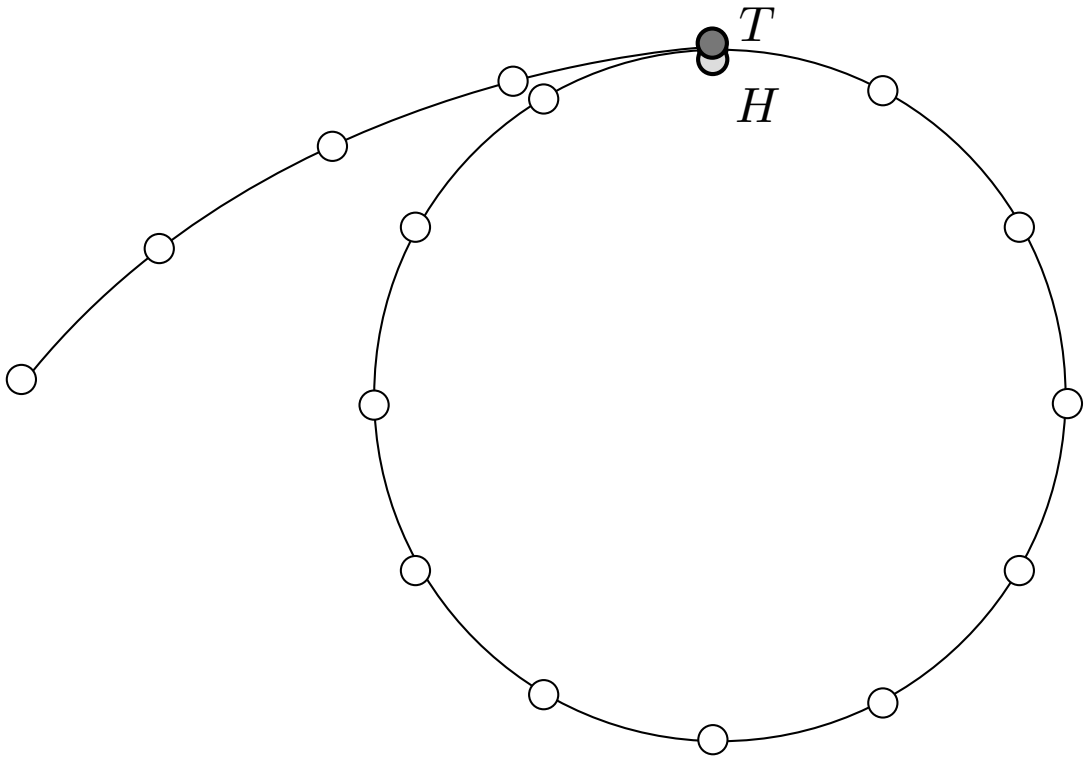
- Find collision and μ



Task 4: Collision Search

Floyd

- Let hare cycle once to find λ



Task 4: Collision Search

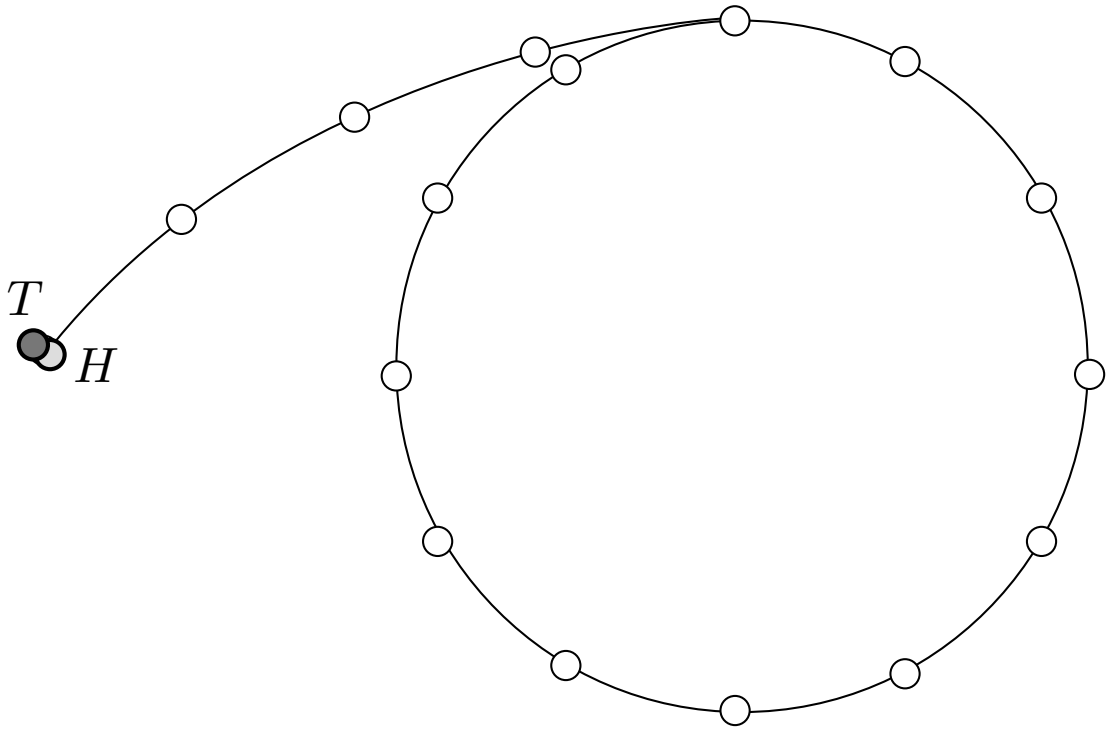
Brent

```
function BRENT(x_0)
    λ ← p ← 1
    T ← x_0
    H ← F(x_0))
    while T ≠ H do                          ▷ Loop #1: Find collision
        if p = λ then
            T ← H                           ▷ Move tortoise to hare
            p ← 2 · p, λ ← 0
        H ← F(H)                            ▷ Move hare 2^i steps
        λ ← λ + 1
    μ ← 0, T ← H ← x_0                      ▷ Reset tortoise and hare
    for i ← 0 to λ − 1 do                   ▷ Loop #2: Set distance to λ
        H ← F(H)                            ▷ Let hare cycle once
    while T ≠ H do                          ▷ Loop #3: Find cycle start
        T ← F(T)                            ▷ Let both meet at cycle start
        H ← F(H)
        μ ← μ + 1
    return (λ, μ)
```

Task 4: Collision Search

Brent

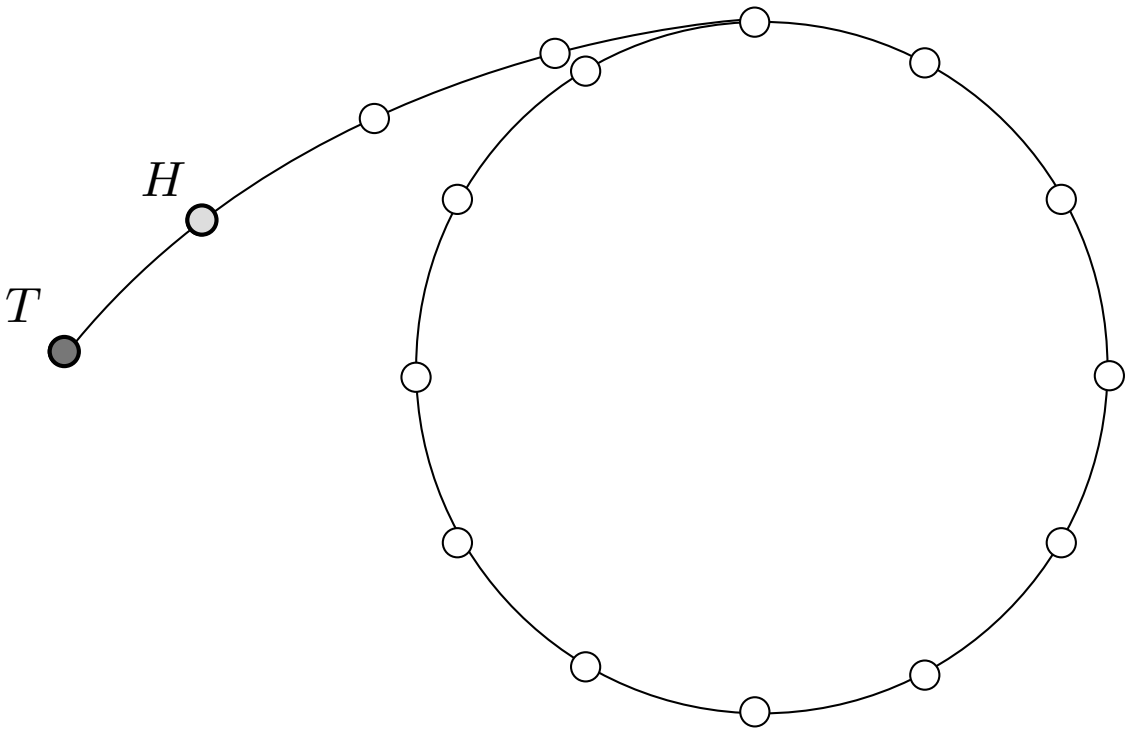
- Start at $T \leftarrow H \leftarrow x_0$



Task 4: Collision Search

Brent

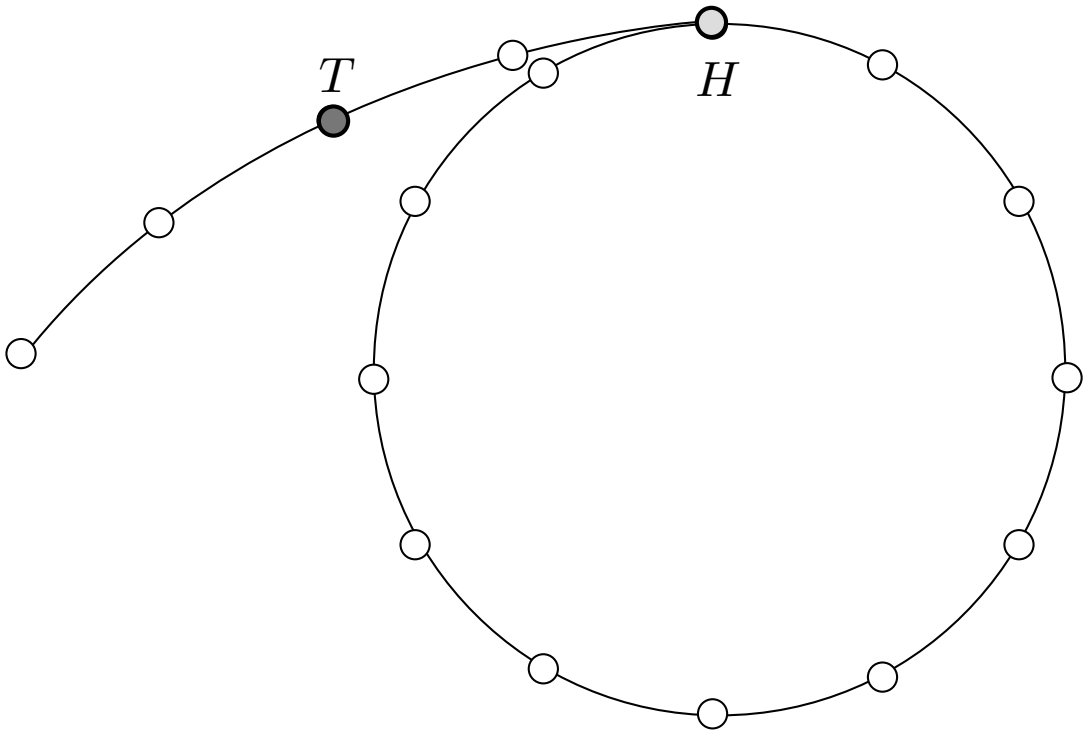
- Move hare 2^0 steps; move tortoise to hare



Task 4: Collision Search

Brent

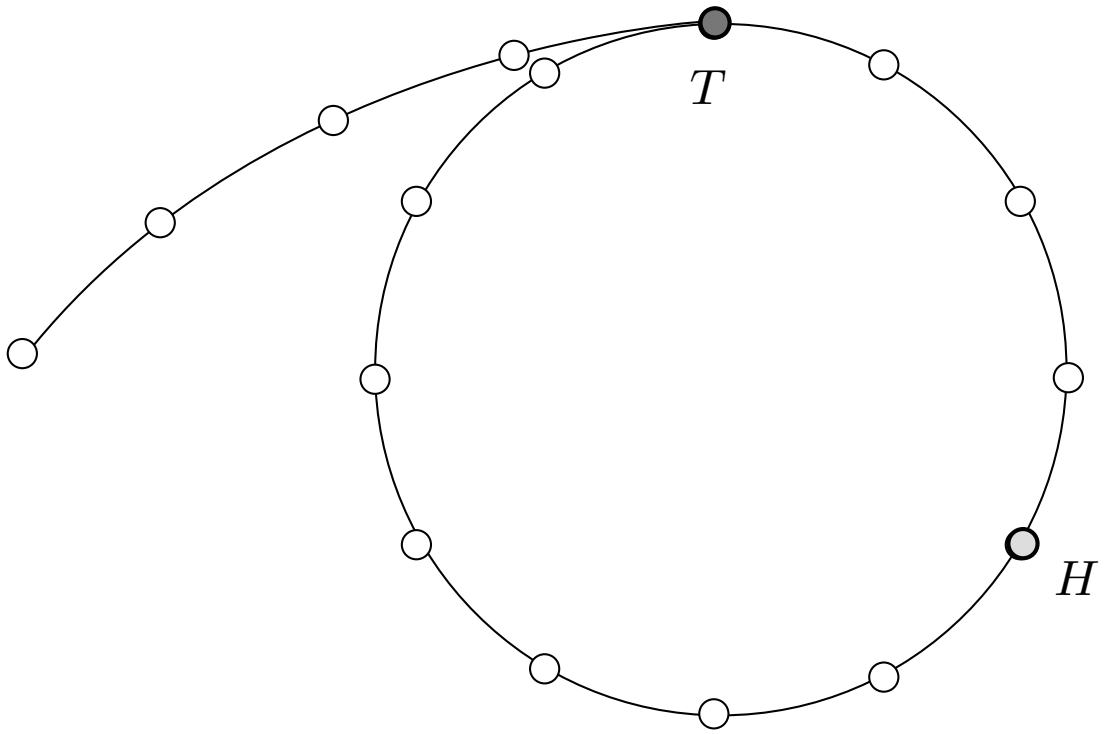
- Move hare 2^1 steps; move tortoise to hare



Task 4: Collision Search

Brent

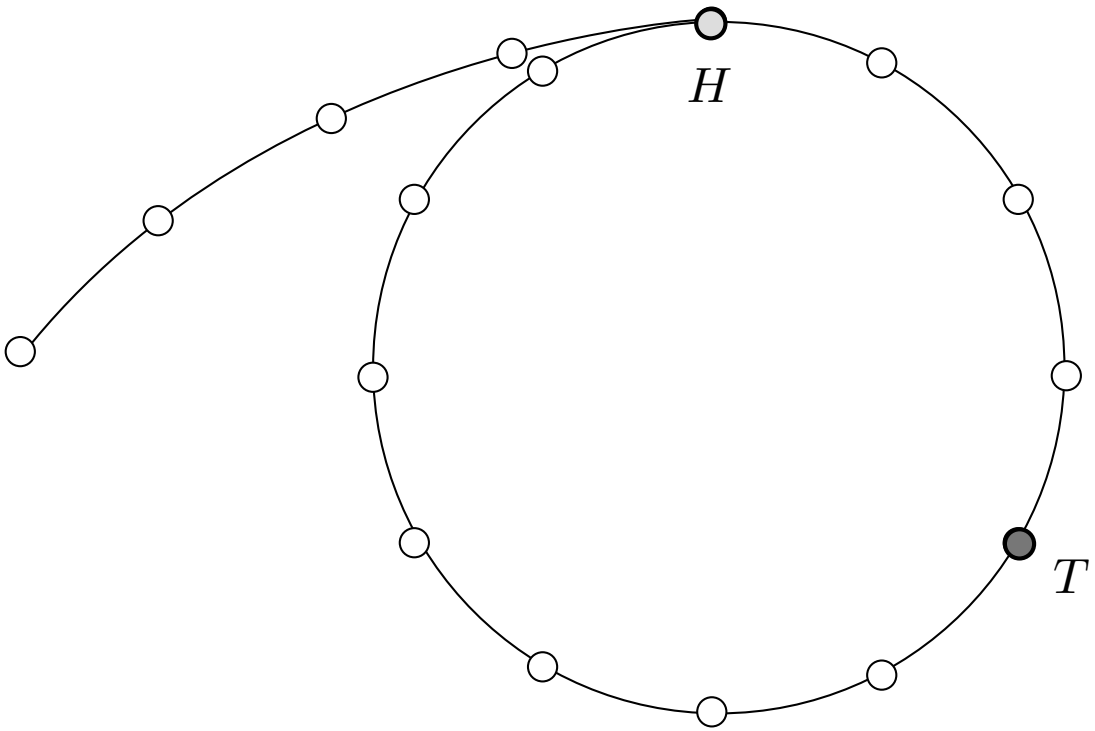
- Move hare 2^2 steps; move tortoise to hare



Task 4: Collision Search

Brent

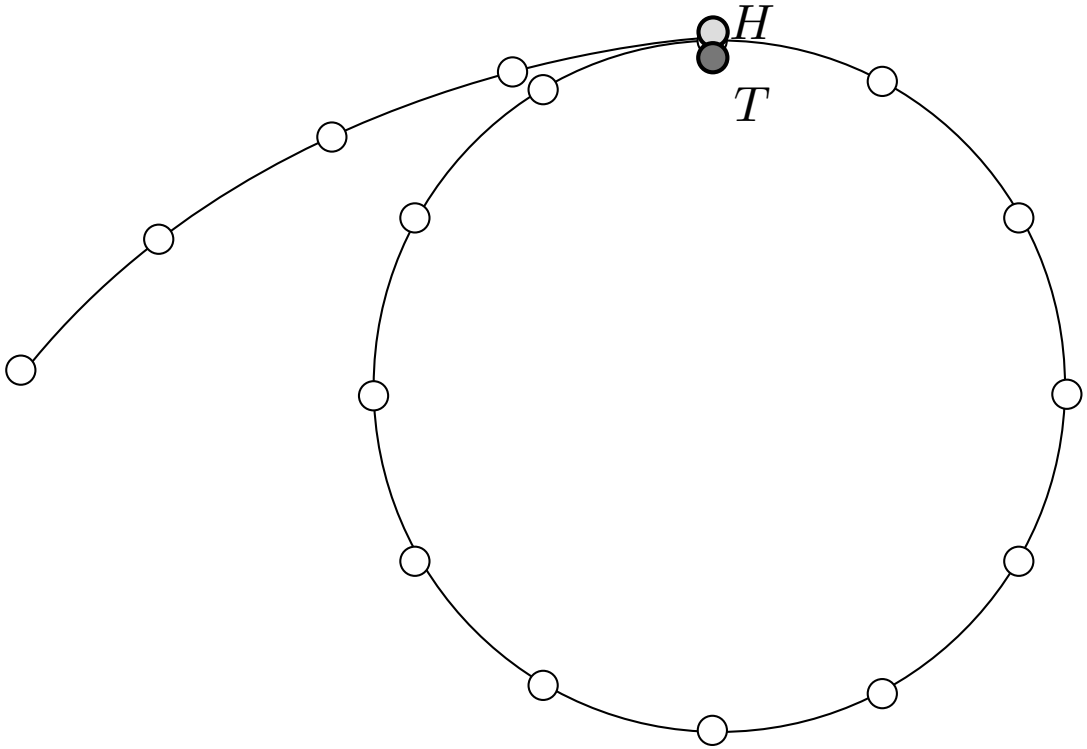
- Move hare 2^3 steps; move tortoise to hare



Task 4: Collision Search

Brent

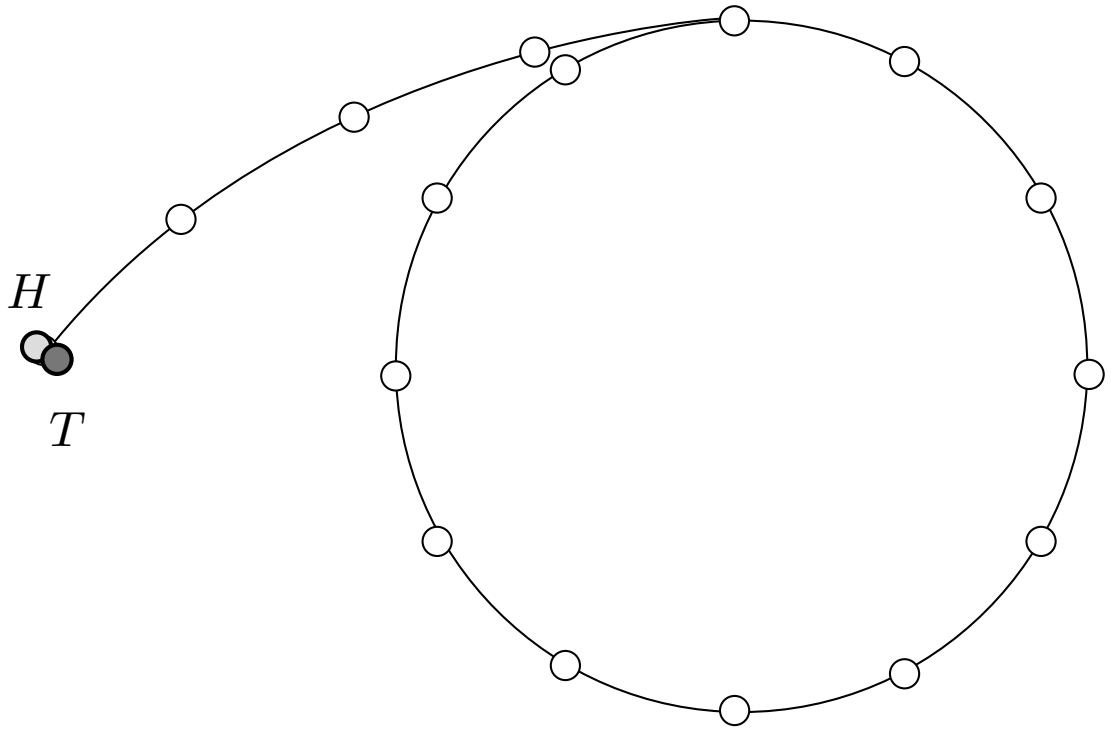
- Move hare 12 (up to 2^4 steps; collide, find λ



Task 4: Collision Search

Brent

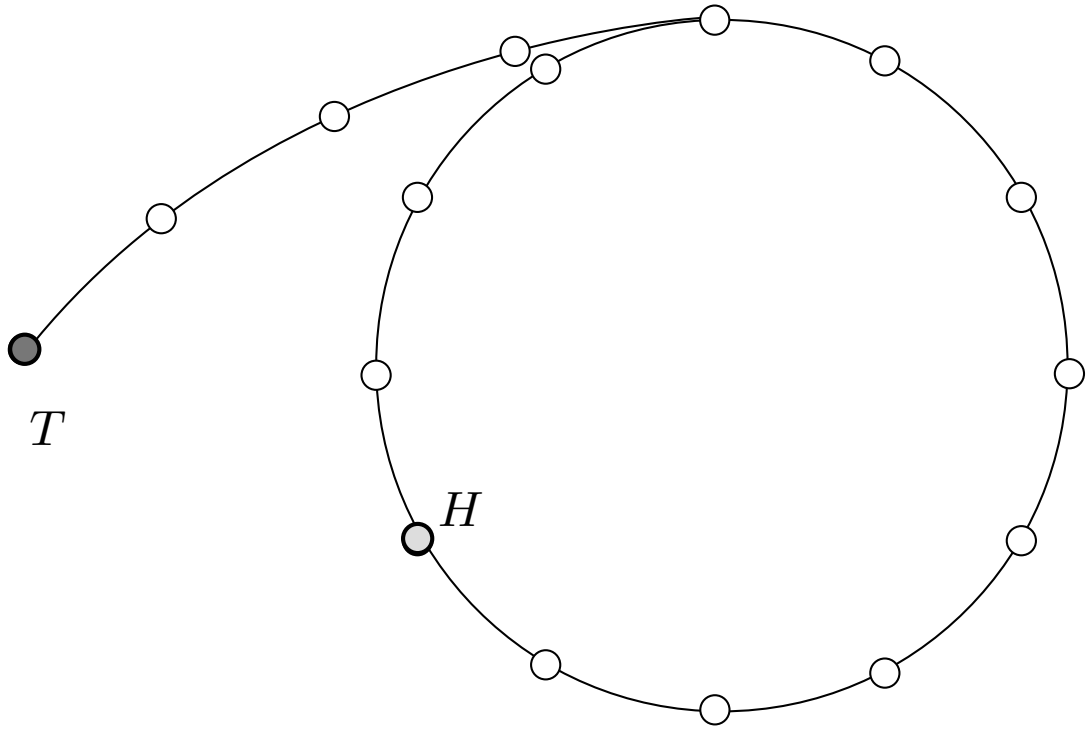
- Move tortoise and hare to x_0



Task 4: Collision Search

Brent

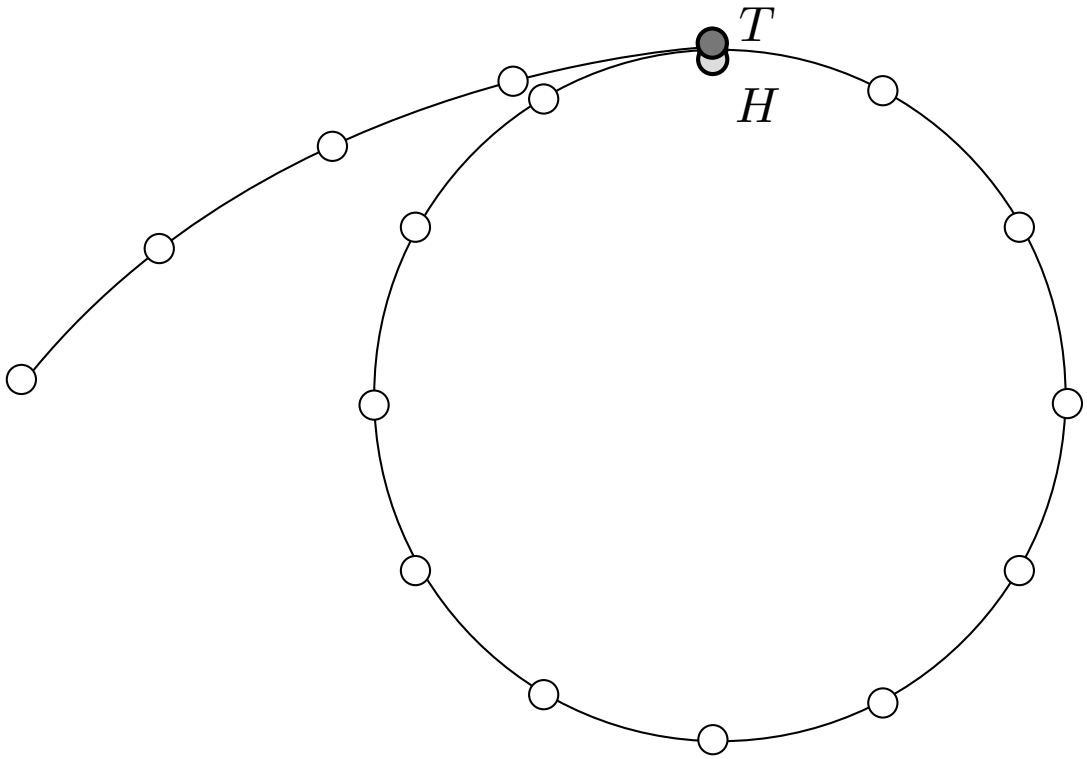
- Move hare λ steps (one cycle)



Task 4: Collision Search

Brent

- Move both to find collision and μ



Task 4: Collision Search

Distinguished Points

List	Walk
...	$f^{n_1}(x_1) = f(x_{n_1}) = \texttt{0x12340000}$

Task 4: Collision Search

Distinguished Points

List	Walk
...	$f^{n_1}(x_1) = f(x_{n_1}) = \texttt{0x12340000}$
$(\texttt{0x12340000}, n_1)$	

Task 4: Collision Search

Distinguished Points

List	Walk
...	$f^{n_1}(x_1) = f(x_{n_1}) = \texttt{0x12340000}$
$(\texttt{0x12340000}, n_1)$	$f(x_{n_2}) = \texttt{0x23410000}$

Task 4: Collision Search

Distinguished Points

List	Walk
...	$f^{n_1}(x_1) = f(x_{n_1}) = \texttt{0x12340000}$
$(\texttt{0x12340000}, n_1)$	$f(x_{n_2}) = \texttt{0x23410000}$
$(\texttt{0x23410000}, n_2)$	

Task 4: Collision Search

Distinguished Points

List	Walk
...	$f^{n_1}(x_1) = f(x_{n_1}) = \texttt{0x12340000}$
$(\texttt{0x12340000}, n_1)$	$f(x_{n_2}) = \texttt{0x23410000}$
$(\texttt{0x23410000}, n_2)$	$f(x_{n_{12}}) = \texttt{0x43210000}$
$(\texttt{0x43210000}, n_{12})$	

Task 4: Collision Search

Distinguished Points

List	Walk
...	$f^{n_1}(x_1) = f(x_{n_1}) = \texttt{0x12340000}$
$(\texttt{0x12340000}, n_1)$	$f(x_{n_2}) = \texttt{0x23410000}$
$(\texttt{0x23410000}, n_2)$	$f(x_{n_{12}}) = \texttt{0x43210000}$
$(\texttt{0x43210000}, n_{12})$	$f(x_{n_q}) = \texttt{0x23410000}$

There must be a collision.

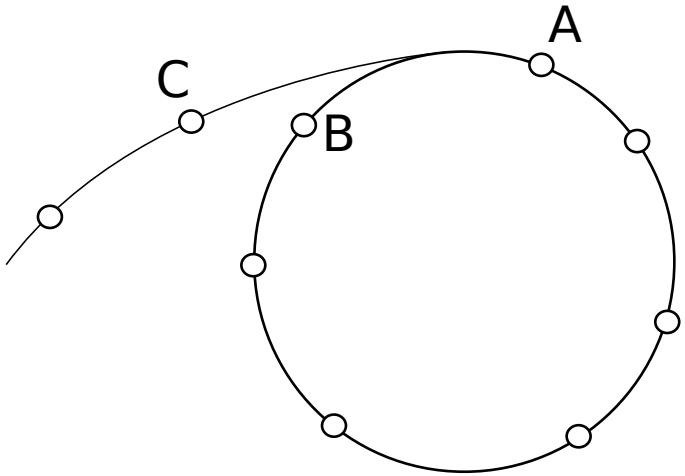
Task 4: Collision Search

Distinguished Points

List	Walk
...	$f^{n_1}(x_1) = f(x_{n_1}) = \texttt{0x12340000}$
C $= (\texttt{0x12340000}, n_1)$	$f(x_{n_2}) = \texttt{0x23410000}$
A $= (\texttt{0x23410000}, n_2)$	$f(x_{n_{12}}) = \texttt{0x43210000}$
B $= (\texttt{0x43210000}, n_{12})$	$f(x_{n_q}) = \texttt{0x23410000}$

- $d_C = n_2 - n_1, d_B = n_q - n_{12}$
- W.l.o.g.: $d_B > d_C$

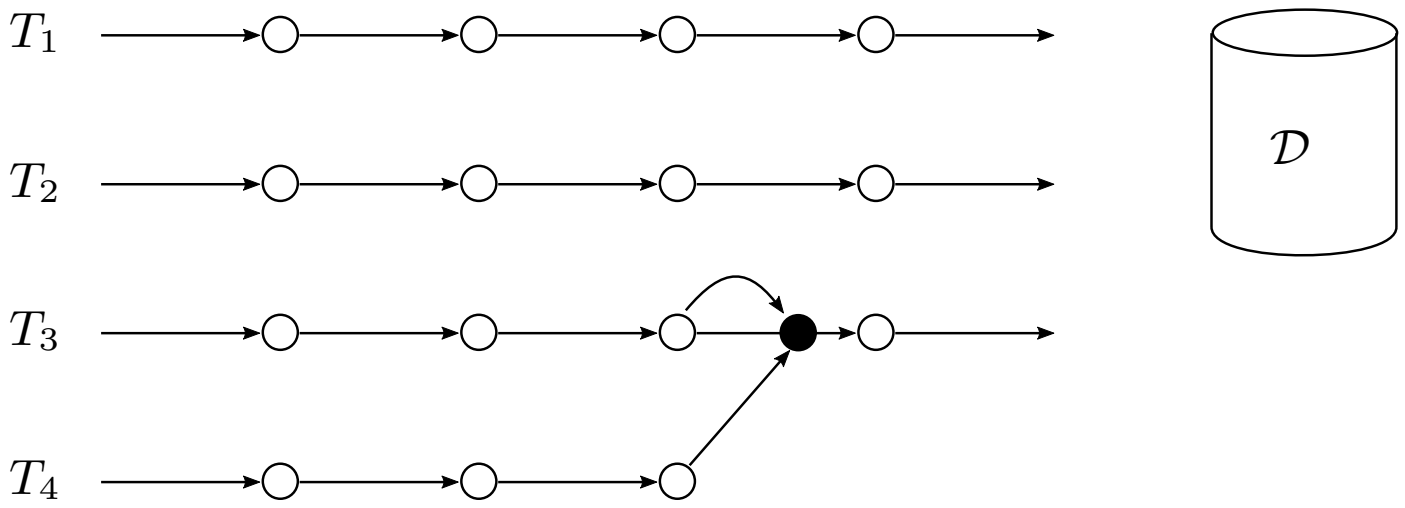
```
ctr ← 0
C ← f^{d_B − d_C}(C) // Now B and C are of equal distance to A
while B != C do
    B ← f(B)
    C ← f(C)
    ctr ← ctr + 1
λ ← nq − n_2
μ ← n_1 + d_b − d_c + ctr
```



Note that the collision can (and probably will) be before A!

Task 4: Collision Search

Distinguished Points



- Single database with distinguished points (DPs)
- 1.) Search collision on DPs
- 2.) Start from preceeding DP to identify real collision
- Linear speed-up

Questions?