データを圧縮する
大量のデータを小さく収納するには？

国立情報学研究所
定兼　邦彦
2011年11月2日

データ圧縮とは

- データとは
  - 数値の集まり　　1, 2.5, 100, 3.14, …
  - 意味のあるデータが「情報」
  - 文字，画像，音声，動画など
  - コンピュータ中では，全てのデータは0,1の列で表される
    ビット
- 圧縮とは
  - データを表現する0,1列の長さを短くすること
  - 圧縮されたデータから元のデータを求めることを，
    復元，伸長，復号，解凍などと言う

2種類の圧縮法

- 可逆圧縮
  - 完全に元に戻る圧縮法
  - 文書，プログラムなど
- 非可逆圧縮
  - 人間には区別できない程度の違いがある
  - 画像 (JPEG)，音声 (mp3)，動画 (MPEG) など

- 今日の話は可逆圧縮のみ

本の検索

- 人が検索するとき
  - 本の索引を使う
  - 索引に載っていない単語は見つからない
- 計算機で検索するとき
  - 本の索引のようなデータ構造を用いる
  - 索引の見出しを増やせば，データ量 (索引のページ数) も増える
  - 全単語を索引に載せると本のページ数が倍以上になる

索引の例

- NTCIR-4 PATENT (日本語特許公報全文5年分)
  - 文書数　3,496,252
  - サイズ　113.8G
  - bzip2での圧縮サイズ　15.2G
  - 従来の索引 (接尾辞配列) + データ　680.4G

簡潔データ構造 (索引+データ)　21.6G
約 1/30 に圧縮

簡潔データ構造

- データに索引を追加しても，サイズが増えない (データも圧縮できる)
- 圧縮されていない索引を用いた場合とほぼ同じ速度で検索ができる

データ圧縮の基本

- 次のデータは何を表しているでしょう？

1000110110010001100101111010011 1
1000111111101110100101011111000 1
1000101001110111110001100101001 00
1000101110000110100011111000 1010

1000 1101 1001 0001 → 8D 91 → 国
1001 0111 1010 0111 → 97 A7 → 立
1000 1111 1110 1110 → 8F EE → 情
1001 0101 1111 0001 → 95 F1 → 報
1000 1010 0111 0111 → 8A 77 →学
1000 1100 1010 0100 → 8C A4 → 研
1000 1011 1000 0110 → 8B 86 → 究
1000 1111 1000 1010 → 8F 8A → 所

データ圧縮法

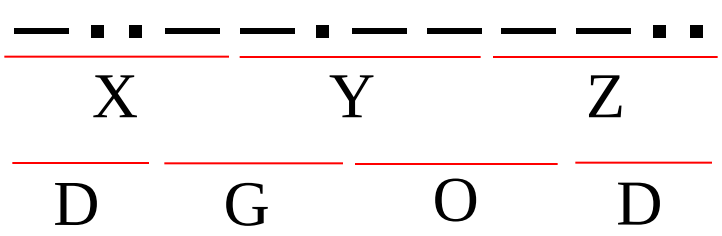
- 「サクラサク」
  - 長い文章を短いものに置き換える
  - 本当は1ビットで良い (1 = 合格, 0 = 不合格)
- 全ての圧縮法はこれを行っている
- 1ビットだと，2種類の文章しか表現できない
- 特定の文章だけでなく，どんな文章でも表現できるようにするには？

モールス符号 (信号)

- 1830年代に提案
- 短点・と長点 ー の組み合わせ
- 長点は短点3つ分の長さ
- SOS = ・・・ー ー ー・・・
- 英語で使われる頻度の高い e, t は短い符号・, ー になっている
  ⇒データ圧縮

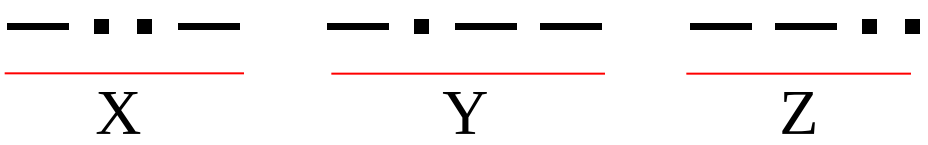
文字	符号	文字	符号
A	・ー	N	ー・
B	ー・・・	O	ーーー
C	ー・ー・	P	・ーー・
D	ー・・	Q	ーー・ー
E	・	R	・ー・
F	・・ー・	S	・・・
G	ーー・	T	ー
H	・・・・	U	・・ー
I	・・	V	・・・ー
J	・ーーー	W	・ーー
K	ー・ー	X	ー・・ー
L	・ー・・	Y	ー・ーー
M	ーー	Z	ーー・・

- 次の符号は何を表しているでしょう？



- 1つの文字を表す符号がどこで切れているのか分からない
「一意に復号可能」ではない

- モールス符号では，文字間には短点3つ分の空きを入れる

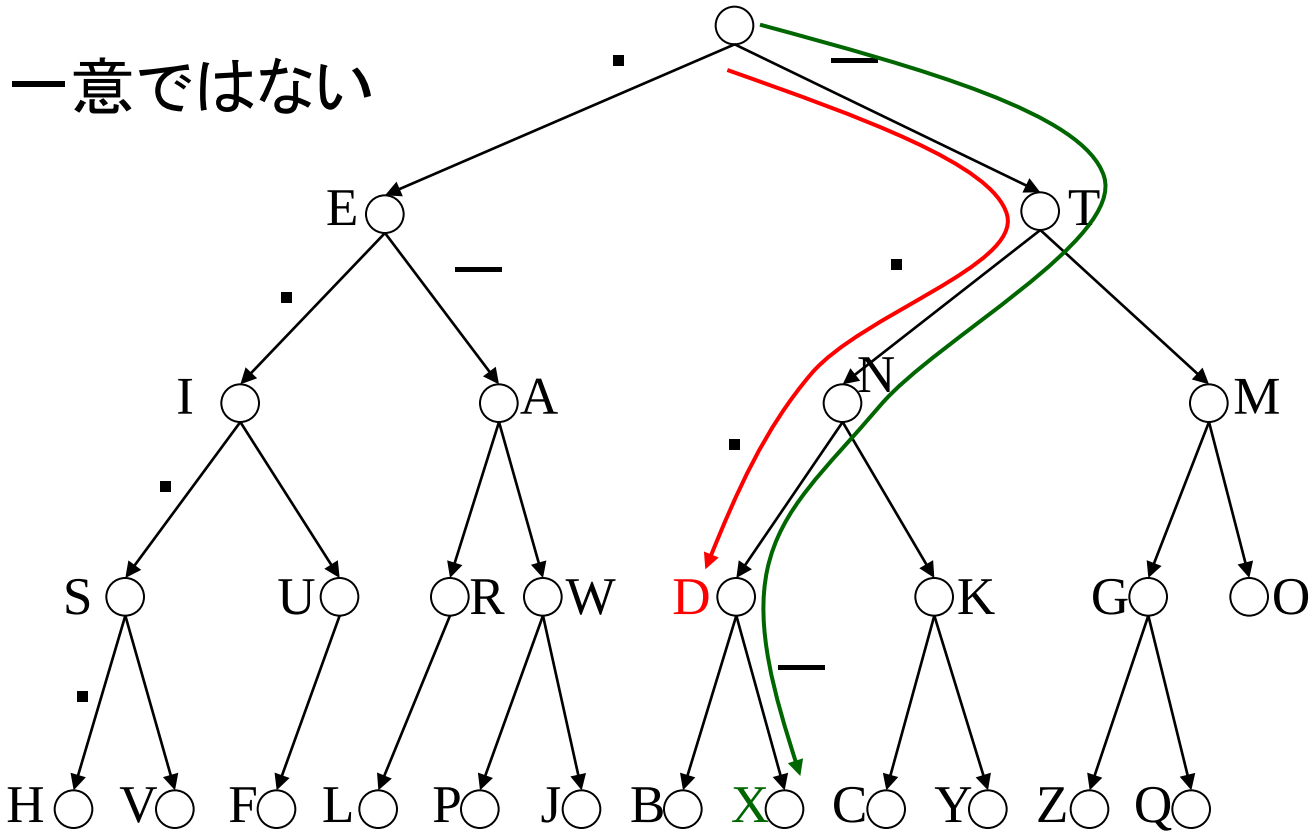


文字	符号	文字	符号
A	・－	N	－・
B	－・・・	O	－－－
C	－・－・	P	・－－・
D	－・・	Q	－－・－
E	・	R	・－・
F	・・－・	S	・・・
G	－－・	T	－
H	・・・・	U	・・－
I	・・	V	・・・－
J	・－－－	W	・－－
K	－・－	X	－・・－
L	・－・・	Y	－・－－
M	－－	Z	－－・・

一意に復元可能な符号

- 符号を先頭から (左から) 見ていったときに，1通りにしか復号されない符号
- モールス符号では空白を入れる必要がある

一意ではない

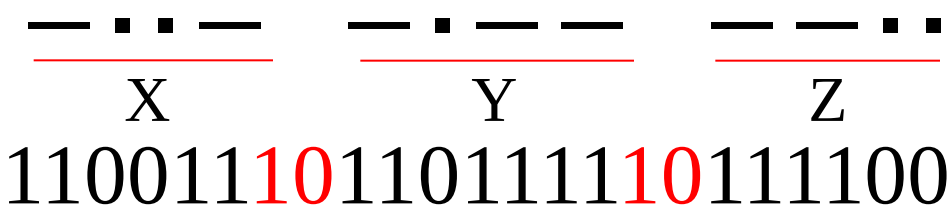


文字	符号	文字	符号
A	・－	N	－・
B	－・・・	O	－－－
C	－・－・	P	・－－・
D	－・・	Q	－－・－
E	・	R	・－・
F	・・－・	S	・・・
G	－－・	T	－
H	・・・・	U	・・－
I	・・	V	・・・－
J	・－－－	W	・－－
K	－・－	X	－・・－
L	・－・・	Y	－・－－
M	－－	Z	－－・・

コンピュータで表現するには？

モールス符号には短点，長点，空白の3種類の文字が必要

- これらを 0,1 のビットで表現する必要がある
- 例:
  - 短点 → 0
  - 長点 → 11
  - 空白 → 10



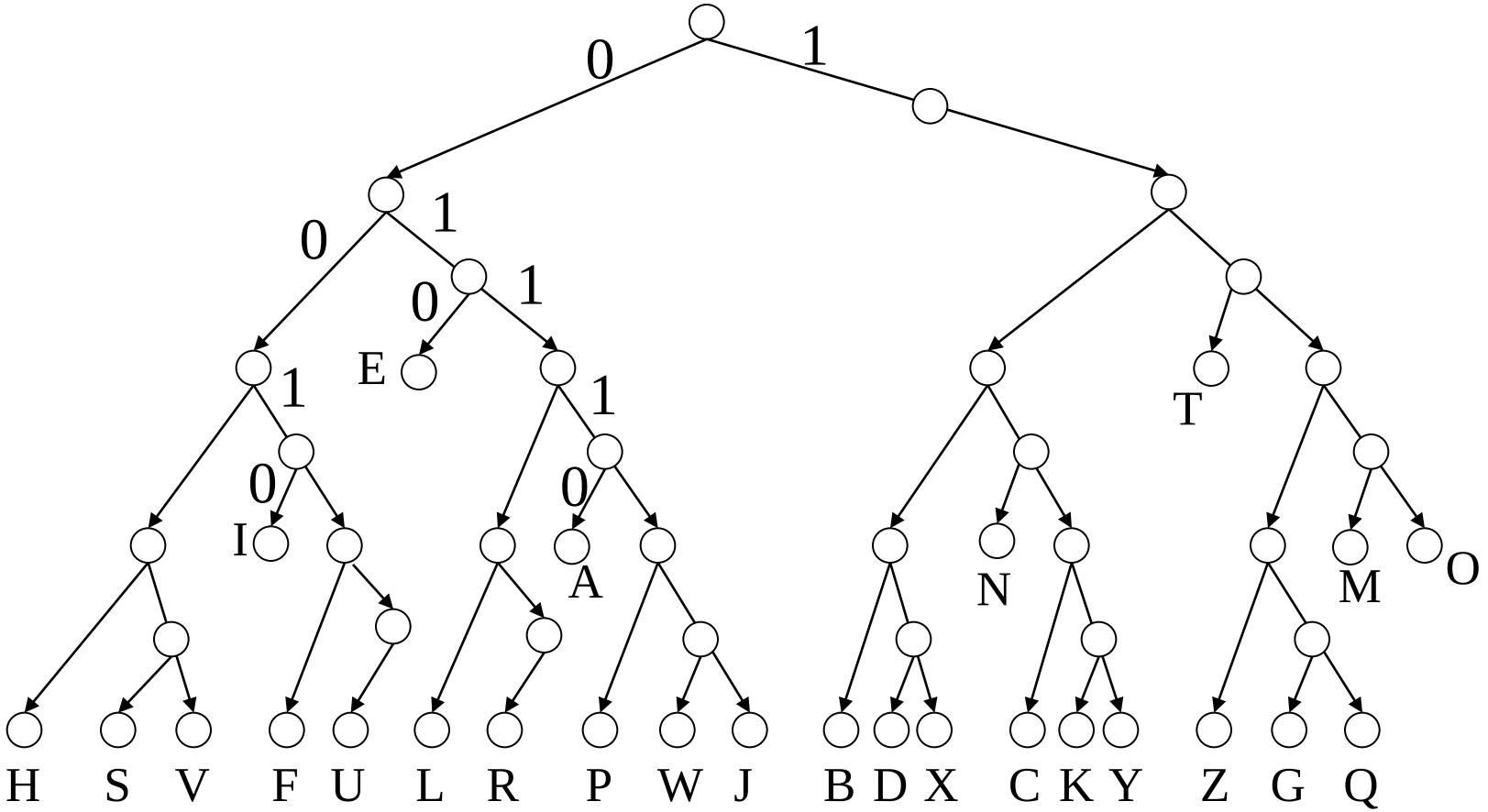
各文字の符号の最後に，空白を表す10 をつけると一意に復号可能になる

E: ・　⇒ 0 10

I: ・・　⇒ 0 0 10

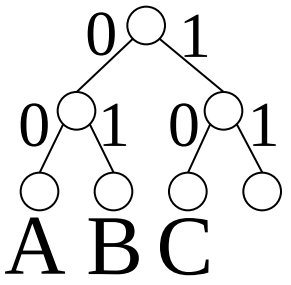
A: ・ー ⇒ 0 11 10

木の内部に符号が割り当てられない

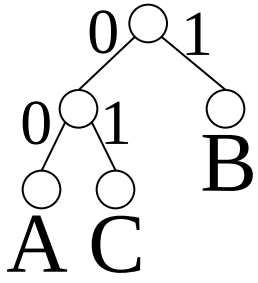


どのような符号が良いか

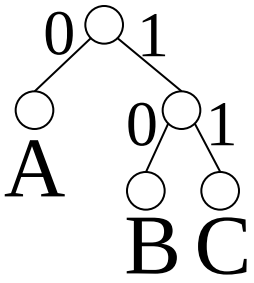
- ABCABCAABAACAABC を圧縮する場合
- 符号1: A ⇒ 00 B ⇒ 01 C ⇒ 10 のとき
  00011000011000000100000100000110
  2 × 8 + 2 × 4 + 2 × 4 = 32 ビット



- 符号2: A ⇒ 00 B ⇒ 1 C ⇒ 01 のとき
  00101001010000100000100000101
  2 × 8 + 1 × 4 + 2 × 4 = 28 ビット



- 符号3: A ⇒ 0 B ⇒ 10 C ⇒ 11 のとき
  0101101011001000110010 11
  1 × 8 + 2 × 4 + 2 × 4 = 24 ビット



- 符号1,2,3の中で，符号3が一番小さくなっている
  - 多く現れる文字 (A) の符号が短いから
  - A: 8回, B: 4回, C: 4回
- どのような符号を使うと一番圧縮できるか？
- 文字 A, B, C の符号の長さを x, y, z とする
- 文字列の符号の長さは $L = 8x + 4y + 4z$
- 一意に復号できる符号のみ考える

$$\frac{1}{2^x}+\frac{1}{2^y}+\frac{1}{2^z}\leq 1$$ を満たす必要がある
(クラフトの不等式)

- L が最小になる x, y, z は？
  - x =1, y = 2, z = 2 つまり符号3が最適

エントロピー

- 文字列中の文字 c の出現確率を $p(c)$ とすると，
  文字列のエントロピーは次のように定義される

$$H = \sum_{c \in A} p(c) \cdot \log_2 \left(\frac{1}{p(c)} \right)$$

A: アルファベット (文字の集合)
$A = \{\mathrm{A}, \mathrm{B}, \mathrm{C}, \ldots, \mathrm{Z}\}$

- 文字 A の出現確率は $\frac{8}{16} = \frac{1}{2}$
- 文字 B, C の出現確率は $\frac{4}{16} = \frac{1}{4}$

$$H = \frac{1}{2} \log_2 \frac{2}{1} + \frac{1}{4} \log_2 \frac{4}{1} + \frac{1}{4} \log_2 \frac{4}{1}$$

$$= \frac{1}{2} \cdot 1 + \frac{1}{4} \cdot 2 + \frac{1}{4} \cdot 2 = \frac{3}{2}$$

- どんな符号でも，1文字あたりの平均ビット数はエントロピーよりも小さくならない
- 符号3の1文字あたりの平均ビット数は $\frac{24}{16}=\frac{3}{2}$ でエントロピーと一致する

$$H=\frac{1}{2}\log_2\frac{2}{1}+\frac{1}{4}\log_2\frac{4}{1}+\frac{1}{4}\log_2\frac{4}{1}$$

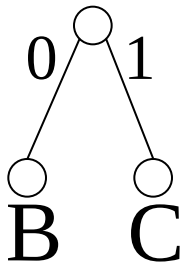
$$=\frac{1}{2}\cdot 1+\frac{1}{4}\cdot 2+\frac{1}{4}\cdot 2=\frac{3}{2}$$

Aの符号長

Aの出現確率

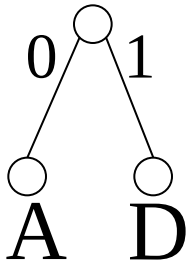
ハフマン符号 (1952年)

- 平均符号長が最小の (= 最も圧縮できる) 符号
- 作り方
  - 出現頻度が最小の文字と，2番目に少ない文字に
    0 と 1 を割り当てる
    それらの文字を合わせて，1つの文字にする
  - A: 8回, B: 4回, C: 4回 のとき，Bに0, Cに1を割り当てる



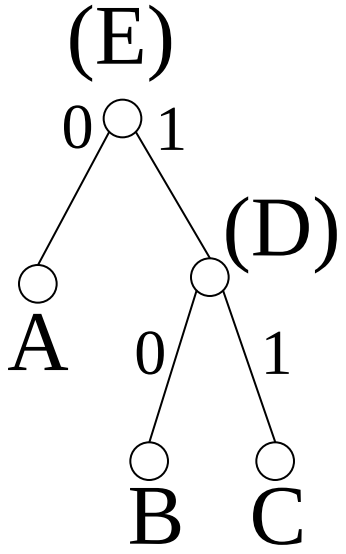
  - B と C を合わせて文字 D が 8 回現れたとみなす
  - 文字が1種類になるまで繰り返す

– A: 8回, D: 8回なので，Aに0, Dに1を割り当てる



– A と D を合わせて文字 E が16回現れたとみなす

– 終了



- ハフマン符号の平均符号長 L は

$$H \leq L < H+1$$

データ圧縮の欠点は？

- 復元が遅い
- 一部分だけ復元することが難しい
- ABCABCAABAACAABC の10文字目は？
- 符号1: A ⇒ 00 B ⇒ 01 C ⇒ 10 のとき
  2×10 = 20ビット目を見ればいい
  0　　1　　2　　3
  0　　0　　0　　0
  00011000011000000100001000000110
- 符号3: A ⇒ 0 B ⇒ 10 C ⇒ 11 のとき
  何ビット目を見ればいいか分からない
  01011010110010001100 1011

- 符号1のように全ての文字が同じビット数の符号を固定長の符号と呼ぶ
  - 文字は$\log \sigma$ビットで表現される(σ:アルファベットサイズ)
- コンピュータで扱うデータは固定長の符号で表現されている (無圧縮)
- 符号3のように可変長の符号を使うと圧縮できるが復元が遅くなる
  - 先頭から1文字ずつ復元していかなければならない

  0101101011001000110010 11

部分復号の高速化

- ABCABCAABAACAABCの10文字目を復元する
  0101101011001000110010 11
- 途中から復元するために，符号の開始位置を記憶
  - 1 文字目は 1 ビット目から
  - 6 文字目は 9 ビット目から
  - 11文字目は 16 ビット目から
  - 16文字目は 23 ビット目から

A ⇒ 0
B ⇒ 10
C ⇒ 11

- どの文字を復元する場合でも，最大 5 文字復元すればいい
- 開始位置は何ビットで記憶できる？

開始位置の記憶

- 文字列の長さを n とする ($n = 16$)
- アルファベットサイズ σ は n 以下とする
- 圧縮後のサイズを m とする
  - $m \leq n \log \sigma \leq n \log n$
- d 文字おきに開始位置を記憶する ($d = 5$)
  - n/d 個の開始位置を記憶する
- 1個の開始位置は何ビットで表現できるか
  - 開始位置は 1 から m の整数
  - $\log m$ ビットで表現できる

- 全ての開始位置を記憶するには

$$\frac{n\log m}{d} \le \frac{n\log(n\log n)}{d} < \frac{2n\log n}{d}$$ ビット必要

- $d = 2\log n$ とすると，開始位置は n ビット以下で記憶できる
  - 1文字あたり 1 ビット以下
- ある文字を復元するために必要な時間は d に比例する
  - d を大きくすると圧縮率が良くなるが復元が遅くなる

開始位置の圧縮

- d を小さくすると文字の復元は早くなるが，
  サイズが大きくなってしまう
- 開始位置をビット列で表現する
- ABCABCAABAACAABC

0101101011001000110010 11 圧縮された文字列

1000000010000001000000 10 5文字おきの開始位置

- ビット列の
  - 1番目の1の位置 (1) ･･･ 1文字目
  - 2番目の1の位置 (9) ･･･ 6文字目
  - 3番目の1の位置 (16) ･･･11文字目
  - 4番目の1の位置 (23) ･･･16文字目

簡潔データ構造

ビットベクトル

- B: 長さ n の 0,1 ベクトル $B[0]B[1]\ldots B[n-1]$
- 操作
  - $rank(B, x)$: $B[0..x] = B[0]B[1]\ldots B[x]$ 内の 1 の数
  - $select(B, i)$: B の先頭から i 番目の 1 の位置 ($i \geq 1$)
- *select* 操作で開始位置が求まる
  - $select(B, 1) = 1$ ・・・ 1文字目の開始位置
  - $select(B, 2) = 9$ ・・・ 6文字目の開始位置
  - $select(B, 3) = 16$ ・・・ 11文字目の開始位置

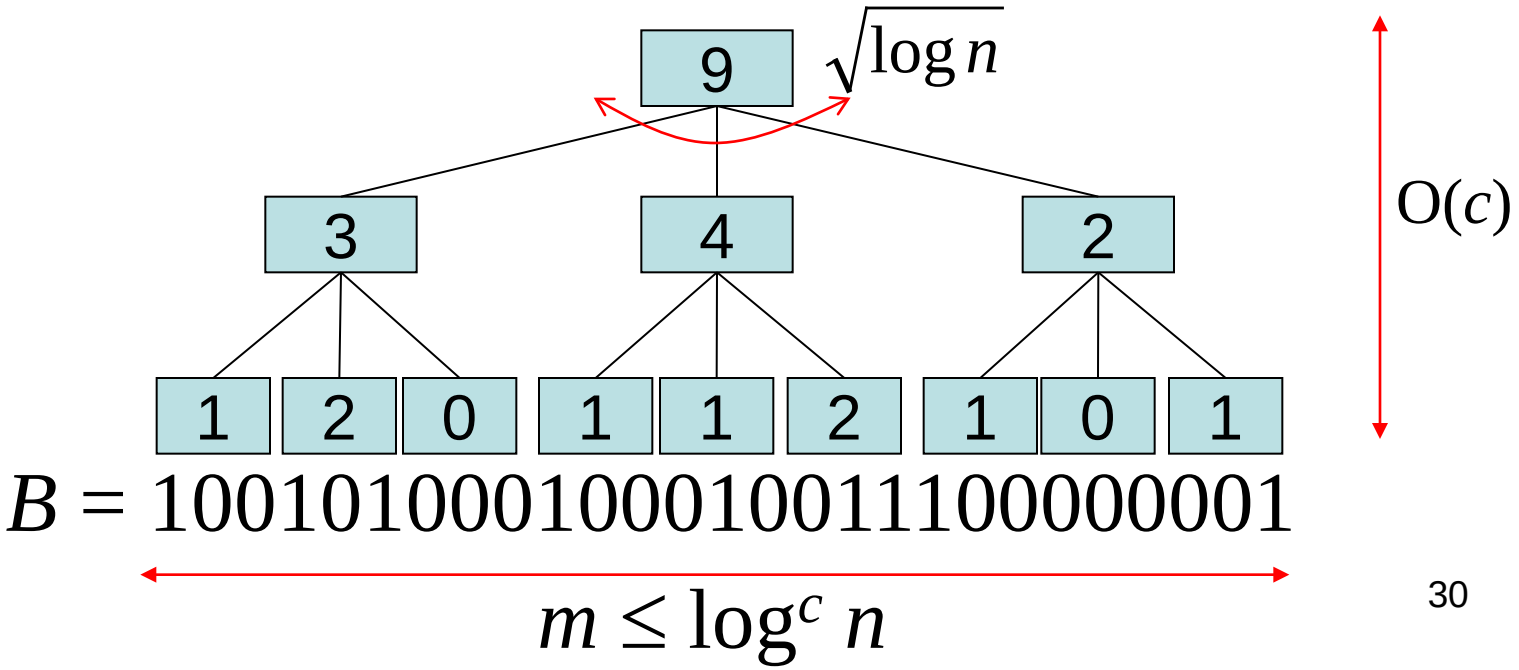
$B = 100000001000000100000010$

$n = 24$

Selectの求め方

- B を，1 を $\log^2 n$ 個含む大ブロックに分割
- 大ブロックごとに2通りのデータ構造を使い分ける
- 大ブロックの長さが $\log^c n$ を超えるとき
  - $\log^2 n$ 個の1の位置をそのまま格納
  - 1つの大ブロックで $\log^2 n \cdot \log n = \log^3 n$ bits
  - そのような大ブロックは最大 $\frac{n}{\log^c n}$ 個
  - 全体で $\frac{n}{\log^c n} \cdot \log^3 n$ bits
  - $c = 4$ とすると $\frac{n}{\log n}$ bits

- 大ブロックの長さ m が $\log^c n$ 以下のとき
  - 長さ $\frac{1}{2} \log n$ の小ブロックに分割
  - 小ブロックを葉に持つ完全 $\sqrt{\log n}$ 分木を構築
  - 木のノードには，その子孫のベクトルの1の数を格納
  - 大ブロック内の 1 の数は $\log^2 n$
    → 各ノードの値は $2 \log \log n$ bits

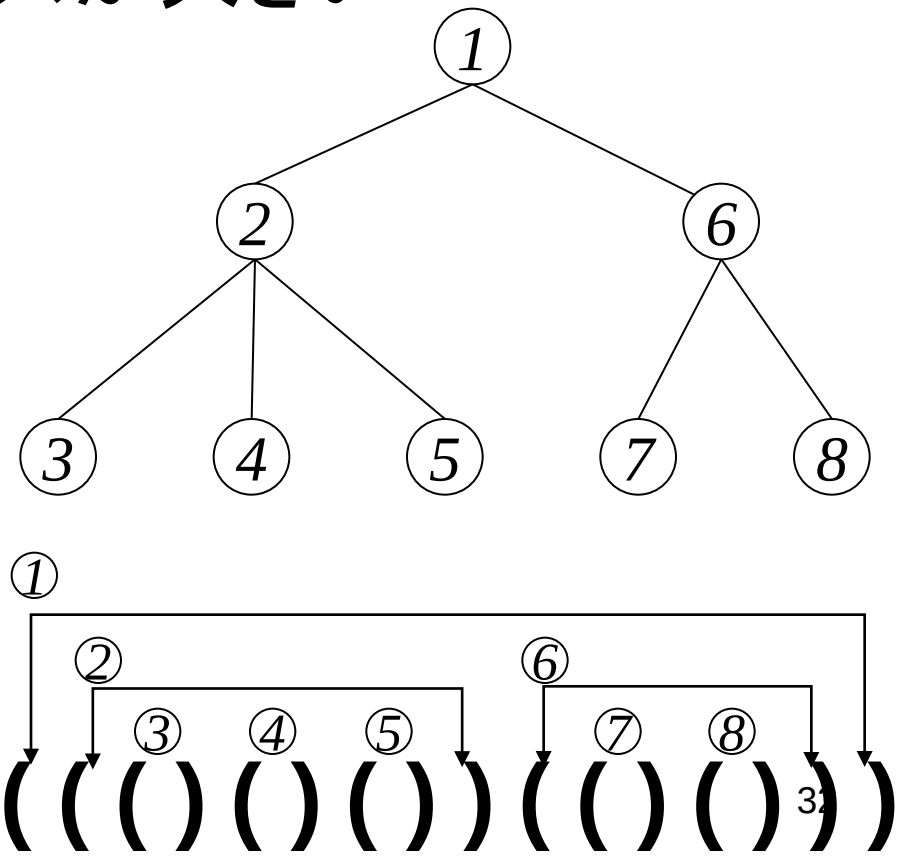


- B は約 n ビット $(n + o(n))$ で表現でき，*rank* と *select* は答えを圧縮しないで記憶した場合と同じ時間 (定数時間) で計算できる

- これを簡潔データ構造と呼ぶ

順序木の簡潔データ構造

- 各節点を対応する括弧のペアで表現
- n 節点で $2n$ bits (サイズの下限と一致)
- 既存手法は複雑でサイズが大きい



圧縮全文索引ライブラリ

- フリーソフトとして公開　http://researchmap.jp/sada/csalib/
- NTCIR-4 PATENT (日本語特許公報全文　1993-1997)
  - 文書数 3,496,252
  - サイズ (タグ除去, utf8)　113.8G
  - bzip2での圧縮サイズ　15.2G
  - 圧縮索引のサイズ　21.6G
  - 従来の索引 (接尾辞配列)　680.4G
- 文書中の頻出パタンの列挙　36秒
  - (文字列長の0.01%以上の頻度)
  - 出力例
  - [99828894586,99842145324] key = 図である。
  - [89135087764,89147590131] key = ることを特徴とする
  - [82281197875,82293217686] key = することができる。