BWT Performance

6.897
L13.5
Apr. 9, 2003

- for any substring w, BWT puts successors(w) together (~~Huge permutation~~)
- Thus BWT partitions string into successors(w_1)|successors(w_2)|---
  where $w_1, w_2, \ldots$ are all strings len. k
- if we could encode $x = x_1 x_2 \cdots x_r$ in $\sum_{i=1}^{r} |x_i| H_0(x_i)$
  then encoding BWT would give us H_k — for all k!
  ~ we'll do it only roughly

MTF transform: instead of writing letters write index of letter in list & move to front
⇒ resulting string has same length, same alphabet size

Theorem: [Manzini – JACM2001]

$$\text{BWT} + \text{MTF} + \text{arithmetic} \leq \underbrace{8 \cdot |T| \cdot H_k(T)}_{\forall k} + \underbrace{\frac{2}{25} \cdot |T|}_{\text{annoying}} + \underbrace{O(\Sigma^{k+1} \lg \Sigma)}_{\text{like } k^{\text{th}} \text{ order Huffman}}$$

Runlength transform: replace string of 0's with a binary string encoding length using special 0/1 symbols for clarity

1 2 0 0 0 0 0 0 5 3 ---
→ 1 2 1 1 0 5 3 ---

Theorem: [Manzini – JACM2001]

$$\text{BWT} + \text{MTF} + \text{runlength} + \text{arithmetic} \leq 5 \cdot |T| \cdot H_k^*(T) + f(k)$$

OPEN: improve constants