

簡潔データ構造 第 5 回

Wavelet Matrix

marimo

2023 年 5 月 30 日

rank と select の拡張

ビットベクトルの rank と select は文字列に拡張できる。

アルファベット A 上の長さ n の文字列 S , $c \in A$ について

- $\text{access}(i) = S[i]$
- $\text{rank}_c(i) = S[0..i]$ 中の c の数を返す
- $\text{select}_c(i) = S$ 中の i 番目の c の位置を返す

Wavelet Tree

一旦 $\sigma = |A|$ を 2 ベキと仮定する。

定理 (Wavelet Tree).

長さ n , アルファベットサイズ σ の文字列に対して, $(n + o(n))\lg \sigma + O(\sigma \lg n)$ bits のデータ構造が存在して, access, rank, select は $O(\lg \sigma)$ 時間で計算できる。

Wavelet Tree

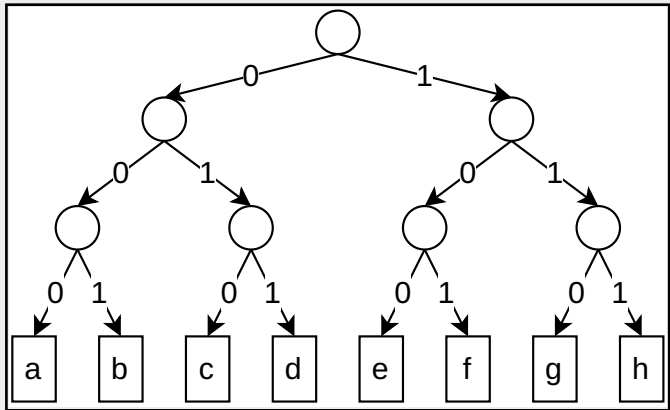
次のように構成される Wavelet Tree によって達成できる。 A の符号には自明な符号化を使う。

- 各ノードにビットベクトルが載っているポインタによる完全 2 分木を作る。
- 符号の i ビット目のビットは深さ i のノードに次のように乗る
 - 根のベクトルは長さ n で, $B[i] = C(S[i])[0]$ で定める。
 - ノード $v_{b_0 \dots b_{i-1}}$ のベクトルは
 - 符号の先頭が $b_0 \dots b_{i-1}$ というビット列になっている文字の符号の i ビット目のビットを文字列の順番で格納する

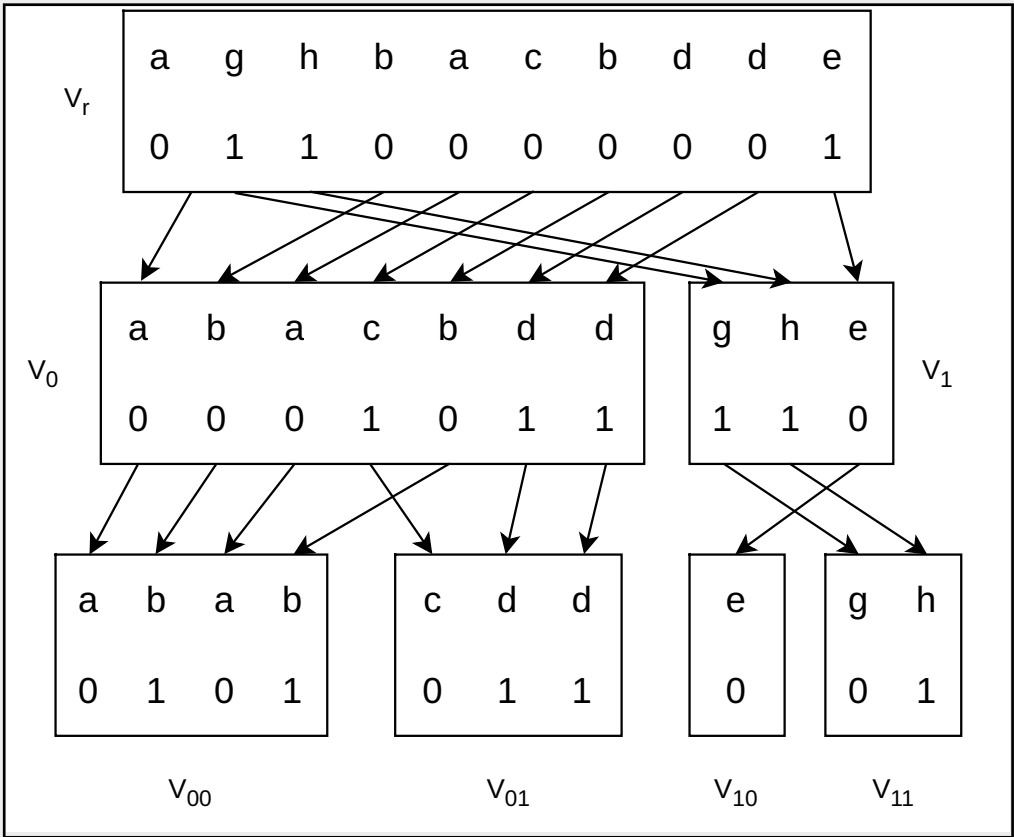
これでは分らんので次の図を見る^{*1}

¹ 基数ソートを途中経過を全部持つてみたいな感じ

Wavelet Tree



自明な符号木 (の 1 つ)



Wavelet Tree

Wavelet Tree

$\text{access}(i)$ は次のようにする

- $s = V[i]$
- $i = \text{rank}(V, i)$
- d を 1 から $\lg \sigma$ まで
 - $b = V_s[i]$
 - $i = \text{rank}(V_s, i)$
 - $s = s + b$
- s のうしろに b をくっつける

Wavelet Tree

$\text{rank}_c(i)$ は次のようにする

- $t = C(c)$
- d を 0 から $\lg \sigma - 1$ まで
 - $b = t[d]$
 - $i = \text{rank}_b(V_{t[0..d]}, i)$

Wavelet Tree

$\text{select}_c(i)$ は次のようにする

- $t = C(c)$
- d を $\lg \sigma - 1$ から 0 まで
 - $b = t[d]$
 - $i = \text{select}(V_{t[0..d]}, i)$

Wavelet Matrix

Wavelet Tree で、同じ深さのベクトルの長さを全部足すと n になる
そうしたらこれを一つのベクトルにしてしまったら簡単になるのでは？
これが Wavelet Matrix

Wavelet Matrix

Wavelet Matrix

Wavelet Matrix

Huffman 符号の使用

これまでアルファベットのサイズを 2 べきと仮定していた。

Wavelet Tree で自明な符号化でなくて, Huffman 符号を使うようにすればアルファベットサイズが 2 べきでなくても簡潔な構造にすることができる。

うれしい