

Shannon Coding for the Discrete Noiseless Channel and Related Problems

Man DU

Mordecai GOLIN

Qin ZHANG

HKUST

AAAC 2008
April 27, 2008

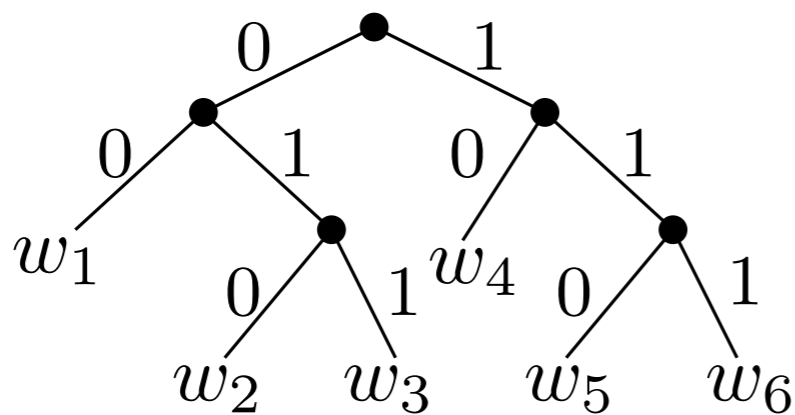


Outline

- ▣ Problems
- ▣ Results
- ▣ The Algorithm
- ▣ A case where the algorithm fails

Prefix-free coding

- Let $\Sigma = \{\sigma_1, \sigma_2, \dots, \sigma_r\}$ be an *encoding alphabet*. Word $w \in \Sigma^*$ is a *prefix* of word $w' \in \Sigma^*$ if $w' = wu$ where $u \in \Sigma^*$ is a non-empty word. A *Code* over Σ is a collection of words $C = \{w_1, \dots, w_n\}$.
- Code C is *prefix-free* if for all $i \neq j$ w_i is not a prefix of w_j .
 $\{0, 10, 11\}$ is prefix-free. $\{0, 00, 11\}$ isn't.
- A prefix-free code can be modelled as (leaves of) a tree



$$\begin{array}{ll} w_1 = 00 & w_4 = 10 \\ w_2 = 010 & w_5 = 110 \\ w_3 = 011 & w_6 = 111 \end{array}$$



The prefix-free coding problem

- Let $cost(w)$ be the *length* or number of characters in w . Let $P = \{p_1, p_2, \dots, p_n\}$ be a fixed discrete probability distribution (P.D.).

Define $cost(C) = \sum_{i=1}^n cost(w_i)p_i$

The prefix-free coding problem

- Let $cost(w)$ be the *length* or number of characters in w . Let $P = \{p_1, p_2, \dots, p_n\}$ be a fixed discrete probability distribution (P.D.).

$$\text{Define } cost(C) = \sum_{i=1}^n cost(w_i)p_i$$

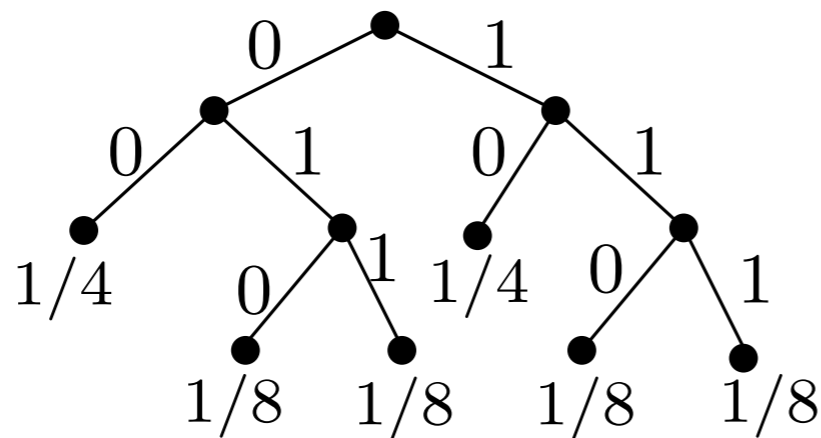
- The *prefix coding* problem, sometimes known as the *Huffman encoding* problem is to find a prefix-free code over Σ of minimum cost.

The prefix-free coding problem

- Let $cost(w)$ be the *length* or number of characters in w . Let $P = \{p_1, p_2, \dots, p_n\}$ be a fixed discrete probability distribution (P.D.).

$$\text{Define } cost(C) = \sum_{i=1}^n cost(w_i)p_i$$

- The *prefix coding* problem, sometimes known as the *Huffman encoding* problem is to find a prefix-free code over Σ of minimum cost.



Equivalent to finding tree with *minimum external path-length*

Generalizations

- Unequal-cost coding

Allow letters to have different costs, say, $c(\sigma_j) = c_j$.

- Discrete Noiseless Channels (in Shannon's original paper)

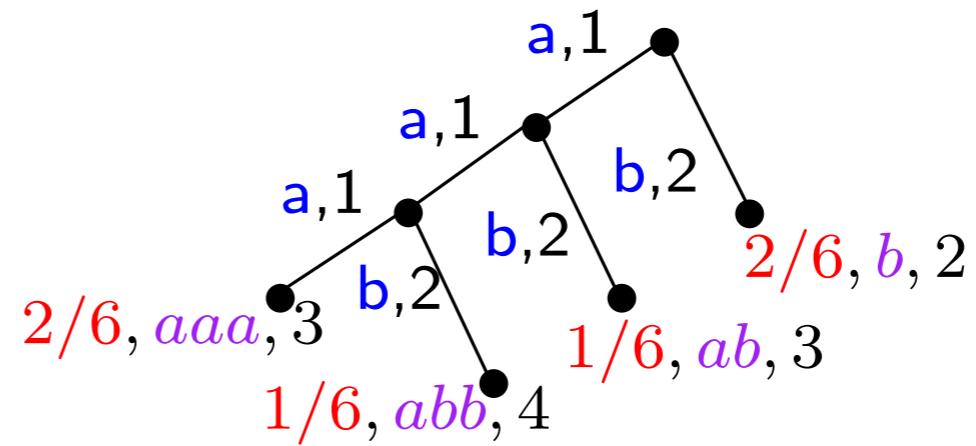
This can be viewed as a **directed graph** with k vertices (states).

1. Each edge leaving a vertex is labelled by an encoding letter $\sigma \in \Sigma$, with at most one σ -edge leaving each vertex.
2. An edge labelled by σ leaving vertex i has cost $c_{i,\sigma}$.

Generalizations: Prefix-free coding

- With Unequal-cost letters

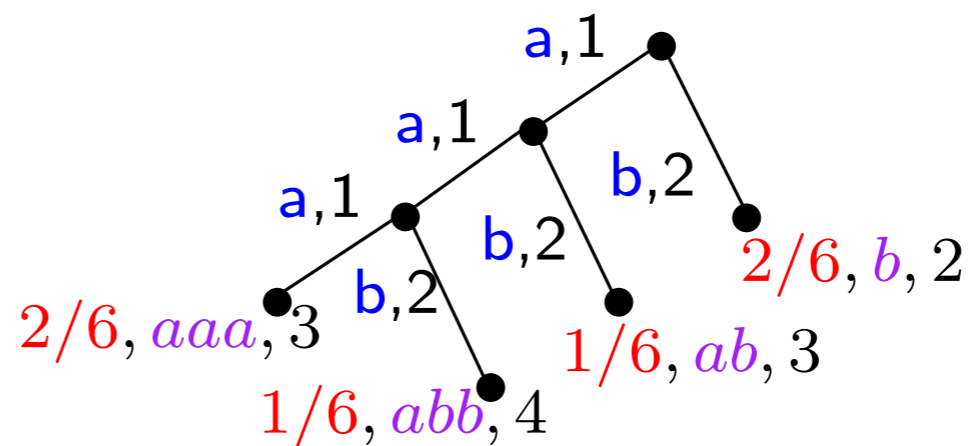
$$c_1 = 1; c_2 = 2.$$



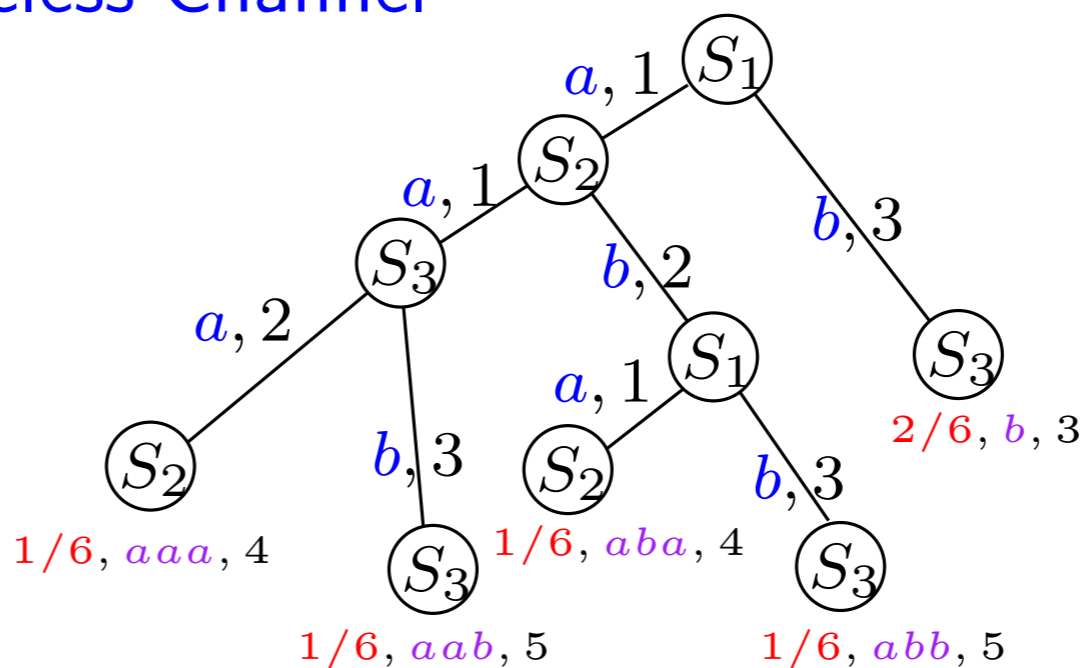
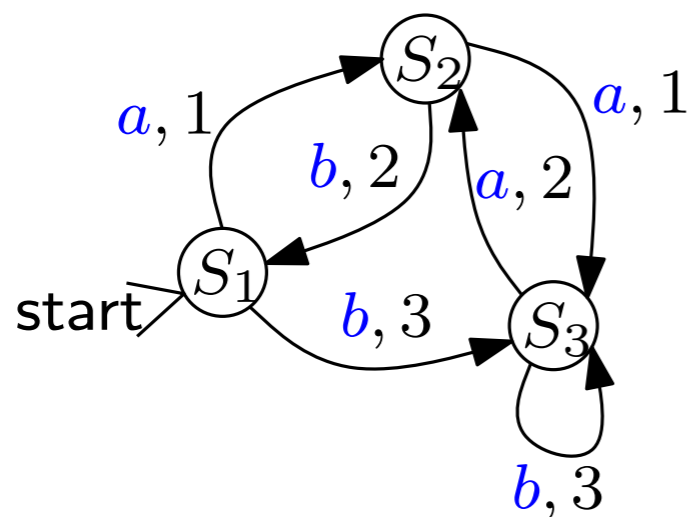
Generalizations: Prefix-free coding

- With Unequal-cost letters

$$c_1 = 1; c_2 = 2.$$



- In a Discrete Noiseless Channel





Outline

- ▣ Problems
- ▣ Results
- ▣ Proofs and the Algorithm
- ▣ An exceptional case

Some known results:

- Optimal codes

Huffman 1952: a well-known $O(rn \log n)$ -time greedy algorithm ($O(rn)$ -time if the p_i are sorted in non-decreasing order)

Some known results:

- Optimal codes

Huffman 1952: a well-known $O(rn \log n)$ -time greedy algorithm ($O(rn)$ -time if the p_i are sorted in non-decreasing order)

- Sub-optimal codes

Shannon coding: (from noiseless coding theorem)
There exists a prefix-free code with word lengths $l_i = \lceil -\log_r p_i \rceil$, $i = 1, 2, \dots, n$.

Shannon-Fano coding: probability splitting
Try to put $\sim \frac{1}{r}$ of the probability in each node.

Some known results:

- Optimal codes

Huffman 1952: a well-known $O(rn \log n)$ -time greedy algorithm ($O(rn)$ -time if the p_i are sorted in non-decreasing order)

- Sub-optimal codes

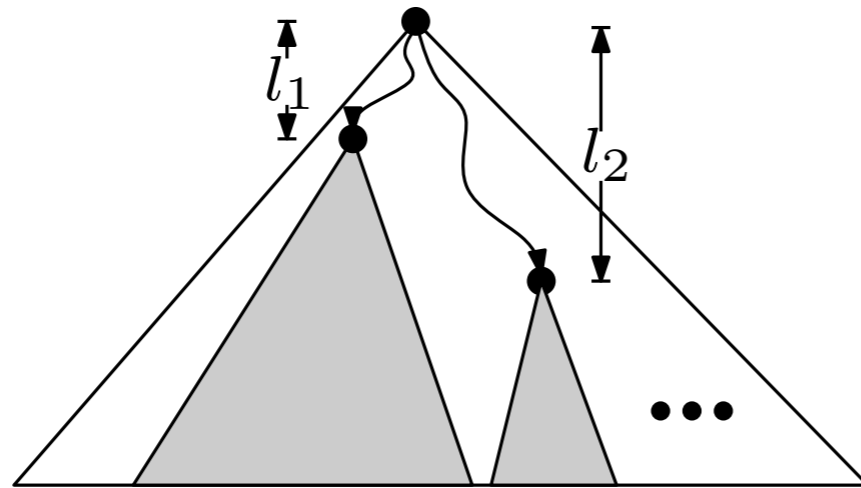
Shannon coding: (from noiseless coding theorem)
There exists a prefix-free code with word lengths $l_i = \lceil -\log_r p_i \rceil, i = 1, 2, \dots, n.$

Shannon-Fano coding: probability splitting
Try to put $\sim \frac{1}{r}$ of the probability in each node.

Both methods have cost within 1 of optimal

Some known results. Cont.

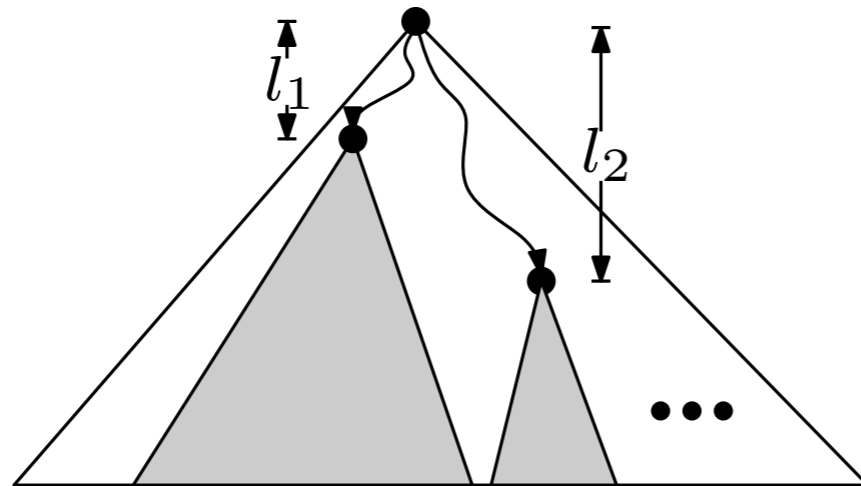
▣ Shannon Coding



$$l_i = \lceil -\log_r p_i \rceil$$

Some known results. Cont.

□ Shannon Coding

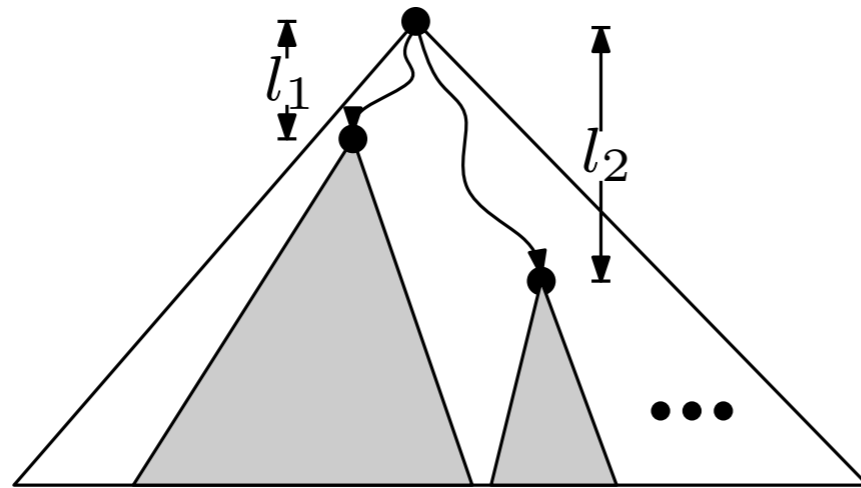


$$l_i = \lceil -\log_r p_i \rceil$$

Grey triangle rooted at node is part of tree that can't be used when node is leaf.

Some known results. Cont.

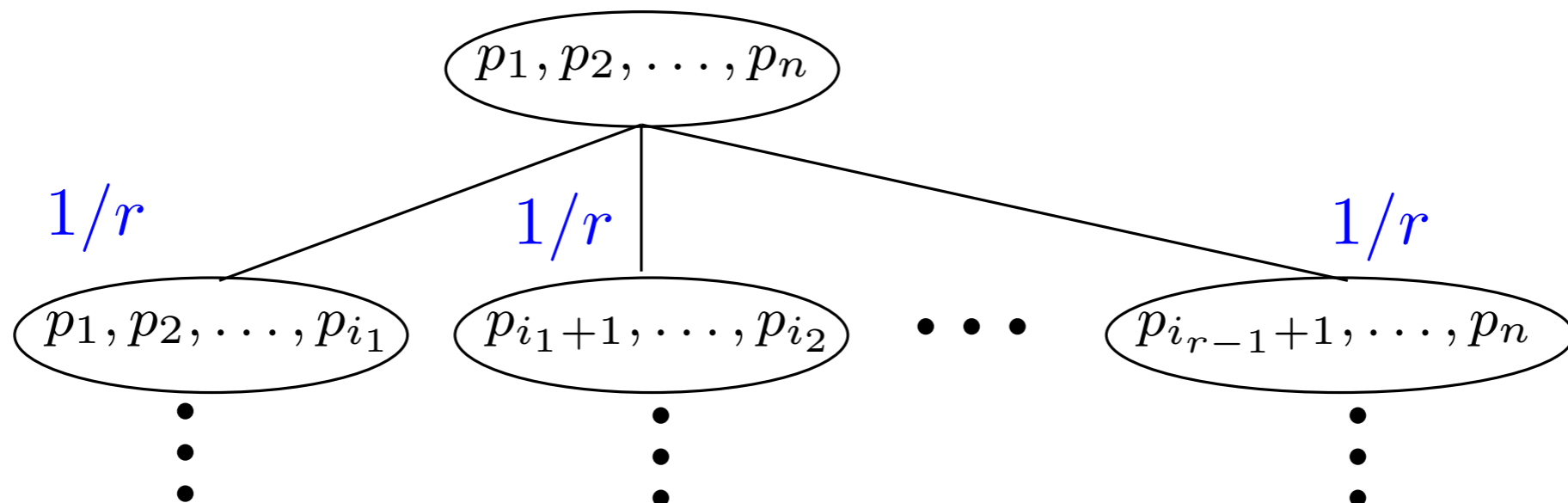
Shannon Coding



$$l_i = \lceil -\log_r p_i \rceil$$

Grey triangle rooted at node is part of tree that can't be used when node is leaf.

Shannon-Fano Coding



Some known results. Unequal Cost Coding

- Letters in Σ have **different** costs c_1, c_2, c_3, \dots
Models different transmission/storage costs
- There are exponential (in the c_i) algorithms and a PTAS.
Still **don't know** if it's NP-Hard, in P or something between.
- Efficient algorithms ($O(n \log n)$ or $O(n)$) that create codes which are **within an additive error of optimal**.

$$COST \leq OPT + K$$

Some known results. Unequal Cost Coding

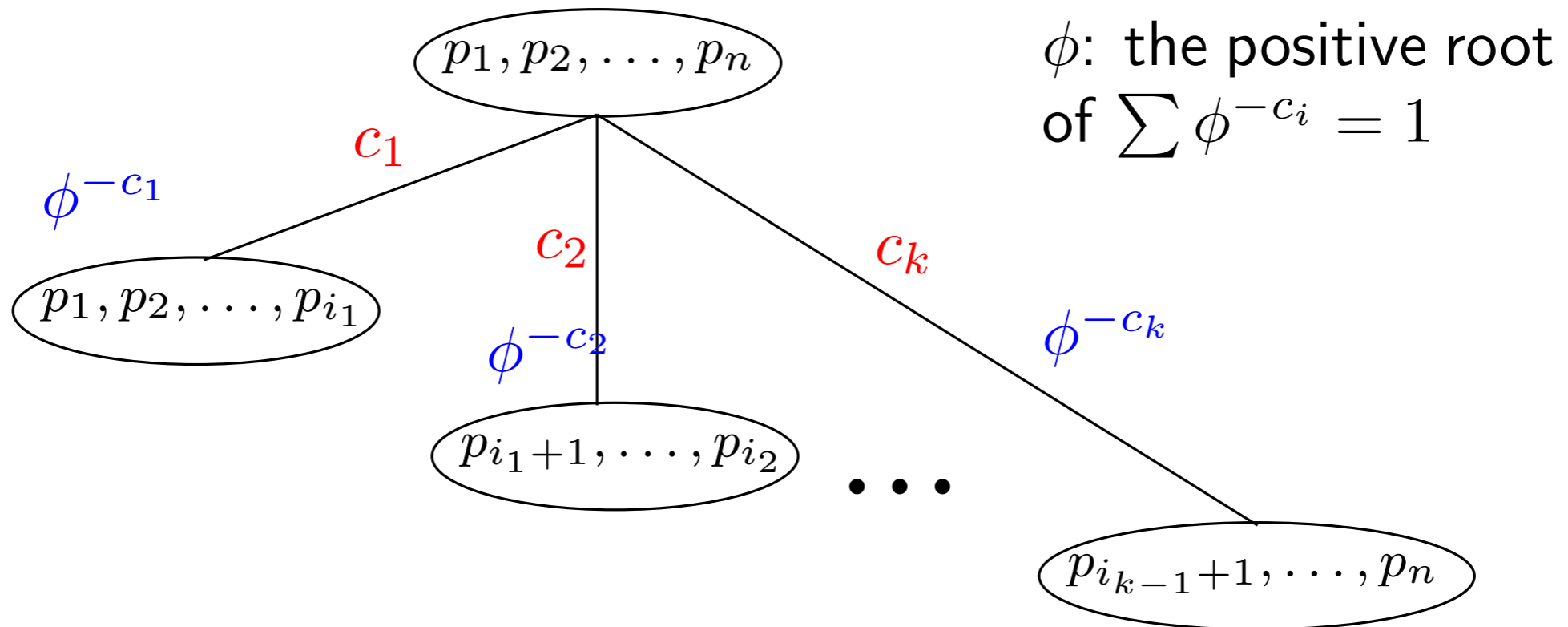
- Letters in Σ have **different** costs c_1, c_2, c_3, \dots
Models different transmission/storage costs
- There are exponential (in the c_i) algorithms and a PTAS.
Still **don't know** if it's NP-Hard, in P or something between.
- Efficient algorithms ($O(n \log n)$ or $O(n)$) that create codes which are **within an additive error of optimal**.

$$COST \leq OPT + K$$

- All of these approximation algorithms are **variations of Shannon Fano coding**; perform probability splitting on nodes of the tree!

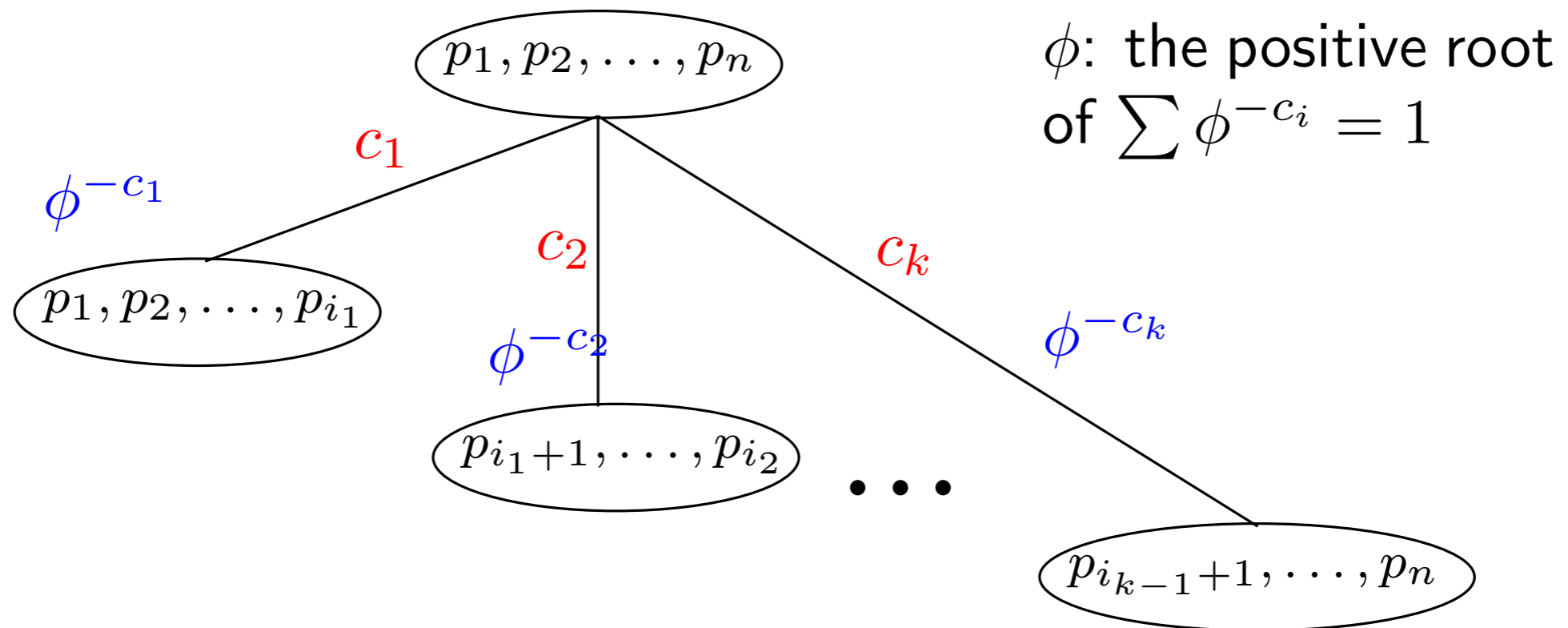
Some known results. Cont.

- Shannon Fano coding for unequal cost codes



Some known results. Cont.

- Shannon Fano coding for unequal cost codes



- Split probabilities so “approximately” ϕ^{-c_i} of the probability in a node is put into its i^{th} child.



Some Shannon-Fano type codes with $COST \leq OPT + K$

- Krause (1962)
- Csiszar (1969)
- Cott (1977)
- Alenkamp and Mehlhorn (1980)
- Mehlhorn (1980)
- Golin and Li (2007)



Some Shannon-Fano type codes with $COST \leq OPT + K$

- Krause (1962)
- Csiszar (1969)
- Cott (1977)
- Altenkamp and Mehlhorn (1980)
- Mehlhorn (1980)
- Golin and Li (2007)

Algorithms differ in how they define the “approximate” split.



Some Shannon-Fano type codes with $COST \leq OPT + K$

- Krause (1962)
- Csiszar (1969)
- Cott (1977)
- Alenkamp and Mehlhorn (1980)
- Mehlhorn (1980)
- Golin and Li (2007)

Algorithms differ in how they define the “approximate” split.

K is a function of letter costs c_1, c_2, c_3, \dots



Some Shannon-Fano type codes with $COST \leq OPT + K$

- Krause (1962)
- Csiszar (1969)
- Cott (1977)
- Alenkamp and Mehlhorn (1980)
- Mehlhorn (1980)
- Golin and Li (2007)

Algorithms differ in how they define the “approximate” split.

K is a function of letter costs c_1, c_2, c_3, \dots

$K(c_1, c_2, c_3, \dots)$ are incomparable between different algorithms

Known Lower Bound:

- Given coding letter lengths $\mathcal{C} = \{c_1, c_2, c_3, \dots\}$, $\gcd(c_i) = 1$, let α be the unique positive root of $g(z) = 1 - \sum_j z^{c_j}$ and $\phi = \alpha^{-1}$;

Known Lower Bound:

- Given coding letter lengths $\mathcal{C} = \{c_1, c_2, c_3, \dots\}$, $\gcd(c_i) = 1$, let α be the unique positive root of $g(z) = 1 - \sum_j z^{c_j}$ and $\phi = \alpha^{-1}$;

Note: ϕ sometimes called the “capacity”

Known Lower Bound:

- Given coding letter lengths $\mathcal{C} = \{c_1, c_2, c_3, \dots\}$, $\gcd(c_i) = 1$, let α be the unique positive root of $g(z) = 1 - \sum_j z^{c_j}$ and $\phi = \alpha^{-1}$;
Note: ϕ sometimes called the “capacity”
- For given P.D. set $H_\phi = -\sum p_i \log_\phi p_i$.

Known Lower Bound:

- Given coding letter lengths $\mathcal{C} = \{c_1, c_2, c_3, \dots\}$, $\gcd(c_i) = 1$, let α be the unique positive root of $g(z) = 1 - \sum_j z^{c_j}$ and $\phi = \alpha^{-1}$;

Note: ϕ sometimes called the “capacity”

- For given P.D. set $H_\phi = -\sum p_i \log_\phi p_i$.

Note: If $c_1 = c_2 = 1$ then $\phi = 2$ and H_ϕ is standard entropy

Known Lower Bound:

- Given coding letter lengths $\mathcal{C} = \{c_1, c_2, c_3, \dots\}$, $\gcd(c_i) = 1$, let α be the unique positive root of $g(z) = 1 - \sum_j z^{c_j}$ and $\phi = \alpha^{-1}$;

Note: ϕ sometimes called the “capacity”

- For given P.D. set $H_\phi = -\sum p_i \log_\phi p_i$.

Note: If $c_1 = c_2 = 1$ then $\phi = 2$ and H_ϕ is standard entropy

- Theorem:

Let OPT be cost of min-cost code for given P.D. and letter costs. Then

$$H_\phi \leq OPT$$

Known Lower Bound:

- Given coding letter lengths $\mathcal{C} = \{c_1, c_2, c_3, \dots\}$, $\gcd(c_i) = 1$, let α be the unique positive root of $g(z) = 1 - \sum_j z^{c_j}$ and $\phi = \alpha^{-1}$;

Note: ϕ sometimes called the “capacity”

- For given P.D. set $H_\phi = -\sum p_i \log_\phi p_i$.

Note: If $c_1 = c_2 = 1$ then $\phi = 2$ and H_ϕ is standard entropy

- Theorem:

Let OPT be cost of min-cost code for given P.D. and letter costs. Then

$$H_\phi \leq OPT$$

Note: If $c_1 = c_2 = 1$ then $\phi = 2$ and this is classic “Shannon Information theoretic Lower Bound”



Our results (Main)

Shannon-Fano splitting is not necessary;
Shannon-coding suffices!

Our results (Main)

Shannon-Fano splitting is not necessary;
Shannon-coding suffices!

- Given coding letter lengths \mathcal{C} , let ϕ be as defined on previous page.

Then $\exists K > 0$ depending only upon \mathcal{C} such that if

1. $P = \{p_1, p_2, \dots, p_n\}$ is any P.D., and
2. l_1, l_2, \dots, l_n any set of integers such that
 $\forall i, l_i \geq K + \lceil -\log_{\phi} p_i \rceil,$

then there exists a prefix free code for which the l_i are the word lengths.

Our results (Main)

Shannon-Fano splitting is not necessary;
Shannon-coding suffices!

- Given coding letter lengths \mathcal{C} , let ϕ be as defined on previous page.

Then $\exists K > 0$ depending only upon \mathcal{C} such that if

1. $P = \{p_1, p_2, \dots, p_n\}$ is any P.D., and
2. l_1, l_2, \dots, l_n any set of integers such that $\forall i, l_i \geq K + \lceil -\log_{\phi} p_i \rceil$,

then there exists a prefix free code for which the l_i are the word lengths.

$$\Rightarrow \sum_i p_i l_i \leq K + 1 + H_{\phi}(P) \leq OPT + K + 1$$

- This gives an additive approximation of same type as Shannon-Fano splitting without the splitting (same time complexity but many fewer operations on reals).



Our results (Main)

- Our method works also for
 1. The Discrete Noiseless Channel and
 2. Regular Languages

Our results (Main)

- Our method works also for
 1. The Discrete Noiseless Channel and
 2. Regular Languages
- The Discrete Noiseless Channel: Only previous result seems to be [Csiszar \(1969\)](#) who gives additive approximation to optimal code using a generalization of Shannon-Fano splitting.



Outline

- ▣ Problems
- ▣ Results
- ▣ Proofs and the Algorithm
- ▣ An exceptional case

Proof of the Main Theorem (for unequal case)

- We first prove the following lemma.

Given \mathcal{C} and corresponding ϕ then

$\exists \beta > 0$ depending only upon \mathcal{C} such that if

$$\sum_{i=1}^n \phi^{-l_i} \leq \beta,$$

then there exists a prefix-free code with word lengths l_1, l_2, \dots, l_n .

Proof of the Main Theorem (for unequal case)

- We first prove the following lemma.

Given \mathcal{C} and corresponding ϕ then

$\exists \beta > 0$ depending only upon \mathcal{C} such that if

$$\sum_{i=1}^n \phi^{-l_i} \leq \beta,$$

then there exists a prefix-free code with word lengths l_1, l_2, \dots, l_n .

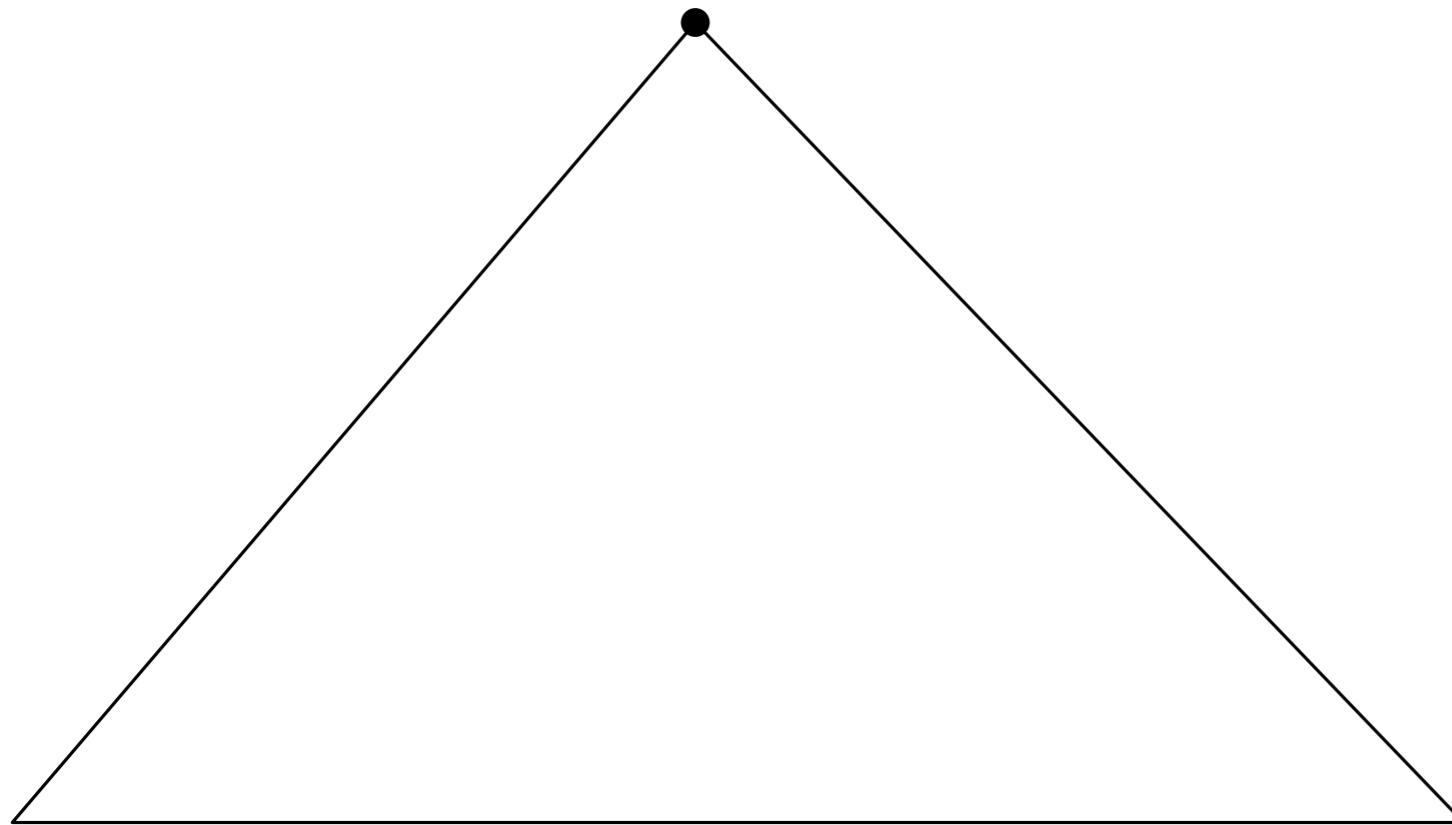
- *Note: if $c_1 = c_2 = 1$ then $\phi = 2$. Let $\beta = 1$ and condition becomes $\sum 2^{-l_i} \leq 1$.*

Lemma then becomes one direction of **Kraft Inequality**.

Proof of the lemma

- Let $L(n)$ be the number of nodes on level n of the infinite tree corresponding to \mathcal{C}

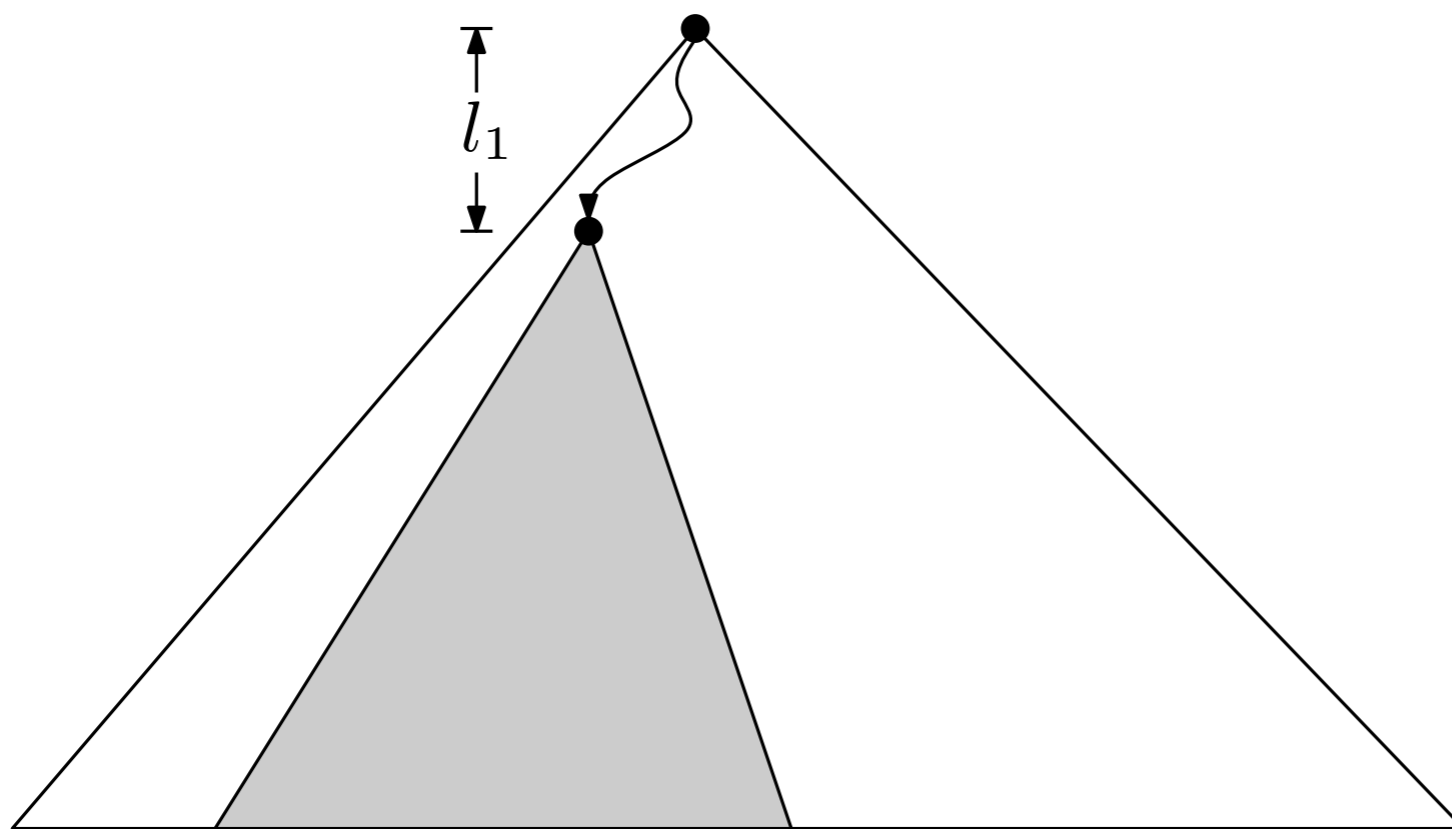
Can show $\exists t_1, t_2$ s.t., $t_1\phi^n \leq L(n) \leq t_2\phi^n$.



Proof of the lemma

- Let $L(n)$ be the number of nodes on level n of the infinite tree corresponding to \mathcal{C}

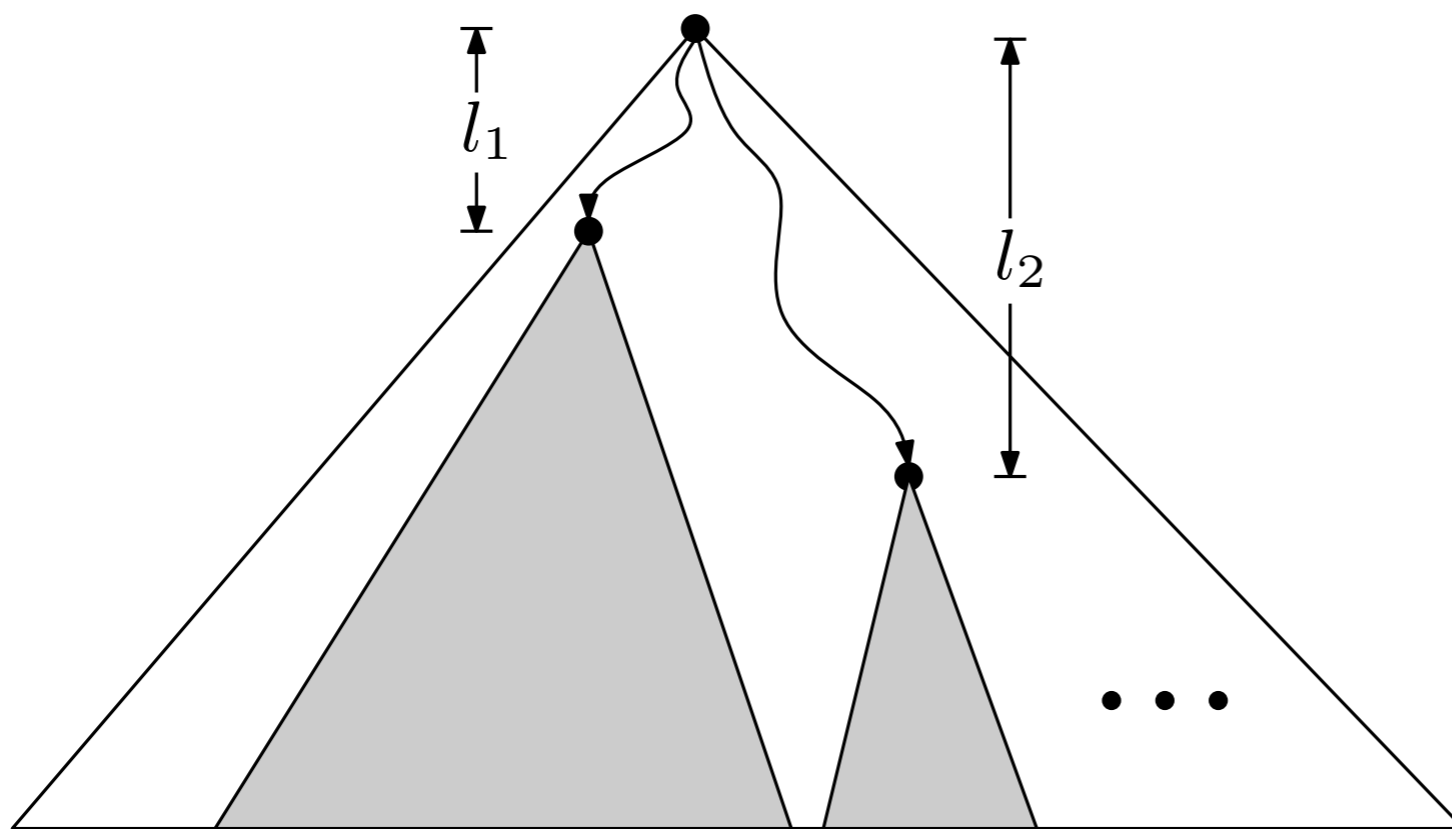
Can show $\exists t_1, t_2$ s.t., $t_1\phi^n \leq L(n) \leq t_2\phi^n$.



Proof of the lemma

- Let $L(n)$ be the number of nodes on level n of the infinite tree corresponding to \mathcal{C}

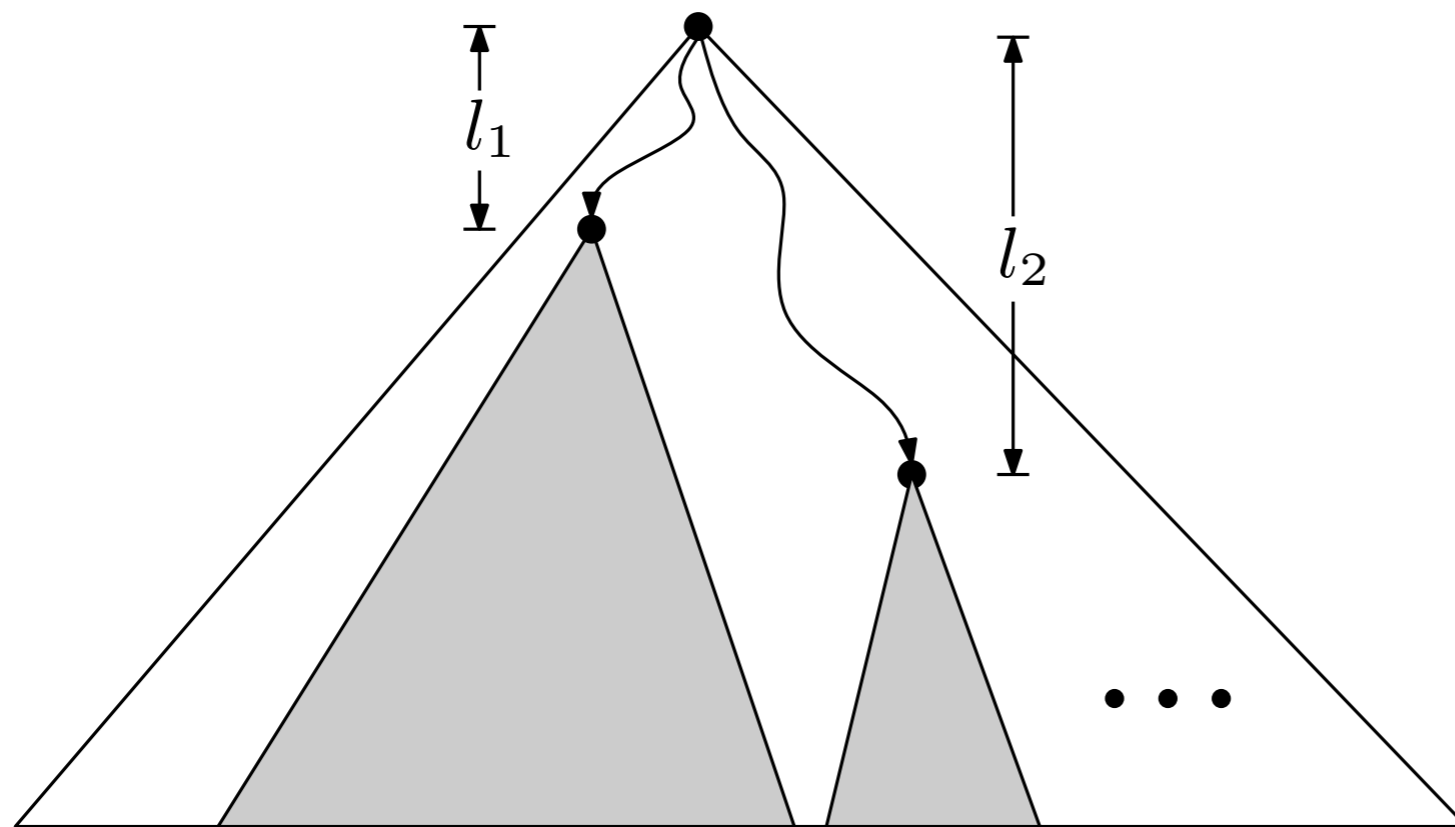
Can show $\exists t_1, t_2$ s.t., $t_1\phi^n \leq L(n) \leq t_2\phi^n$.



Proof of the lemma

- Let $L(n)$ be the number of nodes on level n of the infinite tree corresponding to \mathcal{C}

Can show $\exists t_1, t_2$ s.t., $t_1\phi^n \leq L(n) \leq t_2\phi^n$.

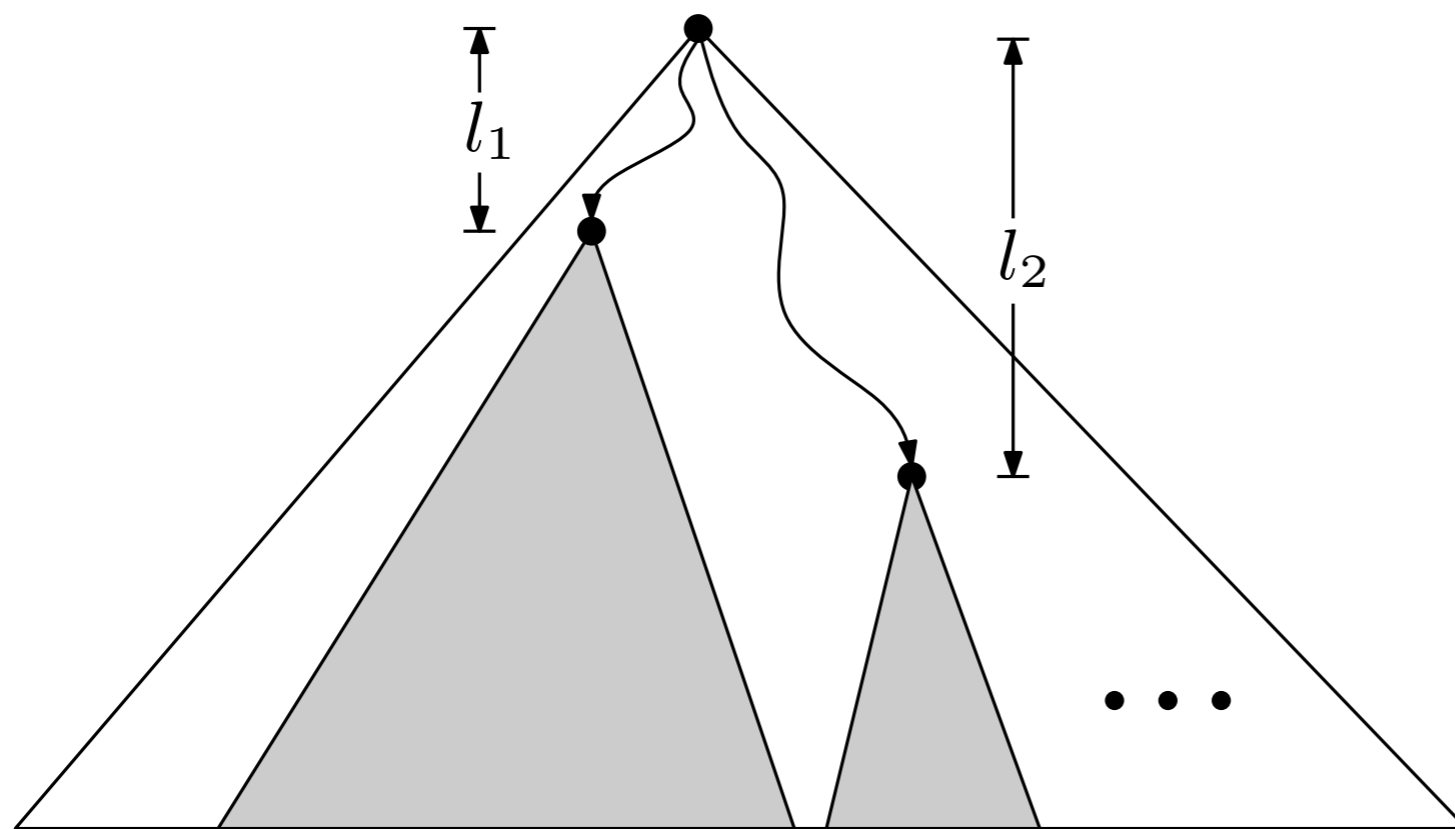


Grey regions are parts of infinite tree that are erased when node i on l_i becomes leaf.

Proof of the lemma

- Let $L(n)$ be the number of nodes on level n of the infinite tree corresponding to \mathcal{C}

Can show $\exists t_1, t_2$ s.t., $t_1\phi^n \leq L(n) \leq t_2\phi^n$.



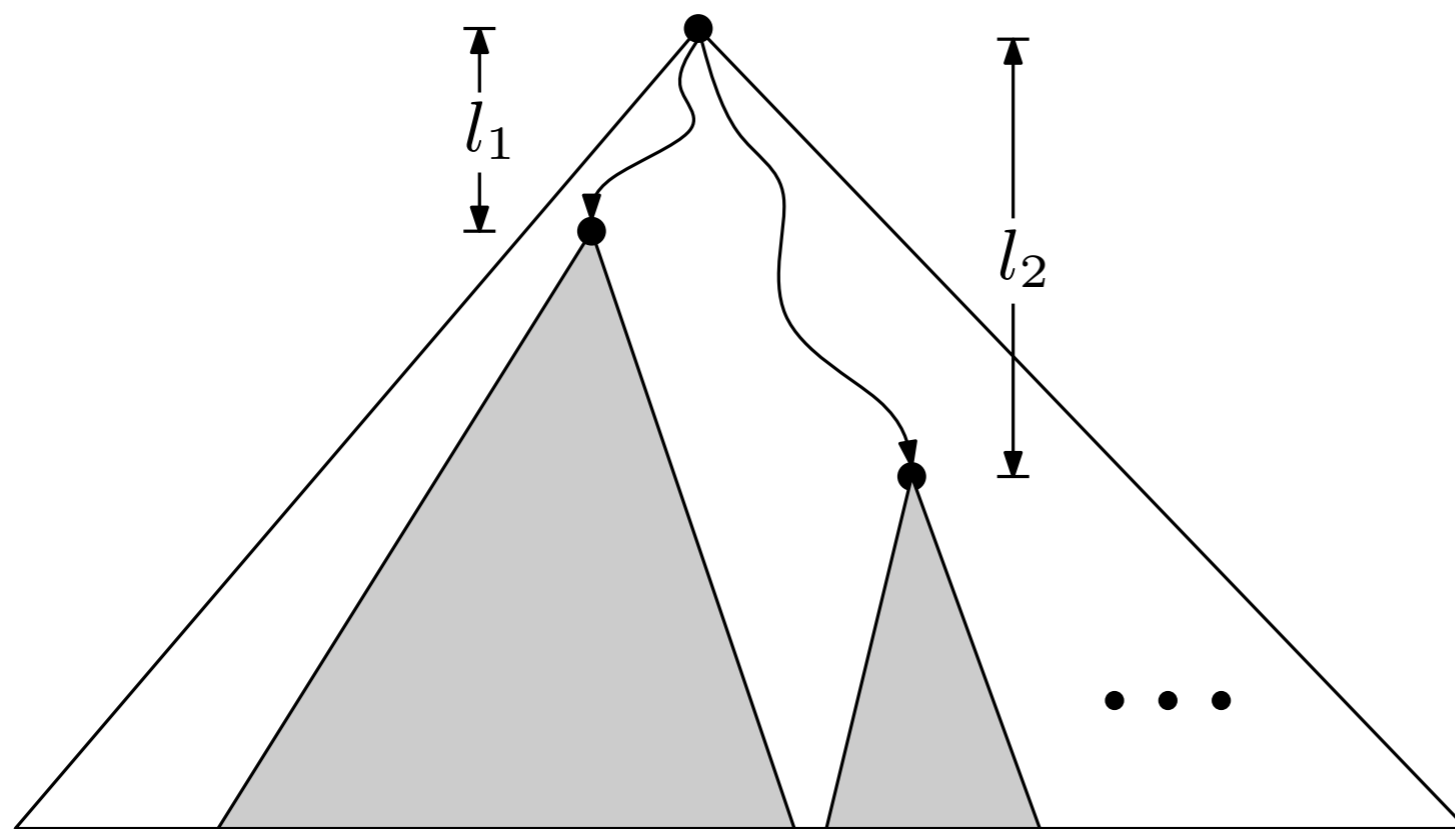
Grey regions are parts of infinite tree that are erased when node i on l_i becomes leaf.

Node on l_k has $L(l_i - l_k)$ descendants on l_i

Proof of the lemma

- Let $L(n)$ be the number of nodes on level n of the infinite tree corresponding to \mathcal{C}

Can show $\exists t_1, t_2$ s.t., $t_1\phi^n \leq L(n) \leq t_2\phi^n$.



Grey regions are parts of infinite tree that are erased when node i on l_i becomes leaf.

Node on l_k has $L(l_i - l_k)$ descendants on l_i

Node on l_i can become leaf iff grey regions do not cover all nodes on level l_i

Proof of the lemma

□ Just need to show that $0 < L(\ell_i) - \sum_{k=1}^{i-1} L(\ell - \ell_k)$.

□

$$\begin{aligned} L(\ell_i) - \sum_{k=1}^{i-1} L(\ell - \ell_k) &\geq t_1 \phi^\ell - t_2 \sum_{k=1}^{i-1} \phi^{\ell - \ell_k} \\ &\geq \phi^\ell \left(t_1 - t_2 \sum_{k=1}^{i-1} \phi^{-\ell_k} \right) \\ &\geq \phi^\ell (t_1 - t_2 \beta) \end{aligned}$$

□ Choose $\beta < \frac{t_1}{t_2}$

Proof of the Main Theorem

- Set $K = -\log_{\phi} \beta$. (Recall $l_i = K + \lceil -\log_{\phi} p_i \rceil$)
Then

$$\begin{aligned} \sum_{i=1}^n \phi^{-l_i} &\leq \sum_{i=1}^n \phi^{-K - \lceil -\log_{\phi} p_i \rceil} \\ &\leq \beta \sum_{i=1}^n \phi^{\log_{\phi} p_i} = \beta \sum_{i=1}^n p_i = \beta \end{aligned}$$

Proof of the Main Theorem

- Set $K = -\log_{\phi} \beta$. (Recall $l_i = K + \lceil -\log_{\phi} p_i \rceil$)
Then

$$\begin{aligned} \sum_{i=1}^n \phi^{-l_i} &\leq \sum_{i=1}^n \phi^{-K - \lceil -\log_{\phi} p_i \rceil} \\ &\leq \beta \sum_{i=1}^n \phi^{\log_{\phi} p_i} = \beta \sum_{i=1}^n p_i = \beta \end{aligned}$$

- From previous lemma, a prefix free code with word lengths l_1, l_2, \dots, l_n exists.



The Algorithm

- A valid K could be found by working through the proof of Theorem. Technically, $O(1)$ but, practically, this would require some complicated operations on reals.

The Algorithm

- A valid K could be found by working through the proof of Theorem. Technically, $O(1)$ but, practically, this would require some complicated operations on reals.
- Alternatively, perform **doubling search for K** , the smallest K for which theorem is valid.

Set $\bar{K} = 1, 2, 2^2, 2^3 \dots$

Test if $l_i = \bar{K} + \lceil -\log_{\phi} p_i \rceil$ has valid code (can be done efficiently)

until \bar{K} is good but $\bar{K}/2$ is not.

The Algorithm

- A valid K could be found by working through the proof of Theorem. Technically, $O(1)$ but, practically, this would require some complicated operations on reals.
- Alternatively, perform **doubling search for K** , the smallest K for which theorem is valid.

Set $\bar{K} = 1, 2, 2^2, 2^3 \dots$

Test if $l_i = \bar{K} + \lceil -\log_{\phi} p_i \rceil$ has valid code (can be done efficiently)

until \bar{K} is good but $\bar{K}/2$ is not.

Note that $\bar{K}/2 < K \leq \bar{K}$

The Algorithm

- A valid K could be found by working through the proof of Theorem. Technically, $O(1)$ but, practically, this would require some complicated operations on reals.
- Alternatively, perform **doubling search for K** , the smallest K for which theorem is valid.

Set $\bar{K} = 1, 2, 2^2, 2^3 \dots$

Test if $\ell_i = \bar{K} + \lceil -\log_{\phi} p_i \rceil$ has valid code (can be done efficiently)

until \bar{K} is good but $\bar{K}/2$ is not.

Note that $\bar{K}/2 < K \leq \bar{K}$

Now set $a = \bar{K}/2, b = \bar{K}$, and binary search for K in $[a, b]$.

The Algorithm

- A valid K could be found by working through the proof of Theorem. Technically, $O(1)$ but, practically, this would require some complicated operations on reals.
- Alternatively, perform **doubling search for K** , the smallest K for which theorem is valid.

Set $\bar{K} = 1, 2, 2^2, 2^3 \dots$

Test if $l_i = \bar{K} + \lceil -\log_{\phi} p_i \rceil$ has valid code (can be done efficiently)
until \bar{K} is good but $\bar{K}/2$ is not.

Note that $\bar{K}/2 < K \leq \bar{K}$

Now set $a = \bar{K}/2, b = \bar{K}$, and binary search for K in $[a,b]$.

Subtle point: Search will find $K' \leq K$ for which code exists.

The Algorithm

- ▣ A valid K could be found by working through the proof of Theorem. Technically, $O(1)$ but, practically, this would require some complicated operations on reals.
- ▣ Alternatively, perform **doubling search for K** , the smallest K for which theorem is valid.

Set $\bar{K} = 1, 2, 2^2, 2^3 \dots$

Test if $l_i = \bar{K} + \lceil -\log_{\phi} p_i \rceil$ has valid code (can be done efficiently)

until \bar{K} is good but $\bar{K}/2$ is not.

Note that $\bar{K}/2 < K \leq \bar{K}$

Now set $a = \bar{K}/2, b = \bar{K}$, and binary search for K in $[a,b]$.

Subtle point: Search will find $K' \leq K$ for which code exists.

- ▣ **Time complexity $O(n \cdot \log K)$.**



Outline

- ▣ Problems
- ▣ Results
- ▣ The Algorithm
- ▣ A case where the algorithm fails

A case the algorithm fails

- A simple example

Let \mathcal{C} be the countably infinite set defined by

$$|\{j \mid c_j = i\}| = 2C_{i-1}$$

where $C_i = \frac{1}{i+1} \binom{2i}{i}$ is the i -th Catalan number.

Constructing prefix-free codes with these \mathcal{C} can be shown to be **equivalent** to constructing **balanced binary prefix-free codes** in which, for every word, the number of **'0'**s equals the number of **'1'**s.

A case the algorithm fails

- Reason

The characteristic equation is

$$g(z) = 1 - \sum_j z^{c_j} = 1 - \sum_i 2C_{i-1}z^i = \sqrt{1-4z}$$

for which **root does not exist** ($z = 1/4$ is an algebraic singularity).

- Can prove that for $\forall \psi, K$, we can always find p_1, p_2, \dots, p_n s.t. there is **no** prefix code with length

$$l_i = K + \lceil \log_{\psi} p_i \rceil$$



The End

THANK YOU

Q and A