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

- $\mathrm{access}(i) = S[i]$
- $\mathrm{rank}_c(i) = S[0..i]$の中の c の数を返す
- $\mathrm{select}_c(i) = S$の中の 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，$\big(n + o(n)\big)\lg\sigma + O\big(\sigma \lg n\big)\,\mathrm{bits}$ のデータ構造が存在して，$\mathrm{access}, \mathrm{rank}, \mathrm{select}$ は $O\big(\lg \sigma\big)$ 時間で計算できる。

Wavelet Tree

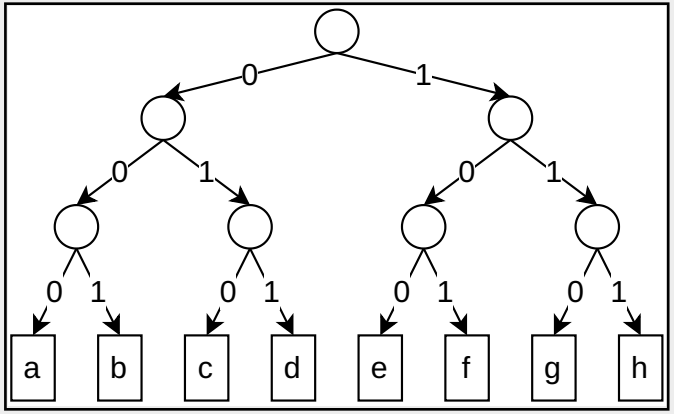
次のように構成される Wavelet Tree によって達成できる。A の符号には自明な符号化を使う。

- 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，$B[i] = C\big(S[i]\big)[0]$ 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

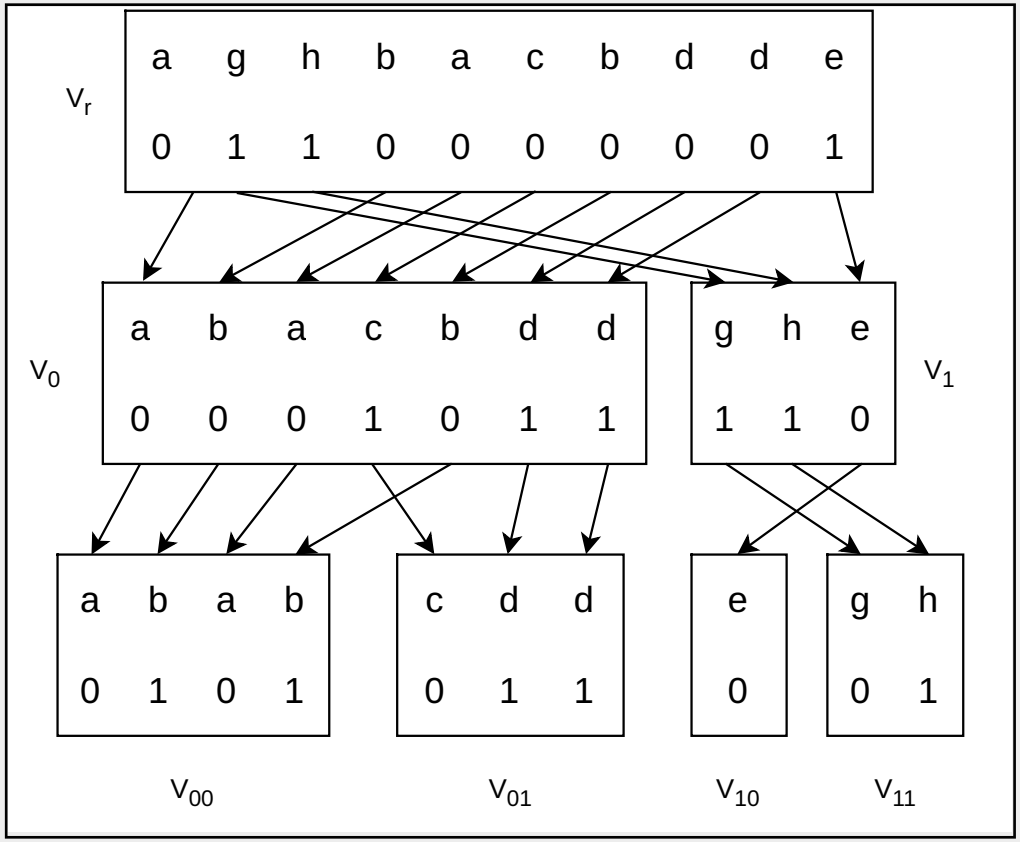
これでは分からんので次の図を見る[*1]

1 基数ソートを途中経過を全部持ってるみたいな感じ

Wavelet Tree



自明な符号木 (の 1 つ)



Wavelet Tree

Wavelet Tree

$\mathrm{access}(i)$ は次のようにする

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

Wavelet Tree

$\mathrm{rank}_c(i)$ は次のようにする

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

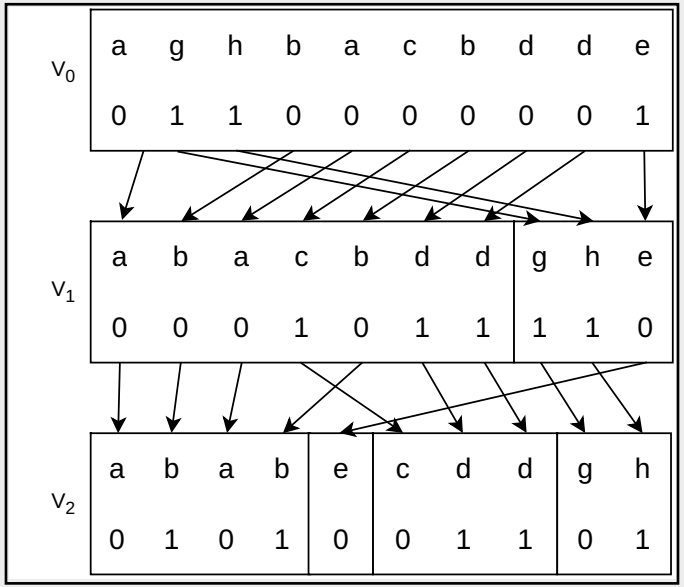
Wavelet Tree

$\mathrm{select}_c(i)$ は次のようにする

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

Wavelet Matrix

Wavelet Tree で，同じ深さのベクトルの長さを全部足すと n でかわらない
全部を一つのベクトルにしてしまったら簡単になるのでは $\rightarrow$Wavelet Matrix
簡単のために深さ d のベクトルをつなげたものを V_d と書くが，実際は V_d は一本のベクトル V の一部



Wavelet Matrix

$\mathrm{access}(i)$ は次のようにする

- $s = NULL$
- d を 0 から $\lg \sigma - 1$ まで
  - $b = V_d[i]$
  - $s = s \circ b$
  - $w = \mathrm{rank}_0\big(V_d, n\big)$
  - if $b = 0$ then $i = \mathrm{rank}_0\big(V_d, i\big)$ else $i = w + \mathrm{rank}_1\big(V_d, i\big)$

Wavelet Matrix

$\mathrm{rank}_c(i)$ は次のようにする

- $t = C(c)$
- $l, r = 0, i$
- d を 0 から $\lg \sigma - 1$ まで
  - $b = t[d]$
  - $w = \mathrm{rank}_0(V_d, n)$
  - if $b = 0$ then $r = \mathrm{rank}_0(V_d, i)$ else $l = w, r = w + \mathrm{rank}_1(V_d, r)$
- $r - l$

Wavelet Matrix

$\mathrm{select}_c(i)$ については疑似コードを書くと長いのでお気持ちだけ書くと

- まず $C(c)$ を使って降りていって，c がいる範囲を知る
- この範囲で $\mathrm{select}_b(i)$ をつかって位置を知る
- 逆向きに上がっていく
  - rank で降りていたことから分かるように逆演算の select で上がる

Huffman 符号の使用

これまでアルファベットのサイズを 2 ベキと仮定していた。
Wavelet Tree で自明な符号化でなくて，Huffman 符号[*2]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

2 prefix code ならなんでも OK

その他の演算

詳しくは述べないが，符号化が順序を保つ場合*3*4いろいろできる

- 区間内の文字の列挙
- 区間内のある範囲に含まれる文字の総数
- 区間内の k 番目に大きい文字の位置
- etc.

3 先の Huffman Code は順序を保たないことに注意

4 この条件を満たすような符号は条件が無い場合に比べて多少長くなることが知られている。最適なものを Hu-Tucker Algorithm で構成できる。