Fast Forward Property
and
Decompositions of Graph of Functions

Tsung-Hsi Tsai

Academia Sinica

AofA2008, at Maresias, Brazil

April 18, 2008

Fast forward property

Definition **Let** $V = \{0, 1, ..., N-1\}$**. A function** $f : V \to V$ **is fast forward if for each** $m \in \mathbb{N}$ **and each** $x \in V$ **the computational complexity of evaluating the** m**th iterate of** f **at** x**,**

$$f^m(x), \quad \textbf{(e.g. } f^2(x) = f\left(f(x)\right)\textbf{)},$$

is small (polynomial in $\log N$**).**

Fast forward property

Definition **Let** $V = \{0, 1, ..., N-1\}$**. A function** $f : V \to V$ **is fast forward if for each** $m \in \mathbb{N}$ **and each** $x \in V$ **the computational complexity of evaluating the** m**th iterate of** f **at** x**,**

$$f^m(x), \quad \textbf{(e.g. } f^2(x) = f\left(f(x)\right)\textbf{)},$$

is small (polynomial in $\log N$**).**

Example **The cyclic permutation** $(0, 1, 2, ..., N-1)$ **is fast forward since** $f^m(x) = x + m$ **mod** N**.**

Applications

Evaluation of iteration of functions is useful in cryptographic applications.

Applications

Evaluation of iteration of functions is useful in cryptographic applications.

Example

Rho methods for factorization (Pollard, 1975).

There compute $f^m(x)$ repeatedly for the function $f(x) = x^2 - 1$ mod N.

The context

The scale of N is usually very large in cryptographic applications.

Here we consider the case that N is not very large.

The context

The scale of N is usually very large in cryptographic applications.

Here we consider the case that N is not very large.

Why?

In the computation of the iteration $f^m(x)$, we need to use $O(N)$ information which is generated in preprocess and stored in memory.

Preliminary

(Recall definitions and results in Tsaban 2003, 2007)

Fast forward permutation

Definition **The fast forward permutation coded by** $(m_0, m_1, ..., m_{\ell-1})$ **is**

$$\pi = \underbrace{(0, ..., s_0 - 1)}_{m_0} \underbrace{(s_0, ..., s_1 - 1)}_{m_1} \cdots \underbrace{(s_{\ell-2}, ..., N - 1)}_{m_{\ell-1}},$$

where $s_i = m_0 + \cdots + m_i$ **for** $0 \leq i \leq \ell - 2$**.**

Fast forward permutation

Definition **The fast forward permutation coded by** $(m_0, m_1, ..., m_{\ell-1})$ **is**

$$\pi = \underbrace{(0, ..., s_0 - 1)}_{m_0} \underbrace{(s_0, ..., s_1 - 1)}_{m_1} \cdots \underbrace{(s_{\ell-2}, ..., N - 1)}_{m_{\ell-1}},$$

where $s_i = m_0 + \cdots + m_i$ **for** $0 \leq i \leq \ell - 2$**.**

Computation of the iteration

$$\pi^m(x) = s_{i(x)} + \left(x - s_{i(x)} + m \bmod \left(s_{i(x)+1} - s_{i(x)}\right)\right),$$

where $s_{i(x)} \leq x < s_{i(x)+1}$ **and the assignments** $x \longrightarrow i(x)$ **and** $i \longrightarrow s_i$ **are preprocessed as lookup tables.**

Arbitrary permutation

For any permutation

$$f = \underbrace{(b_0, b_1, ..., b_{s_0-1})}_{m_0} \underbrace{(b_{s_0}, ..., b_{s_1-1})}_{m_1} \cdots \underbrace{(b_{s_{\ell-2}}, ..., b_{N-1})}_{m_{\ell-1}},$$

define $\sigma(x) = b_x$ **for** $x \in V$**. Then**

$$f = \sigma \circ \pi \circ \sigma^{-1} \quad \textbf{and} \quad f^m = \sigma \circ \pi^m \circ \sigma^{-1}.$$

Arbitrary permutation

For any permutation

$$f = \underbrace{(b_0, b_1, ..., b_{s_0-1})}_{m_0} \underbrace{(b_{s_0}, ..., b_{s_1-1})}_{m_1} \cdots \underbrace{(b_{s_{\ell-2}}, ..., b_{N-1})}_{m_{\ell-1}},$$

define $\sigma(x) = b_x$ **for** $x \in V$**. Then**

$$f = \sigma \circ \pi \circ \sigma^{-1} \quad \textbf{and} \quad f^m = \sigma \circ \pi^m \circ \sigma^{-1}.$$

Thus, f **is fast forward if lookup tables for** σ**,** σ^{-1}**,** $i(x)$ **and** s_i **are stored in memory.**

Fast forward function

Definition **The fast forward function coded by** $(m_0, m_1, ..., m_{\ell-1})$ **and an auxiliary sequence** $(p_0, ..., p_{\ell-1})$**,** $0 \leq p_i < s_i$**, is**

$$\theta(x) = \begin{cases} p_i & \text{if } x = s_i - 1, \\ \pi(x) & \text{otherwise,} \end{cases}$$

where π **is the fast forward permutation codes by** $(m_0, m_1, ..., m_{\ell-1})$**.**

Fast forward function

Definition **The fast forward function coded by $(m_0, m_1, ..., m_{\ell-1})$ and an auxiliary sequence $(p_0, ..., p_{\ell-1})$, $0 \leq p_i < s_i$, is**

$$\theta(x) = \begin{cases} p_i & \text{if } x = s_i - 1, \\ \pi(x) & \text{otherwise,} \end{cases}$$

where π is the fast forward permutation codes by $(m_0, m_1, ..., m_{\ell-1})$.

Note that a cycle of π, $(s_{i-1}, ..., s_i - 1)$, is connected to a previous cycle by the mapping $\theta(s_i - 1) = p_i$ if $p_i < s_{i-1}$. We call this a descent.

Evaluating $\theta^m(x)$

Case 1. if m is small ($s_{i(x)} \leq x + m < s_{i(x)+1}$), then

$$\theta^m(x) = x + m.$$

Evaluating $\theta^m(x)$

Case 1. if m is small ($s_{i(x)} \leq x + m < s_{i(x)+1}$), then

$$\theta^m(x) = x + m.$$

Case 2. if $x + m \geq s_{i(x)+1}$ and $p_{i(x)+1} \geq s_{i(x)}$. Then

$$\theta^m(x) = p_{i(x)+1} + \Big((x + m - s_{i(x)+1}) \textbf{ mod } (s_{i(x)+1} - p_{i(x)+1}) \Big) .$$

Evaluating $\theta^m(x)$

Case 1. if m is small ($s_{i(x)} \le x + m < s_{i(x)+1}$), then

$$\theta^m(x) = x + m.$$

Case 2. if $x + m \ge s_{i(x)+1}$ and $p_{i(x)+1} \ge s_{i(x)}$. Then

$$\theta^m(x) = p_{i(x)+1} + \big((x + m - s_{i(x)+1}) \text{ mod } (s_{i(x)+1} - p_{i(x)+1})\big).$$

Case 3. if $x + m \ge s_{i(x)+1}$ and $p_{i(x)+1} < s_{i(x)}$. Then $\theta^m(x)$ is computed recursively by

$$\theta^m(x) = \theta^{x+m-s_{i(x)+1}}(p_{i(x)+1})$$

(here $\theta^0 = I$). (case of descent)

The complexity

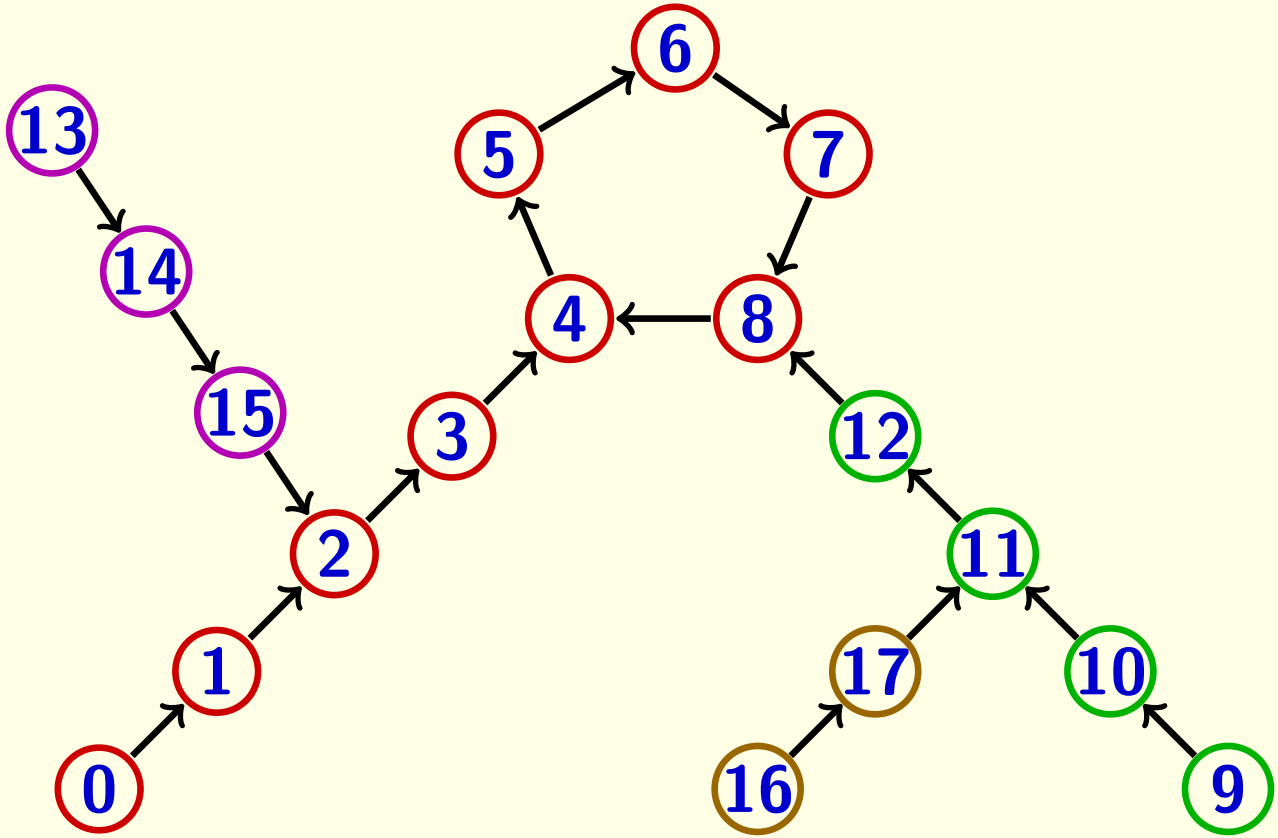
Theorem **(Tsaban, 2007)**

The complexity of evaluating $\theta^m(x)$ is measured by the number of descents (recursions) on the tour $x, \theta(x), \theta^2(x), \theta^3(x), ...$

Notation of the number of descents: $D_\theta(x)$.

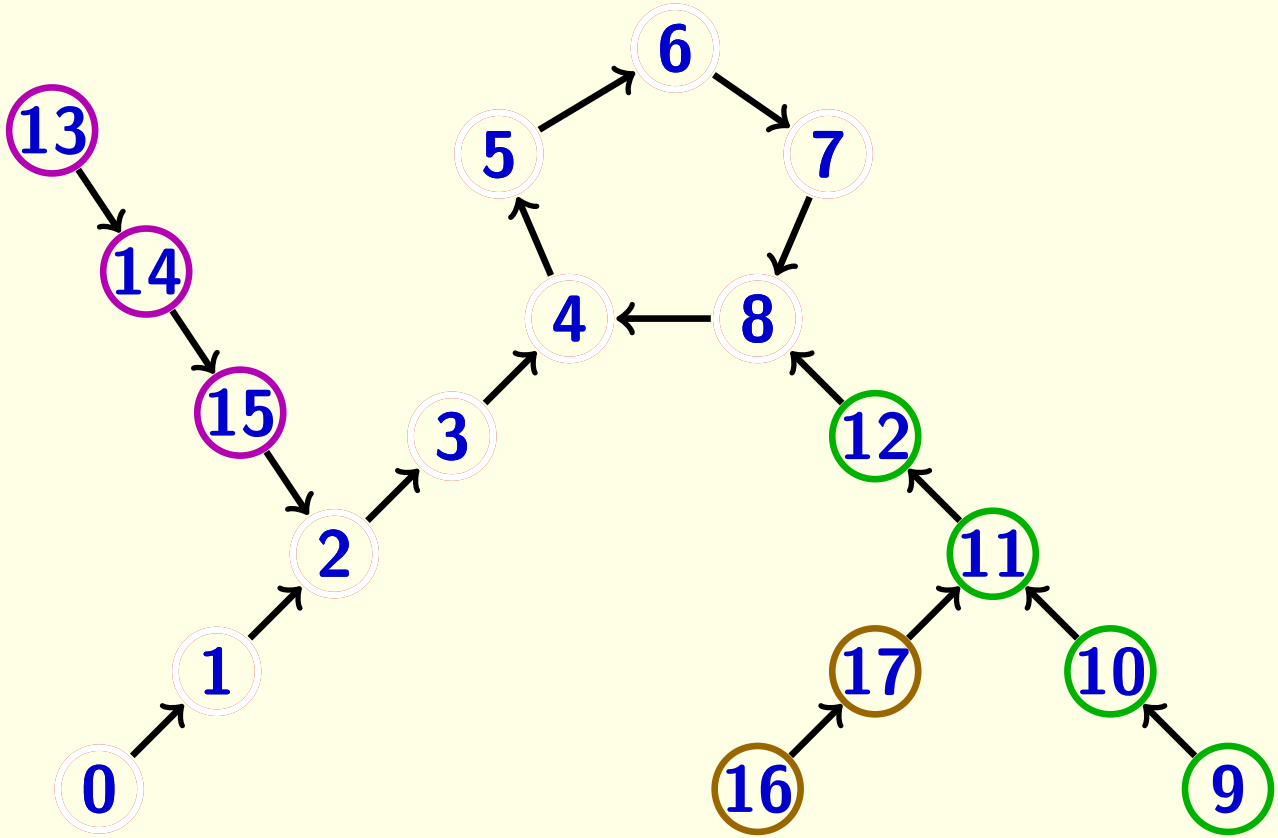
Example for number of descents

$D_\theta(x) = 0$ **for** $x = 0, ..., 8$**;**



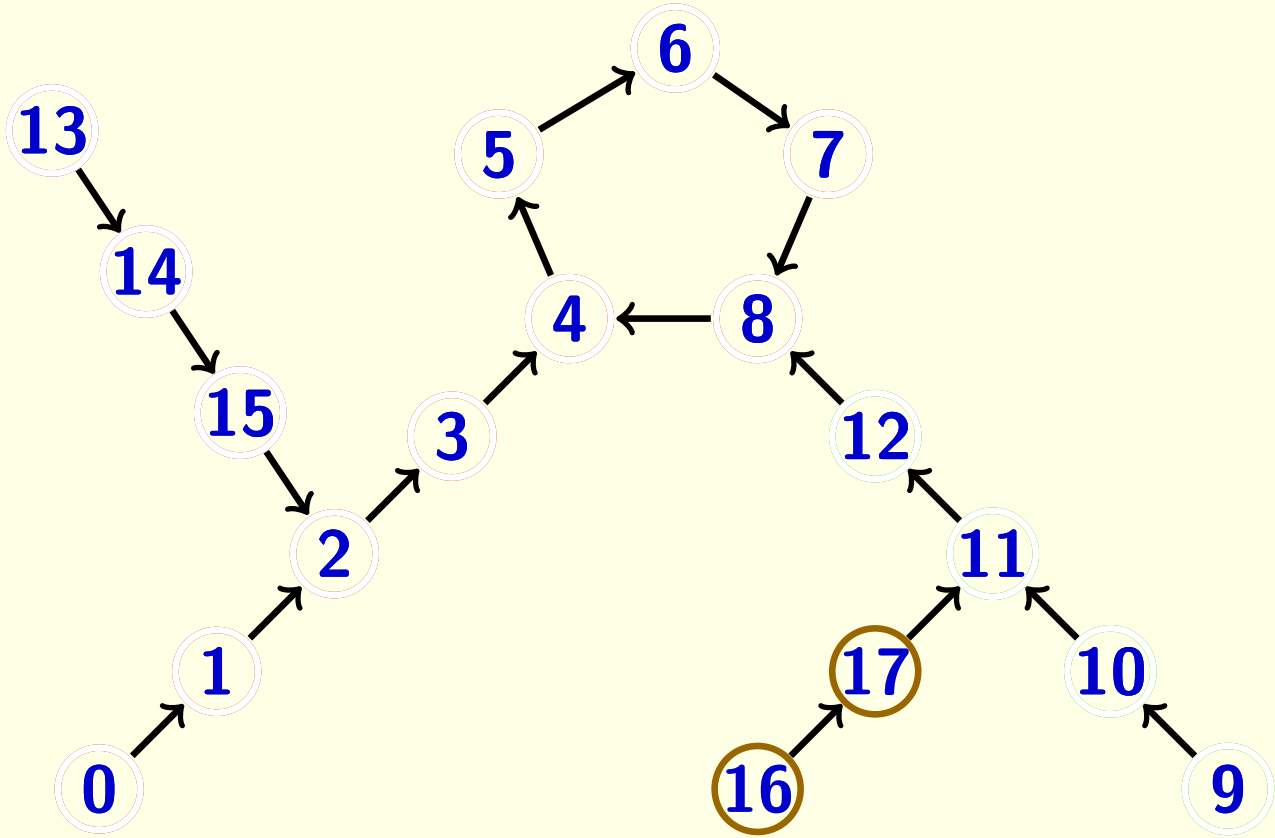
Example for number of descents

$D_\theta(x) = 0$ **for** $x = 0, ..., 8$**;** $D_\theta(x) = 1$ **for** $x = 9, ..., 15$**;**



Example for number of descents

$D_\theta(x) = 0$ **for** $x = 0, ..., 8$**;** $D_\theta(x) = 1$ **for** $x = 9, ..., 15$**;**
$D_\theta(x) = 2$ **for** $x = 16, 17$**.**



Arbitrary function

If there is a fast forward function θ such that

$$f = \sigma \circ \theta \circ \sigma^{-1},$$

then

$$f^m = \sigma \circ \theta^m \circ \sigma^{-1}$$

and the complexity of evaluating $f^m(x)$ is $O(D_\theta(x) + 1)$.

Arbitrary function

If there is a fast forward function θ such that

$$f = \sigma \circ \theta \circ \sigma^{-1},$$

then

$$f^m = \sigma \circ \theta^m \circ \sigma^{-1}$$

and the complexity of evaluating $f^m(x)$ is $O(D_\theta(x) + 1)$.

How to find such a θ?

An approach proposed by Tsaban

Orbit decomposition

Definition **The orbit of an element x in $U \subset V$ is the simple tour $\{x, f(x), f^2(x), ..., f^k(x)\}$ such that $f^{k+1}(x) = f^i(x)$ for some $0 \leq i \leq k$ or $f^{k+1}(x) \notin U$.**

An approach proposed by Tsaban

Orbit decomposition

Definition **The orbit of an element x in $U \subset V$ is the simple tour $\{x, f(x), f^2(x), ..., f^k(x)\}$ such that $f^{k+1}(x) = f^i(x)$ for some $0 \leq i \leq k$ or $f^{k+1}(x) \notin U$.**

Definition **An orbit decomposition of f is $C_0, C_1, ..., C_{\ell-1}$ defined as follows: C_0 is an orbit in V and C_i is an orbit in $V - C_0 \cup \cdots \cup C_{i-1}$ for $i > 0$.**

Given an arbitrary function f, and

$$\underbrace{(b_0, b_1, ..., b_{s_0-1})}_{C_0} \quad \underbrace{(b_{s_0}, ..., b_{s_1-1})}_{C_1} \quad \ldots \quad \underbrace{(b_{s_{\ell-2}}, ..., b_{N-1})}_{C_{\ell-1}}$$

is an orbit decomposition of f.

Given an arbitrary function f, and

$$\underbrace{(b_0, b_1, ..., b_{s_0-1})}_{C_0} \quad \underbrace{(b_{s_0}, ..., b_{s_1-1})}_{C_1} \quad \ldots \quad \underbrace{(b_{s_{\ell-2}}, ..., b_{N-1})}_{C_{\ell-1}}$$

is an orbit decomposition of f. Then

$$f = \sigma \circ \theta \circ \sigma^{-1},$$

where $\sigma(x) = b_x$ and θ is the fast forward function coded by $(|C_0|, |C_1|, ..., |C_{\ell-1}|)$ and the auxiliary sequence $(p_0, ..., p_{\ell-1})$ such that $f(b_{s_i-1}) = b_{p_i}$.

Example **Let f be the function:**

$9 \to 8 \to 7 \to 6 \to 5 \to 4 \to 3 \to 2 \to 1 \to 0 \circlearrowleft$

Example **Let f be the function:**

$9 \to 8 \to 7 \to 6 \to 5 \to 4 \to 3 \to 2 \to 1 \to 0 \circlearrowleft$

An orbit decomposition of f can be

$(9, 8, 7, 6, 5, 4, 3, 2, 1, 0)$**,** $\quad D_\theta(9) = 0$**,**

Example **Let f be the function:**

$9 \to 8 \to 7 \to 6 \to 5 \to 4 \to 3 \to 2 \to 1 \to 0 \circlearrowleft$

An orbit decomposition of f can be

$(9, 8, 7, 6, 5, 4, 3, 2, 1, 0)$**,** $\quad D_\theta(9) = 0$**,**

or $(4, 3, 2, 1, 0)(9, 8, 7, 6, 5)$**,** $\quad D_\theta(9) = 1$**,**

Example **Let f be the function:**

$9 \to 8 \to 7 \to 6 \to 5 \to 4 \to 3 \to 2 \to 1 \to 0 \circlearrowleft$

An orbit decomposition of f can be

$(9,8,7,6,5,4,3,2,1,0)$**,** $\quad D_\theta(9) = 0$**,**

or $(4,3,2,1,0)(9,8,7,6,5)$**,** $\quad D_\theta(9) = 1$**,**

or $(1,0)(4,3,2)(6,5)(9,8,7)$**,** $\quad D_\theta(9) = 3$**,**

Example **Let f be the function:**

$9 \to 8 \to 7 \to 6 \to 5 \to 4 \to 3 \to 2 \to 1 \to 0 \circlearrowleft$

An orbit decomposition of f can be

$(9,8,7,6,5,4,3,2,1,0)$**,** $\quad D_\theta(9) = 0$**,**

or $(4,3,2,1,0)(9,8,7,6,5)$**,** $\quad D_\theta(9) = 1$**,**

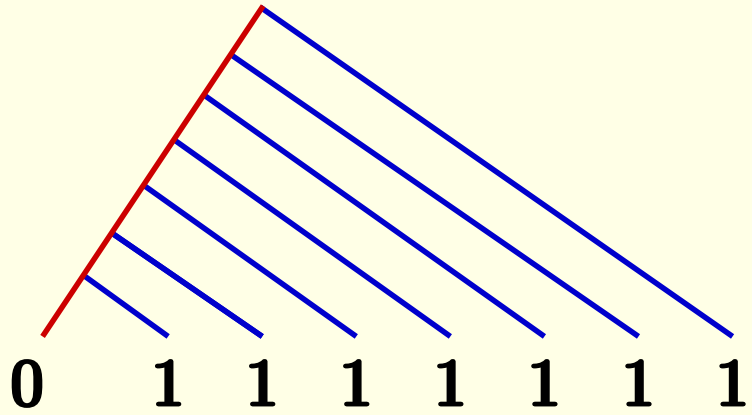
or $(1,0)(4,3,2)(6,5)(9,8,7)$**,** $\quad D_\theta(9) = 3$**,**

... or even the worst

$(0)(1)(2)(3)(4)(5)(6)(7)(8)(9)$**,** $\quad D_\theta(9) = 9$**.**

Example **Two different orbit decompositions:**

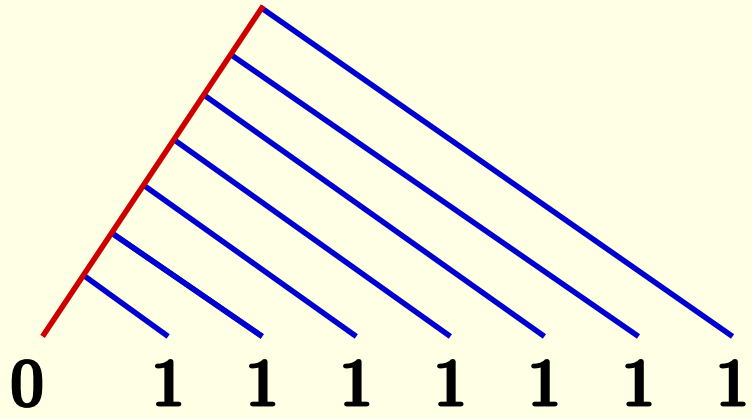
(here the numbers indicate the numbers of descents)



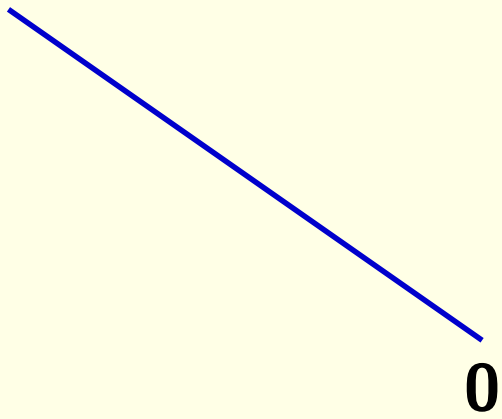
good **poor**

Example **Two different orbit decompositions:**

(here the numbers indicate the numbers of descents)



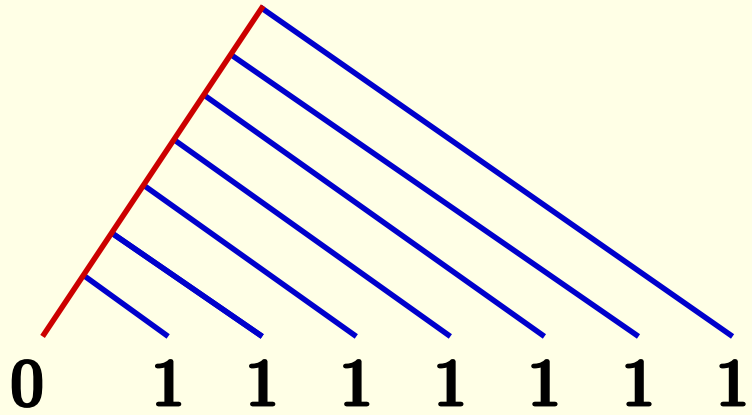
good



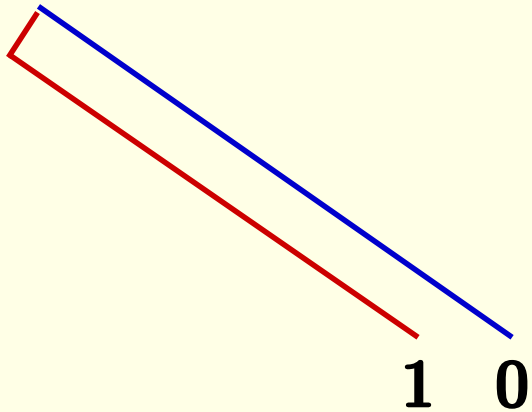
poor

Example **Two different orbit decompositions:**

(here the numbers indicate the numbers of descents)



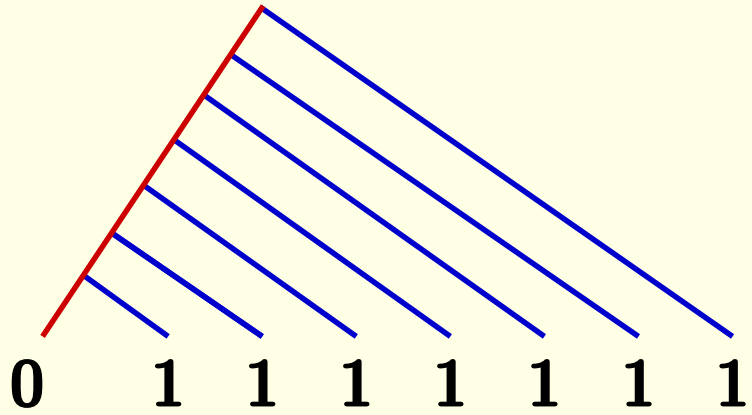
good



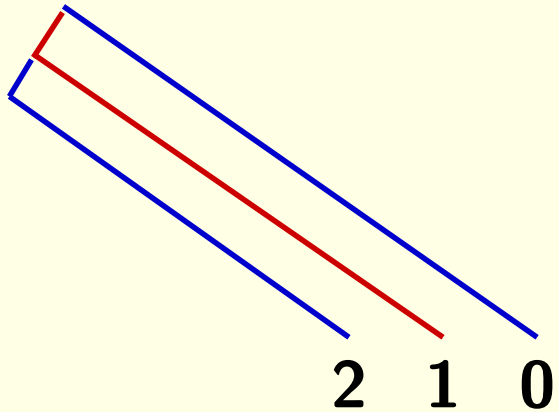
poor

Example **Two different orbit decompositions:**

(here the numbers indicate the numbers of descents)



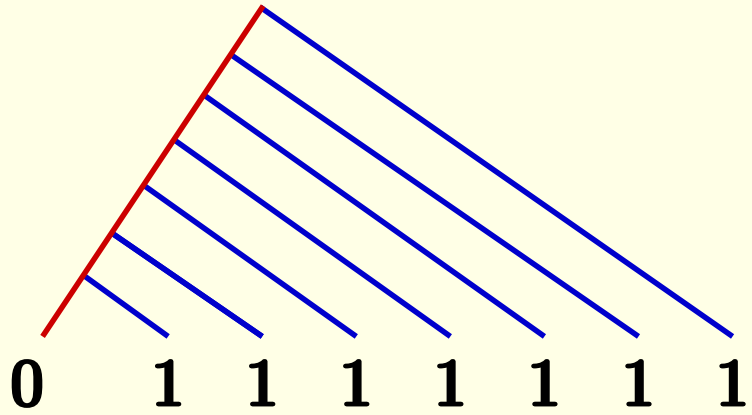
good



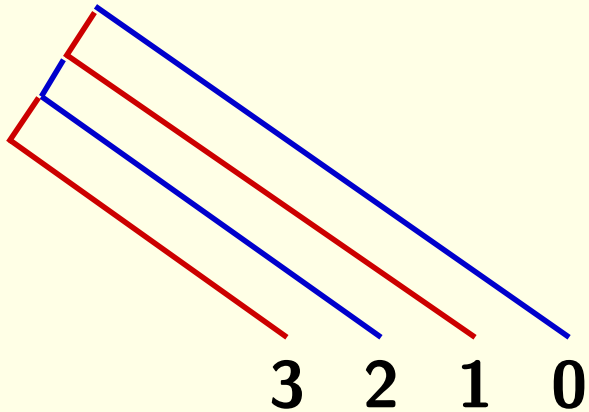
poor

Example **Two different orbit decompositions:**

(here the numbers indicate the numbers of descents)



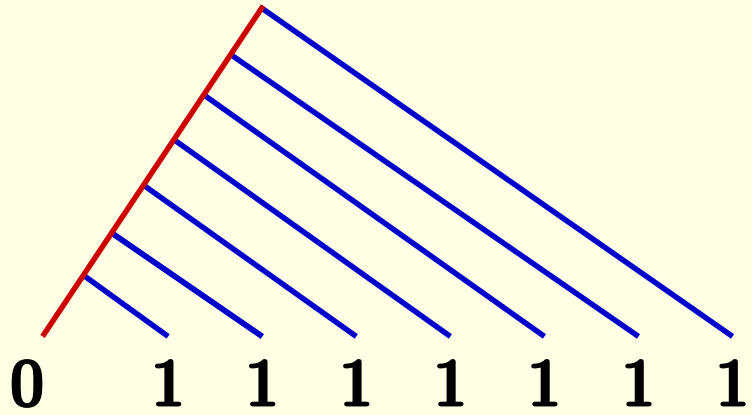
good



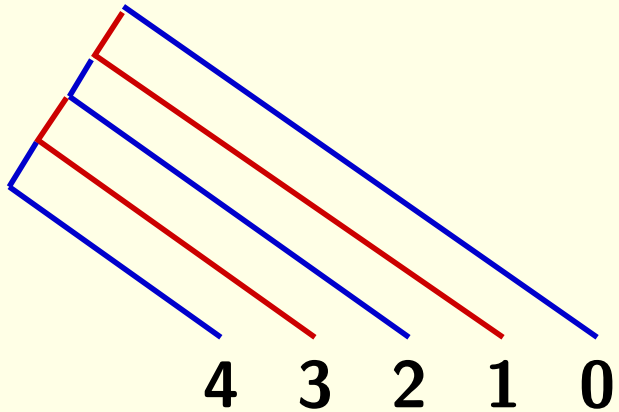
poor

Example **Two different orbit decompositions:**

(here the numbers indicate the numbers of descents)



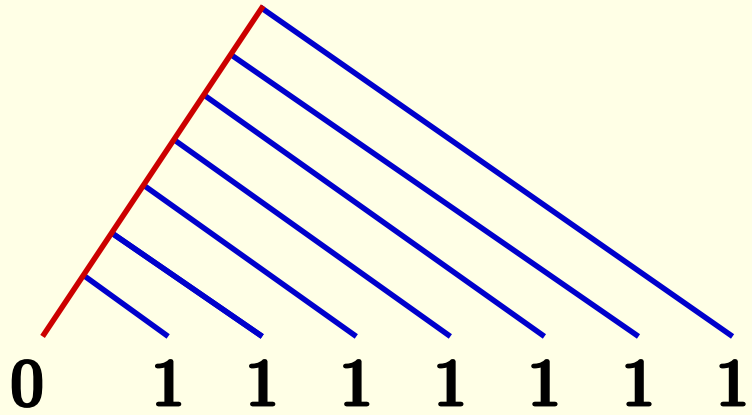
good



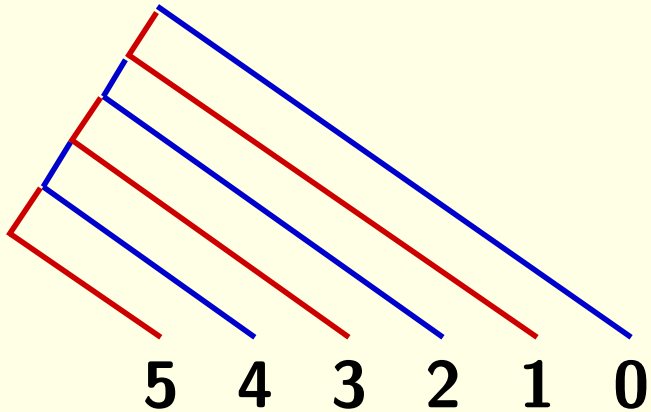
poor

Example **Two different orbit decompositions:**

(here the numbers indicate the numbers of descents)



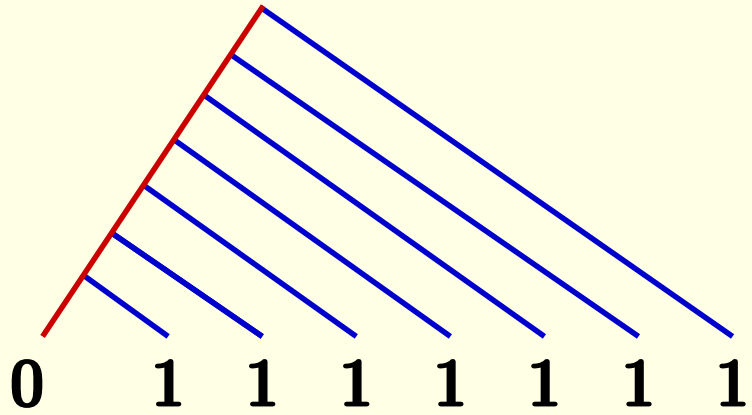
good



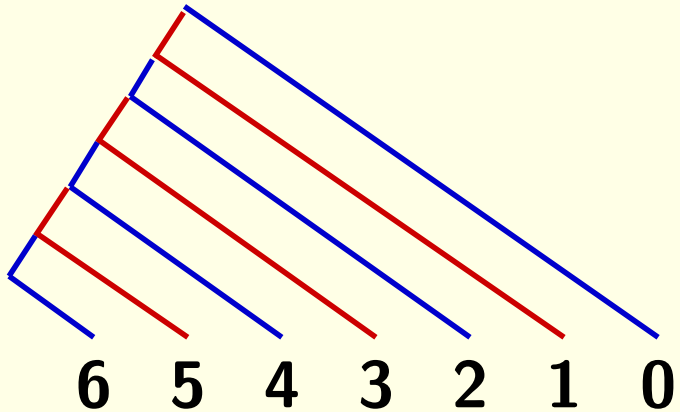
poor

Example **Two different orbit decompositions:**

(here the numbers indicate the numbers of descents)



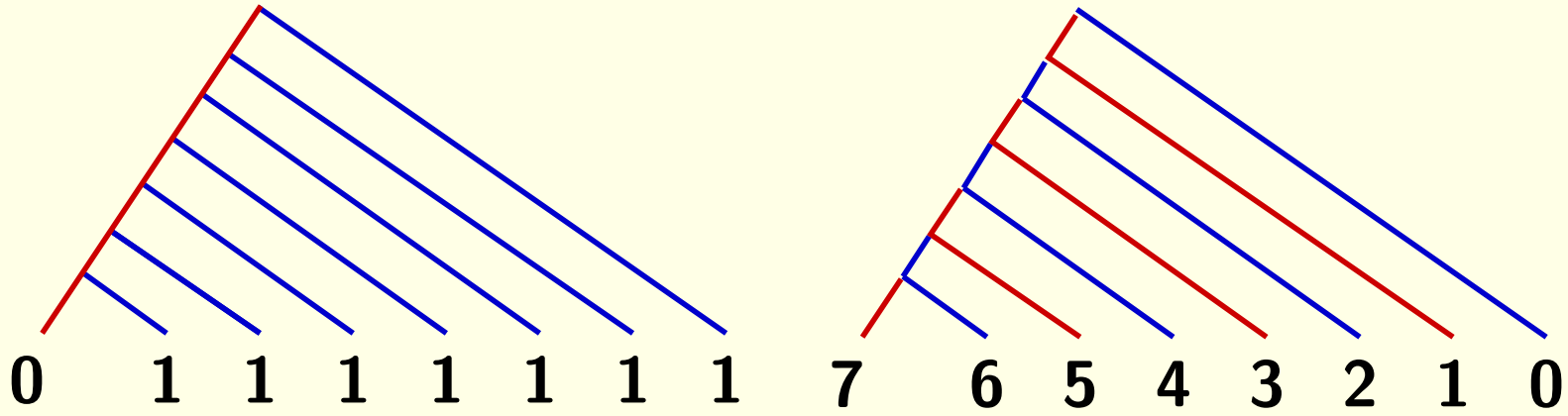
good



poor

Example **Two different orbit decompositions:**

(here the numbers indicate the numbers of descents)



good **poor**

Greedy orbit decomposition

Definition **(Tsaban, 2007)**

The greedy orbit decomposition of f is $C_0, C_1, ..., C_{\ell-1}$ defined as follows:

Greedy orbit decomposition

Definition **(Tsaban, 2007)**

The greedy orbit decomposition of f is $C_0, C_1, ..., C_{\ell-1}$ defined as follows:

C_0 is the maximal length orbit in V

Greedy orbit decomposition

Definition **(Tsaban, 2007)**

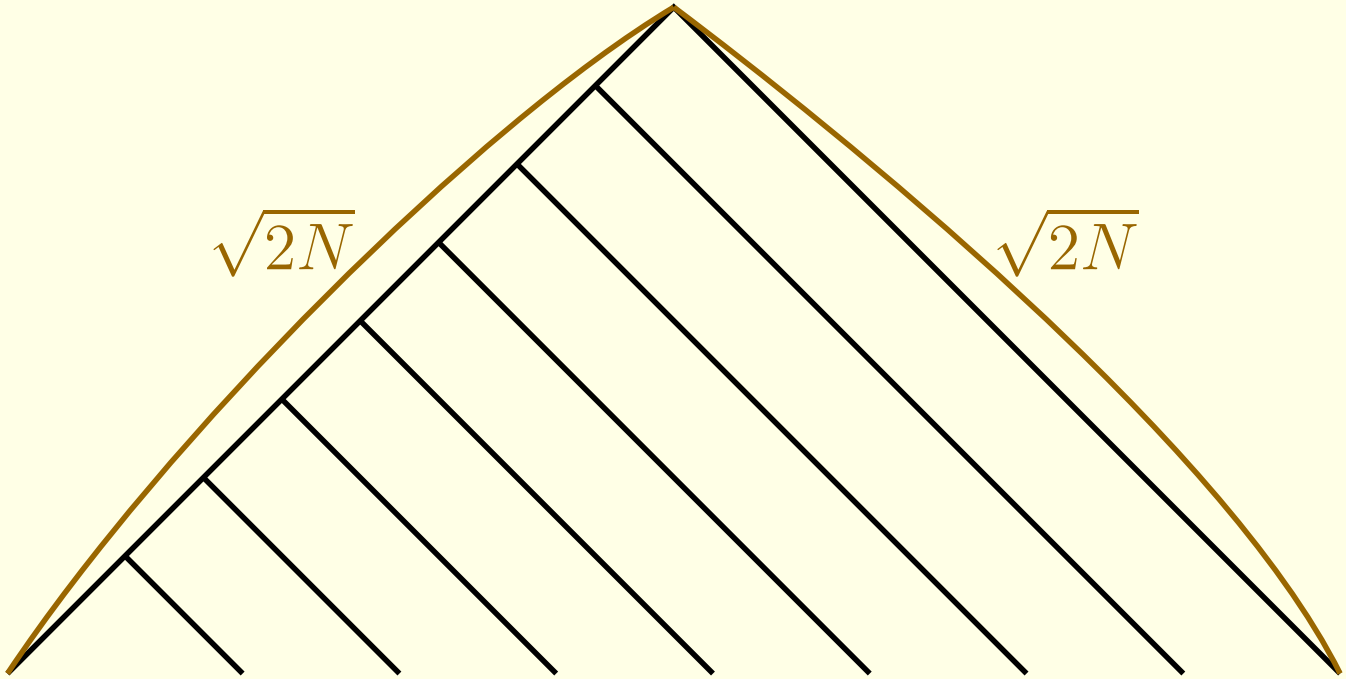
The greedy orbit decomposition of f is $C_0, C_1, ..., C_{\ell-1}$ defined as follows:

C_0 is the maximal length orbit in V and

C_i is the maximal length orbit in $V - C_0 \cup \cdots \cup C_{i-1}$ for $i > 0$.

The worst case

The maximal number of descents for greedy orbit decomposition is about $\sqrt{2N}$.



The average case

The experiment in Tsaban 2007 indicates that the expected number of descents for the greedy orbit decomposition is about

$$\frac{\log_2 N}{5}$$

for random functions and random points.

Problems in Tsaban 2007

1. "Does there exist an efficient algorithm to find an orbit decomposition for which the maximal number of descents is as small as it can be?"

Problems in Tsaban 2007

1. "Does there exist an efficient algorithm to find an orbit decomposition for which the maximal number of descents is as small as it can be?" That is to find a θ which minimizes

$$\max_{x \in V} D_\theta(x).$$

Problems in Tsaban 2007

1. "Does there exist an efficient algorithm to find an orbit decomposition for which the maximal number of descents is as small as it can be?" That is to find a θ which minimizes

$$\max_{x \in V} D_\theta(x).$$

2. "Finding an approach which reduces the average number of descents seems to be of great practical interest."

Problems in Tsaban 2007

1. "Does there exist an efficient algorithm to find an orbit decomposition for which the maximal number of descents is as small as it can be?" That is to find a θ which minimizes

$$\max_{x \in V} D_\theta(x).$$

2. "Finding an approach which reduces the average number of descents seems to be of great practical interest." That is to find a θ which minimizes

$$\sum_{x \in V} D_\theta(x).$$

New Results

Orbit: graphical point of view

The graph of a function is a disjoint union of components of graph with one cycle.

Orbit: graphical point of view

The graph of a function is a disjoint union of components of graph with one cycle.

The orbit is a line ending at the root (we can consider a cycle as a root).

Orbit: graphical point of view

The graph of a function is a disjoint union of components of graph with one cycle.

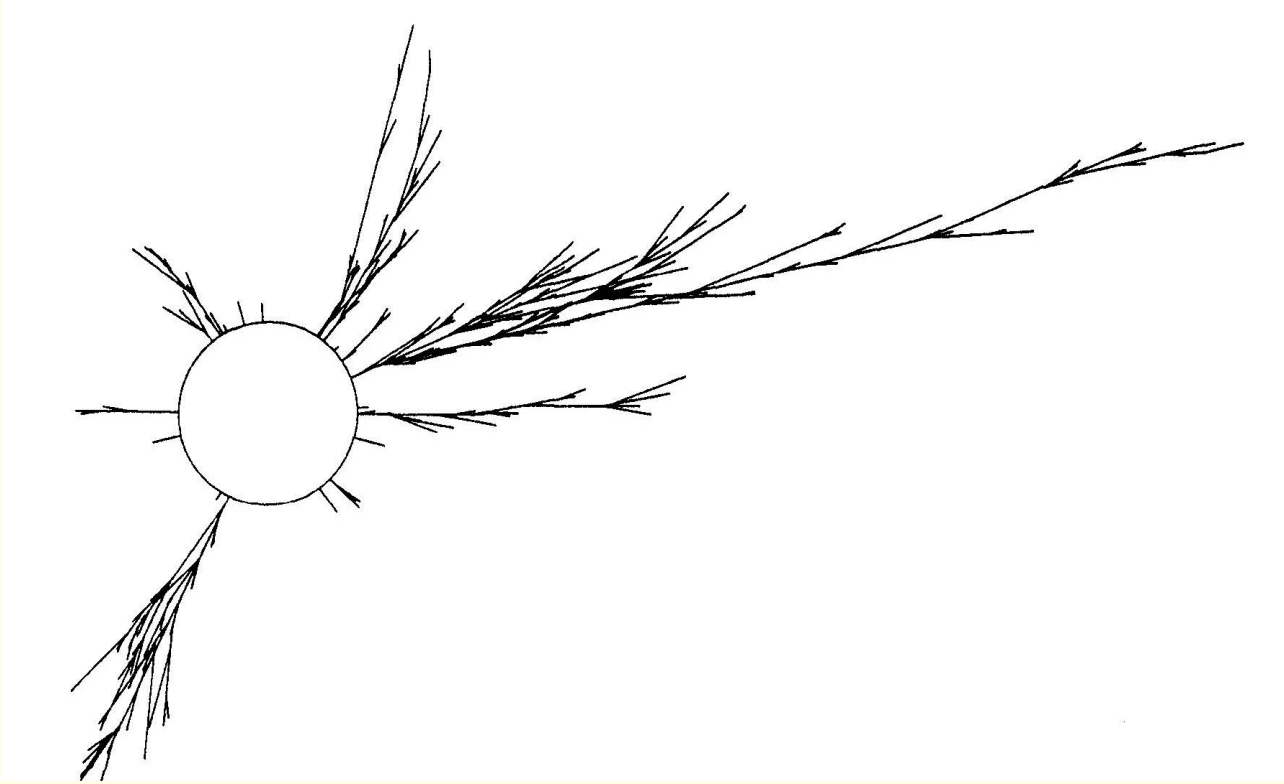
The orbit is a line ending at the root (we can consider a cycle as a root).

The first orbit in an original component ends on the circle. When we erase an orbit from a component, the remainder is a disjoint union of components of rooted tree.

Typical graph of function

a giant component

rendered by Quisquater and Delescaille 1988.



Descent: graphical point of view

Erase an orbit from each component each time.

The number of descents of x is the time right before x being erased.

Descent: graphical point of view

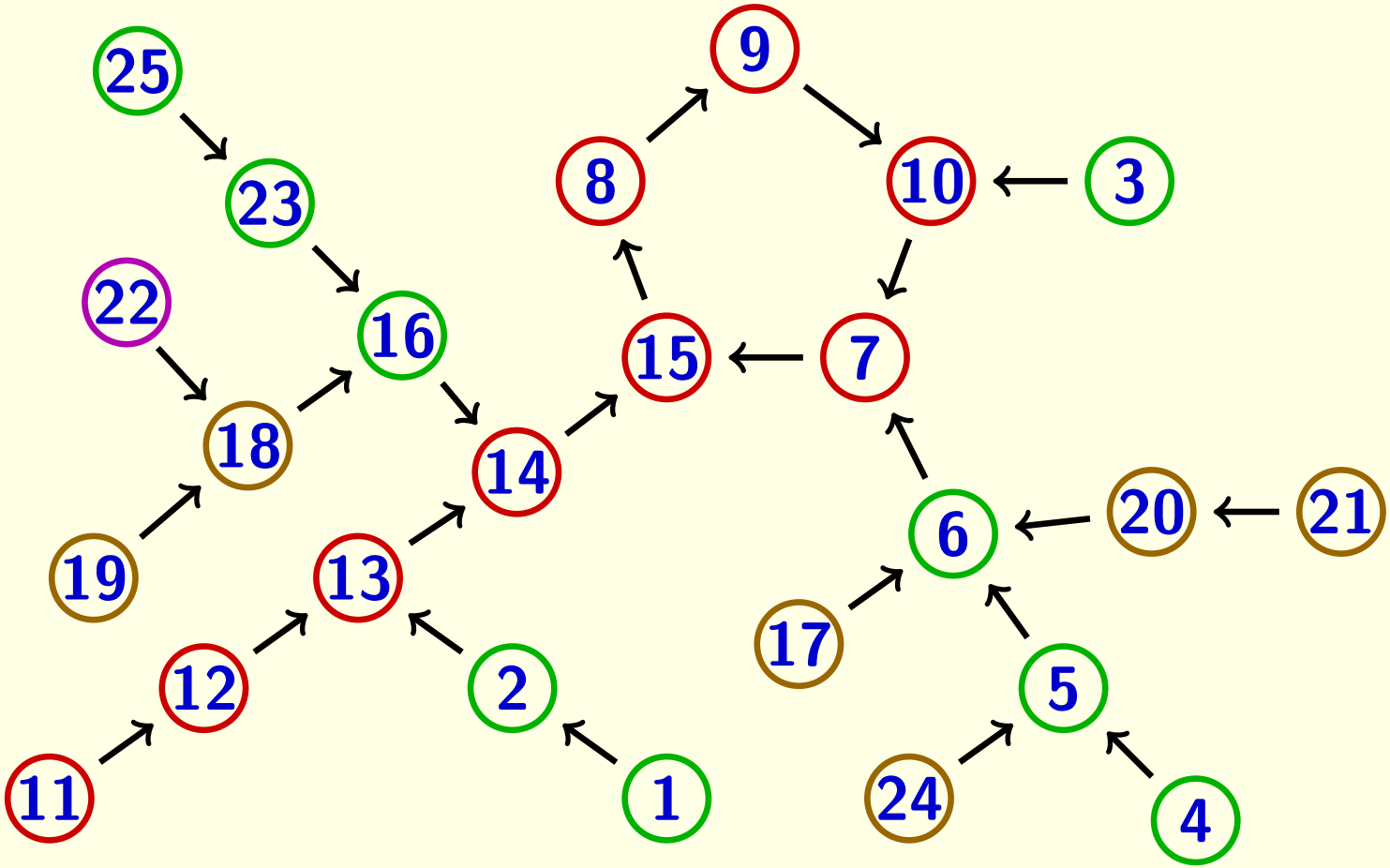
Erase an orbit from each component each time.

The number of descents of x is the time right before x being erased.

The number of descents only depends on the decomposition of a graph (independent of the order of components).

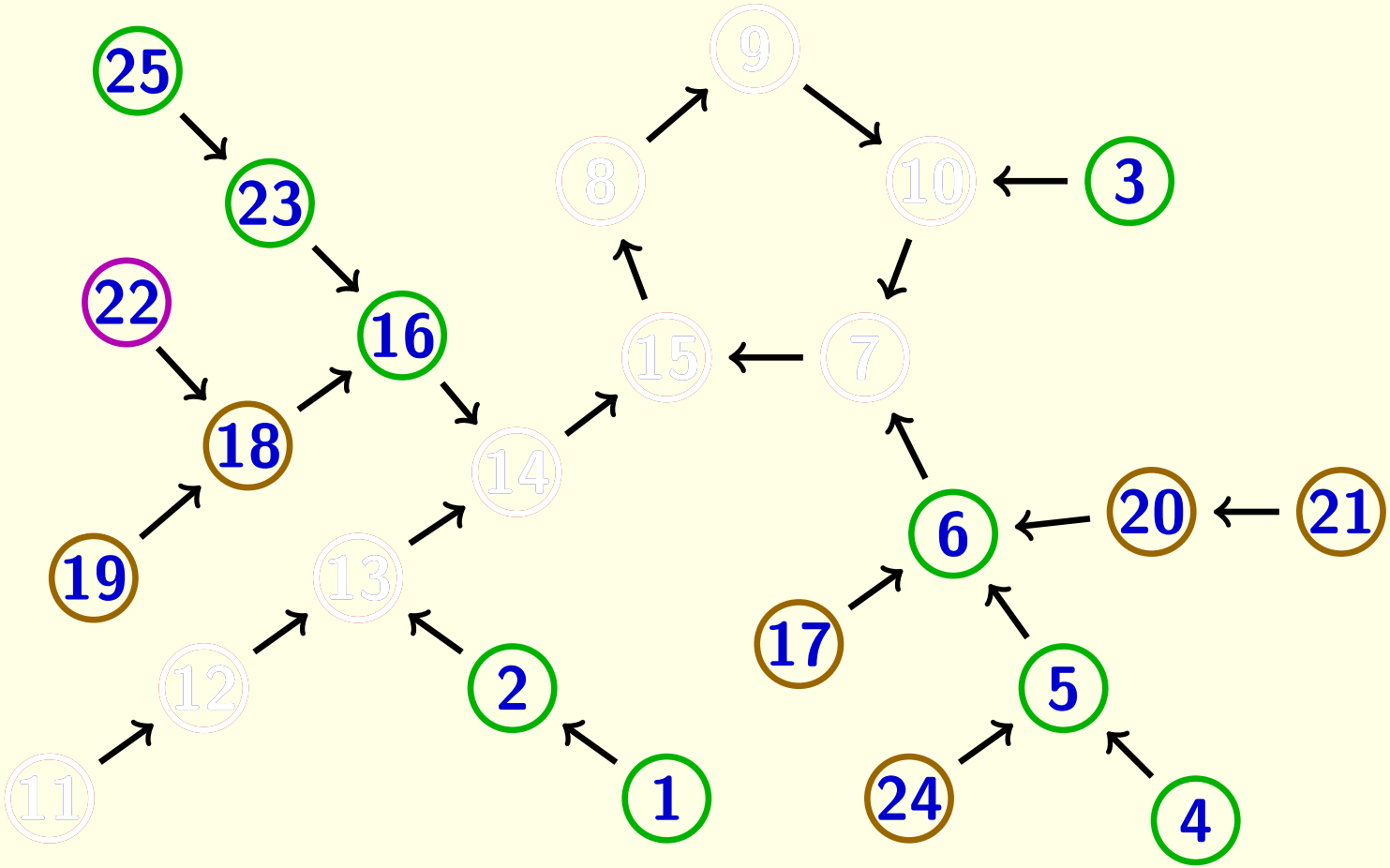
Example for numbers of descents

red: 0



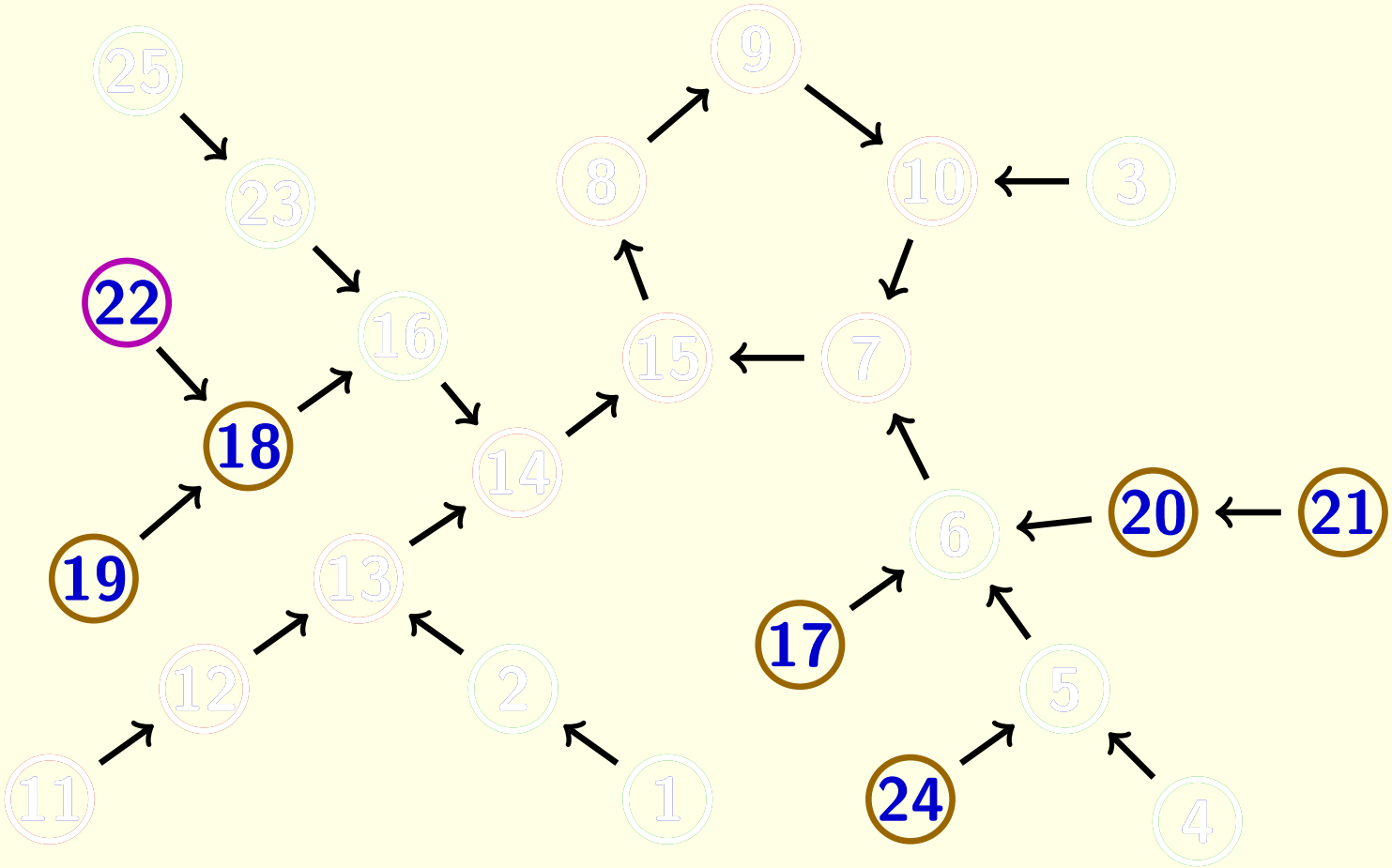
Example for numbers of descents

red: 0 **green**: 1



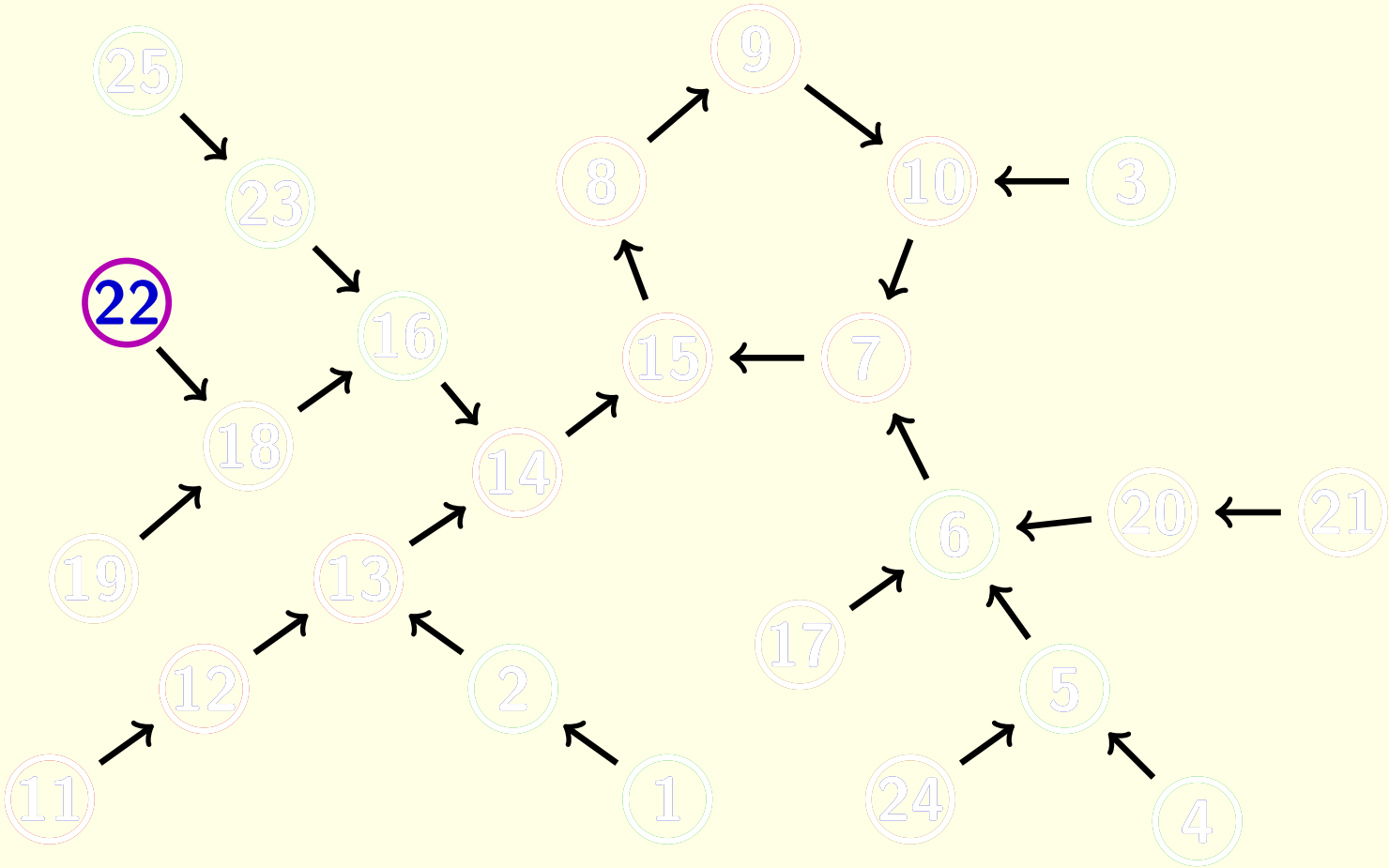
Example for numbers of descents

red: 0 **green**: 1 **brown**: 2



Example for numbers of descents

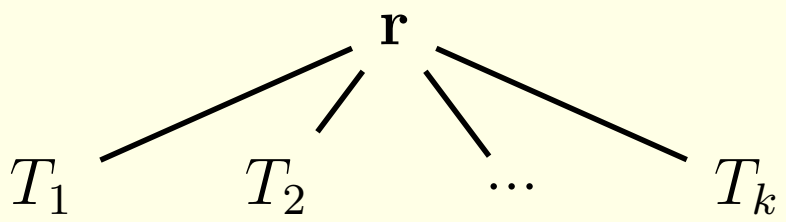
red: 0 **green: 1** **brown: 2** **purple: 3**



Now, the problem is to find a decomposition of a tree that minimizes the number of descents.

Bottom-up construction

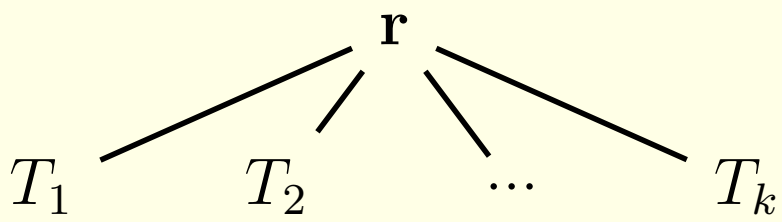
Lemma 1 **Let T be a rooted tree:**



Give each T_i a decomposition. Let $D_{old}(x)$ be the number of descents of x.

Bottom-up construction

Lemma 1 **Let T be a rooted tree:**

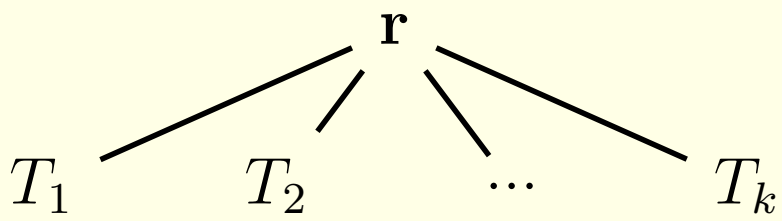


Give each T_i a decomposition. Let $D_{old}(x)$ be the number of descents of x.

Now, extend the decomposition to T: r and the root of T_a in a same orbit. Then $D_{new}(\mathrm{r}) = 0$ and

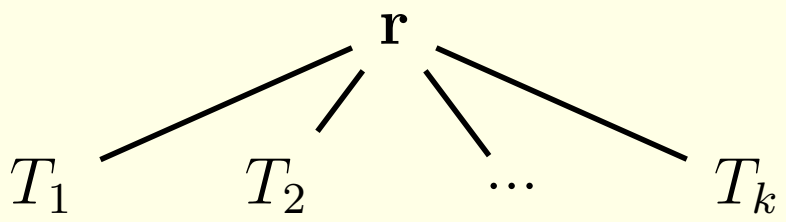
$$D_{new}(x) = \begin{cases} D_{old}(x), & \text{if } x \in T_a, \\ D_{old}(x) + 1, & \text{otherwise.} \end{cases}$$

Which one is the best connection?



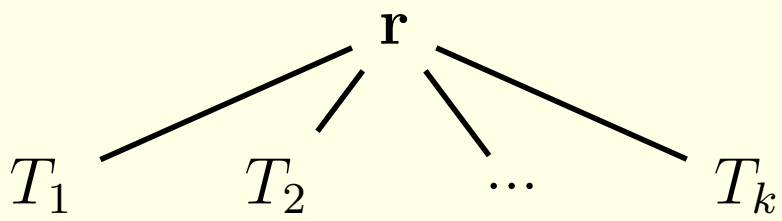
Choose the subtree which has the largest number of descents.

Which one is the best connection?



Choose the subtree which has the largest number of descents. If there are more than one subtree with the same largest number of descents, then choose anyone of them. By Lemma 1, the maximal number of descents increases 1 in this case.

Which one is the best connection?



Choose the subtree which has the largest number of descents. If there are more than one subtree with the same largest number of descents, then choose anyone of them. By Lemma 1, the maximal number of descents increases 1 in this case.

The function characterized by above rule has been studied in other areas.

The Horton-Strahler number

It's originally used to classify river system. Later also appeared in computer science as register function.

The Horton-Strahler number

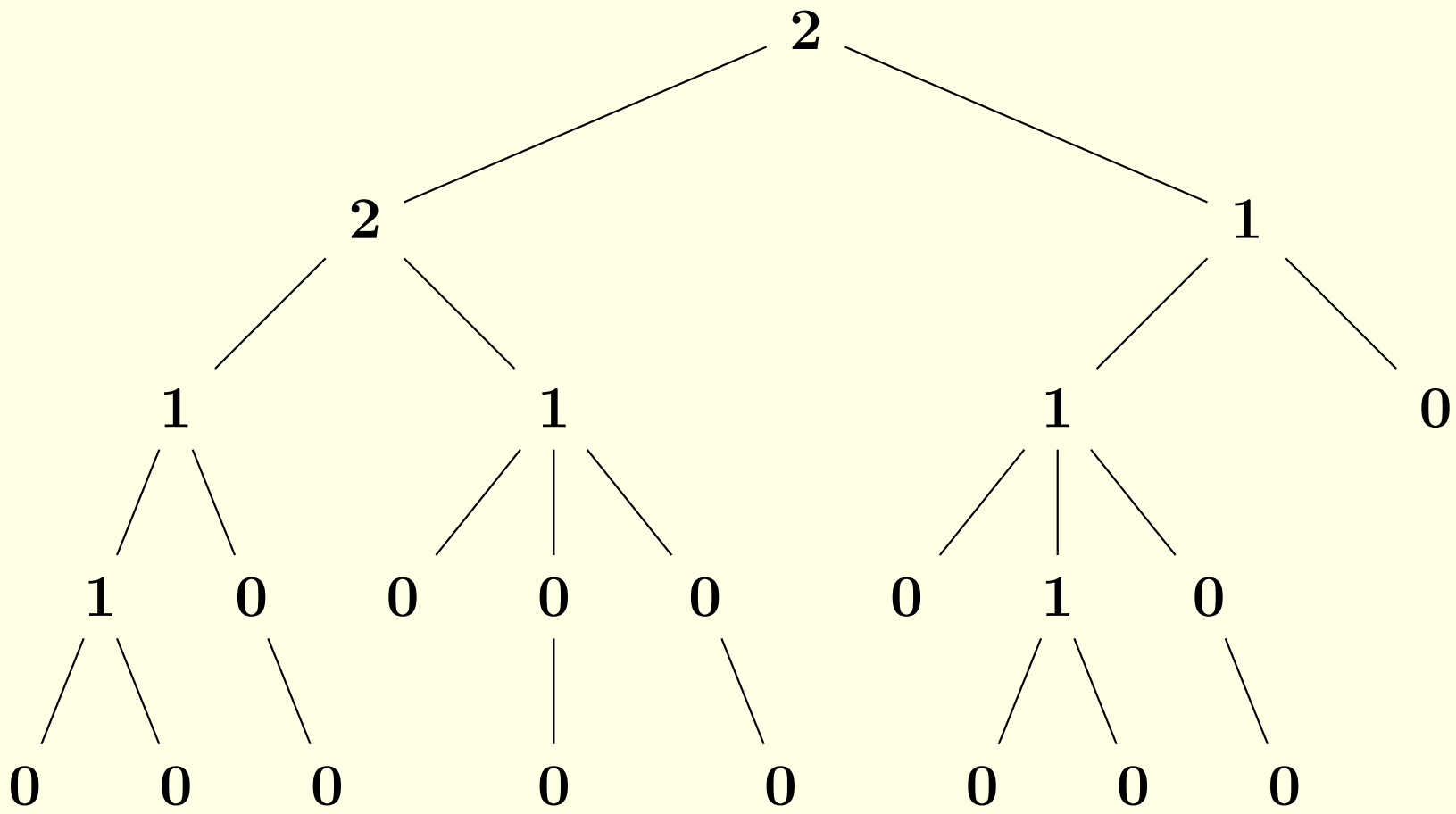
It's originally used to classify river system. Later also appeared in computer science as register function.

It was defined on binary trees. Now, we extend the definition to rooted trees with branching factor ≥ 1:

$$S(\mathbf{r}) = \begin{cases} 0 & \textbf{if } \mathrm{r} \textbf{ has no child,} \\ M(\mathbf{r}) & \textbf{if only one child with } S(\mathbf{u}) = M(\mathbf{r}), \\ M(\mathbf{r}) + 1 & \textbf{otherwise,} \end{cases}$$

where $M(\mathbf{r}) = \max\{S(\mathbf{u}) : \mathbf{u} \textbf{ is a child of } \mathrm{r}\}$.

Example of the Horton-Strahler number



The least maximal number of descents

Find a decomposition with minimal $\max_{x \in V} D_\theta(x)$**.**

The least maximal number of descents

Find a decomposition with minimal $\max_{x \in V} D_\theta(x)$**.**

Theorem 1

The optimal orbit decomposition is constructed componentwise by the orbits, a path top-down from the root pass nodes with the largest Horton-Strahler number locally.

The least maximal number of descents

Find a decomposition with minimal $\max_{x \in V} D_\theta(x)$.

Theorem 1

The optimal orbit decomposition is constructed componentwise by the orbits, a path top-down from the root pass nodes with the largest Horton-Strahler number locally.

Moreover, the least maximal number of descents is equal to the largest Horton-Strahler number in the tree.

Horton-Strahler number for binary tree

Let T be a binary tree with N nodes. Let S_N be the Horton-Strahler number of the root of T.

Horton-Strahler number for binary tree

Let T be a binary tree with N nodes. Let S_N be the Horton-Strahler number of the root of T.

Flajolet et al. (1979), Kemp (1979) and Meir et al. (1980) independently proved that

$$\mathbb{E}S_N = \log_4 N + O(1).$$

Horton-Strahler number for binary tree

Let T be a binary tree with N nodes. Let S_N be the Horton-Strahler number of the root of T.

Flajolet et al. (1979), Kemp (1979) and Meir et al. (1980) independently proved that

$$\mathbb{E}S_N = \log_4 N + O(1).$$

Later, Prodinger (1987) and Devroye et al. (1995) derived more results.

Horton-Strahler number for binary tree

Let T be a binary tree with N nodes. Let S_N be the Horton-Strahler number of the root of T.

Flajolet et al. (1979), Kemp (1979) and Meir et al. (1980) independently proved that

$$\mathbb{E}S_N = \log_4 N + O(1).$$

Later, Prodinger (1987) and Devroye et al. (1995) derived more results.

Recently, Auber et al. (2004) studied a different extension.

The worst case

– the tree with the largest least maximal number of descents

The worst case

– the tree with the largest least maximal number of descents

The complete binary tree is the worst case.

The worst case

– the tree with the largest least maximal number of descents

The complete binary tree is the worst case.

Let d be the number of levels. Then the maximal number of descents is $d-1$ and the total number of nodes is $2^d - 1 = N$.

Thus, generally, the least maximal number of descents is bounded by $\log_2(N+1) - 1$.

To minimize average number of descents

– **find a decomposition with minimal** $\sum_{x \in V} D_\theta(x)$**.**

To minimize average number of descents

– find a decomposition with minimal $\sum_{x \in V} D_\theta(x)$.

Which one is the best child to be connected with?

To minimize average number of descents

– find a decomposition with minimal $\sum_{x \in V} D_\theta(x)$.

Which one is the best child to be connected with?

The one with the most offsprings!

To minimize average number of descents

– find a decomposition with minimal $\sum_{x \in V} D_\theta(x)$.

Which one is the best child to be connected with?

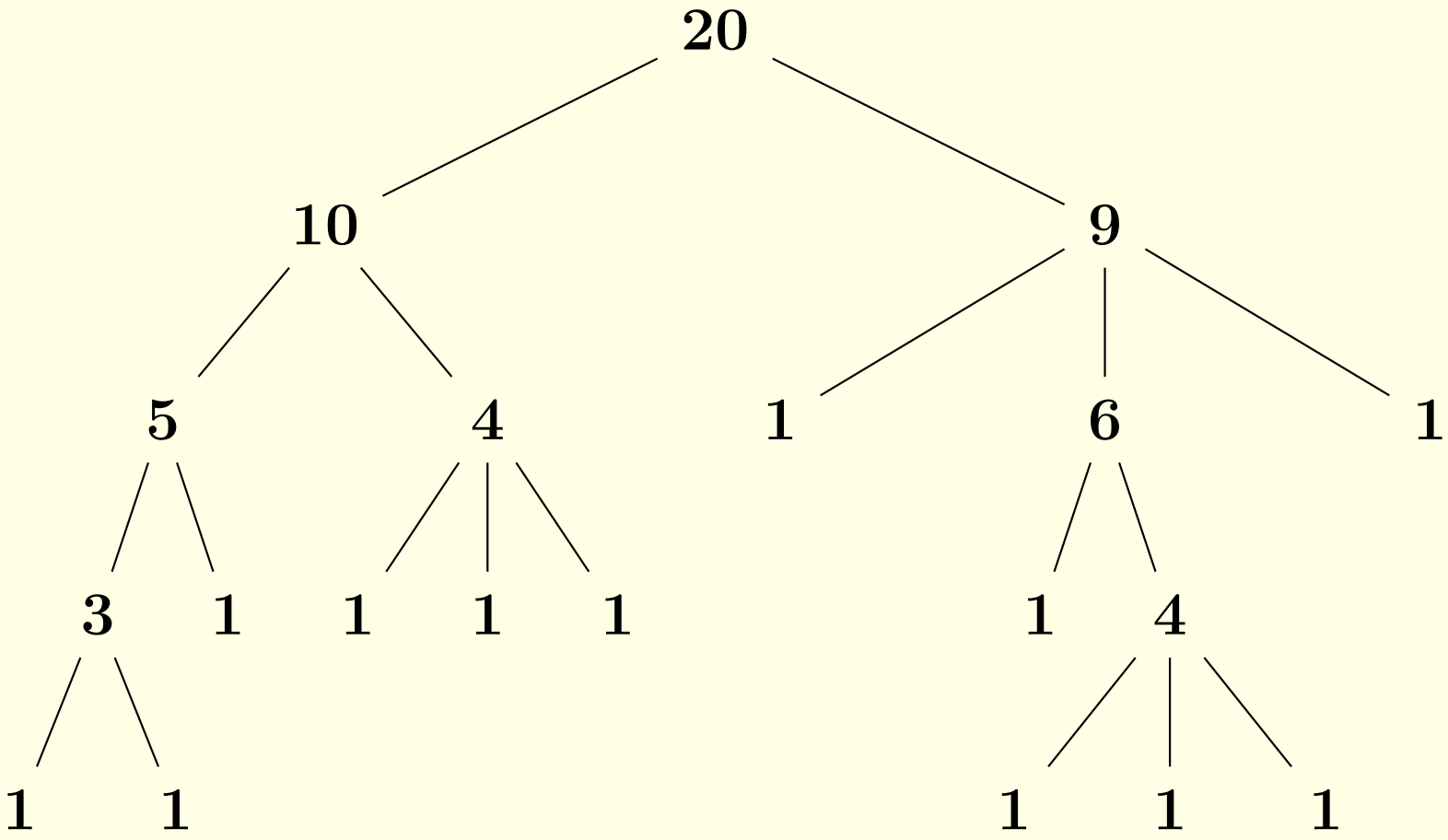
The one with the most offsprings!

Orbit decomposition:

Compute for each node the number of subtree size.

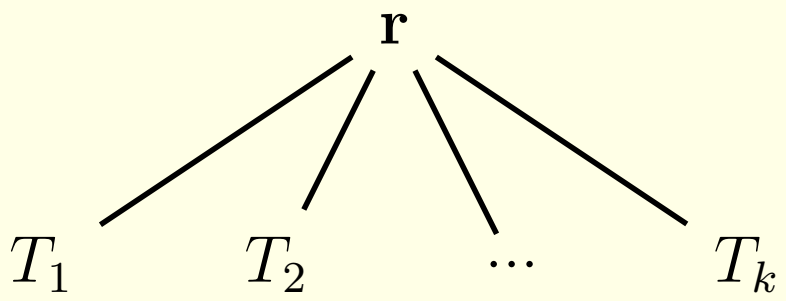
For each component, starting from the root we choose the path of the orbit through the nodes with the largest number of subtree size.

Example **The number is the size of the subtree rooted at the node.**



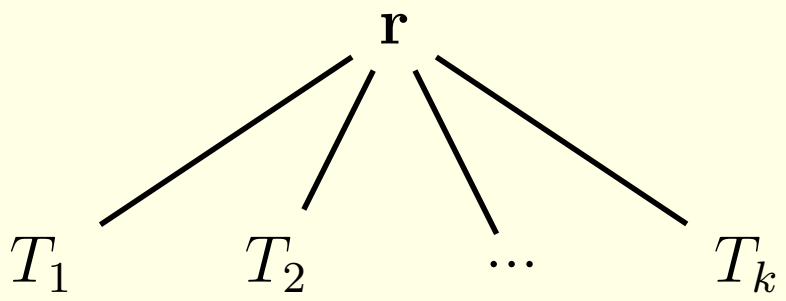
Compare to greedy orbit decomposition

The main difference is the rule to connect the nodes and their children:



Compare to greedy orbit decomposition

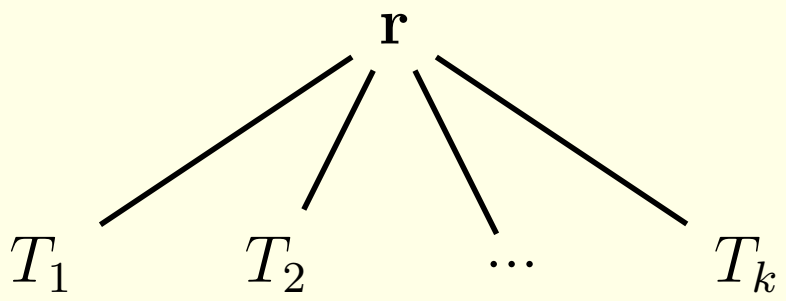
The main difference is the rule to connect the nodes and their children:



greedy the tallest subtree (with most levels)

Compare to greedy orbit decomposition

The main difference is the rule to connect the nodes and their children:

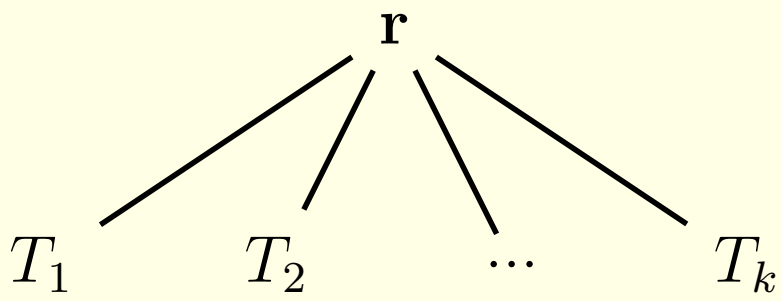


greedy the tallest subtree (with most levels)

least max the subtree with largest H-S number

Compare to greedy orbit decomposition

The main difference is the rule to connect the nodes and their children:



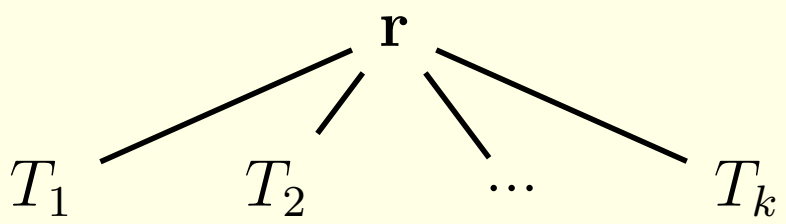
greedy the tallest subtree (with most levels)

least max the subtree with largest H-S number

least avg the heaviest subtree (with most nodes)

Theorem 2

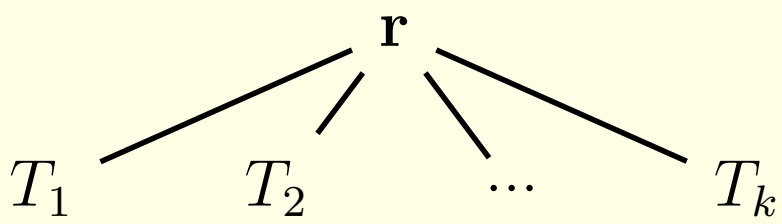
Let T be a rooted tree with N nodes:



Let m_i be the total number of descents in T_i and n_i be the total number of nodes in T_i.

Theorem 2

Let T be a rooted tree with N nodes:



Let m_i be the total number of descents in T_i and n_i be the total number of nodes in T_i.

Then the best choice is the T_i with the maximal n_i and the total number of descents of T is

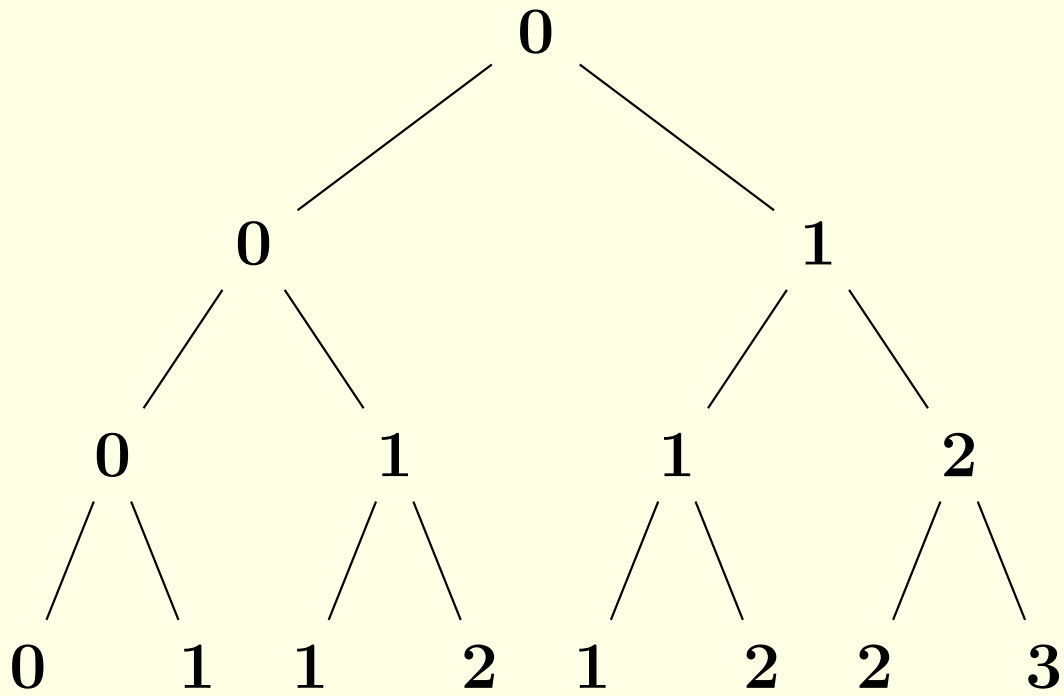
$$\sum_{i=1}^{k} m_i + (N - 1 - \max\{n_1, ..., n_k\}) .$$

The worst case

The worst case is also complete binary tree.

The worst case

The worst case is also complete binary tree.

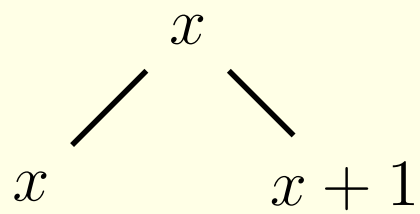


Here the number indicates the number of descents.

Average number of descents of complete binary tree

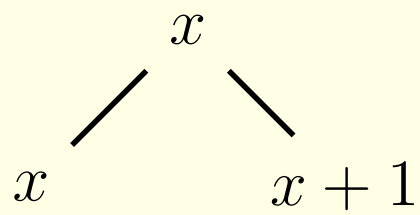
Let c_k be the total number of descents at level k.

Then $c_1 = 0$ and $c_k = 2c_{k-1} + 2^{k-2}$ for $k \geq 2$. (Why?)



Average number of descents of complete binary tree

Let c_k be the total number of descents at level k. Then $c_1 = 0$ and $c_k = 2c_{k-1} + 2^{k-2}$ for $k \geq 2$. (Why?)



Thus $c_k = (k-1)2^{k-2}$ and the average number descents in this case is

$$\frac{\sum_{k=1}^{d}(k-1)2^{k-2}}{\sum_{k=1}^{d}2^{k-1}} = \frac{d}{2} - 1 + \frac{d}{2^d - 1},$$

where d is the number of levels.

Conclusion

1. **For any function f on $\{0, 1, ..., N-1\}$, the iteration $f^m(x)$ for each m and x can be computed in time $O(\log N)$ by using preprocessed information. The preprocess need linear space and linear time.**

Conclusion

1. **For any function f on $\{0, 1, ..., N-1\}$, the iteration $f^m(x)$ for each m and x can be computed in time $O(\log N)$ by using preprocessed information. The preprocess need linear space and linear time.**
2. **Open problems: analysis of the Horton-Strahler number and the average number of descents for random functions.**

Conclusion

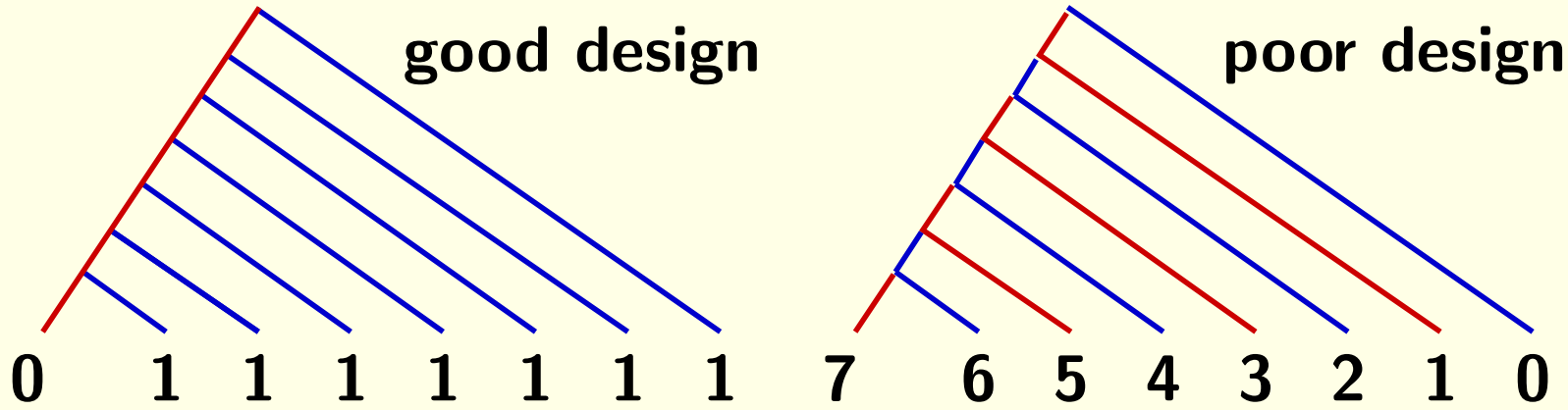
1. **For any function f on $\{0, 1, ..., N-1\}$, the iteration $f^m(x)$ for each m and x can be computed in time $O(\log N)$ by using preprocessed information. The preprocess need linear space and linear time.**
2. **Open problems: analysis of the Horton-Strahler number and the average number of descents for random functions.**
3. **Application? Maybe use in the construction of pseudo-random functions.**

Conclusion

1. **For any function f on $\{0, 1, ..., N-1\}$, the iteration $f^m(x)$ for each m and x can be computed in time $O(\log N)$ by using preprocessed information. The preprocess need linear space and linear time.**
2. **Open problems: analysis of the Horton-Strahler number and the average number of descents for random functions.**
3. **Application? Maybe use in the construction of pseudo-random functions.**
4. **Another interpretation of our work. (next page)**

Another interpretation of our work

Transportation system



Consider orbit as bus line and descent as transfer.

Our work is to design a bus system that minimizes the number of transfers.

Thank You!