counting colours in compressed strings

Travis Gagie
Juha Kärkkäinen

CPM 2011

counting colours in compressed strings

Travis Gagie
Juha Kärkkäinen

CPM 2011

Theorem

Given a string $s[1..n]$*, we can build a data structure that takes* $nH_0(s) + \mathcal{O}(n) + o(nH_0(s))$ *bits such that later, given a substring's endpoints* i *and* j*, in* $\mathcal{O}(\log \ell)$ *time we can count how many distinct characters it contains, where* $\ell = j - i + 1$*.*

source	space	time
BKM&T	$\mathcal{O}(n \log n)$	$\mathcal{O}(\log n)$
Muthu + WT	$n \log n + o(n \log n)$	$\mathcal{O}(\log n)$
GN&P	$n \log \sigma + \mathcal{O}(n \log \log n)$	$\mathcal{O}(\log n)$
this paper	$nH_0(s) + \mathcal{O}(n) + o(nH_0(s))$	$\mathcal{O}(\log \ell)$

counting colours in compressed strings

[c, o, u, n, t, i, n, g, c, o, l, o, u, r, s, i, n,
c, o, m, p, r, e, s, s, e, d, s, t, r, i, n, g, s]

[0, 0, 0, 0, 0, 0, 4, 0, 1, 2, 0, 10, 3, 0, 0, 6, 7,
9, 12, 0, 0, 14, 0, 15, 24, 23, 0, 25, 5, 22, 16, 17, 28]

counting colours in compressed strings

[c, o, u, n, t, i, n, g, c, o, l, o, u, r, s, i, n,
c, o, m, p, r, e, s, s, e, d, s, t, r, i, n, g, s]

[0, 0, 0, 0, 0, 0, 4, 0, 1, 2, 0, 10, 3, 0, 0, 6, 7,
9, 12, 0, 0, 14, 0, 15, 24, 23, 0, 25, 5, 22, 16, 17, 28]

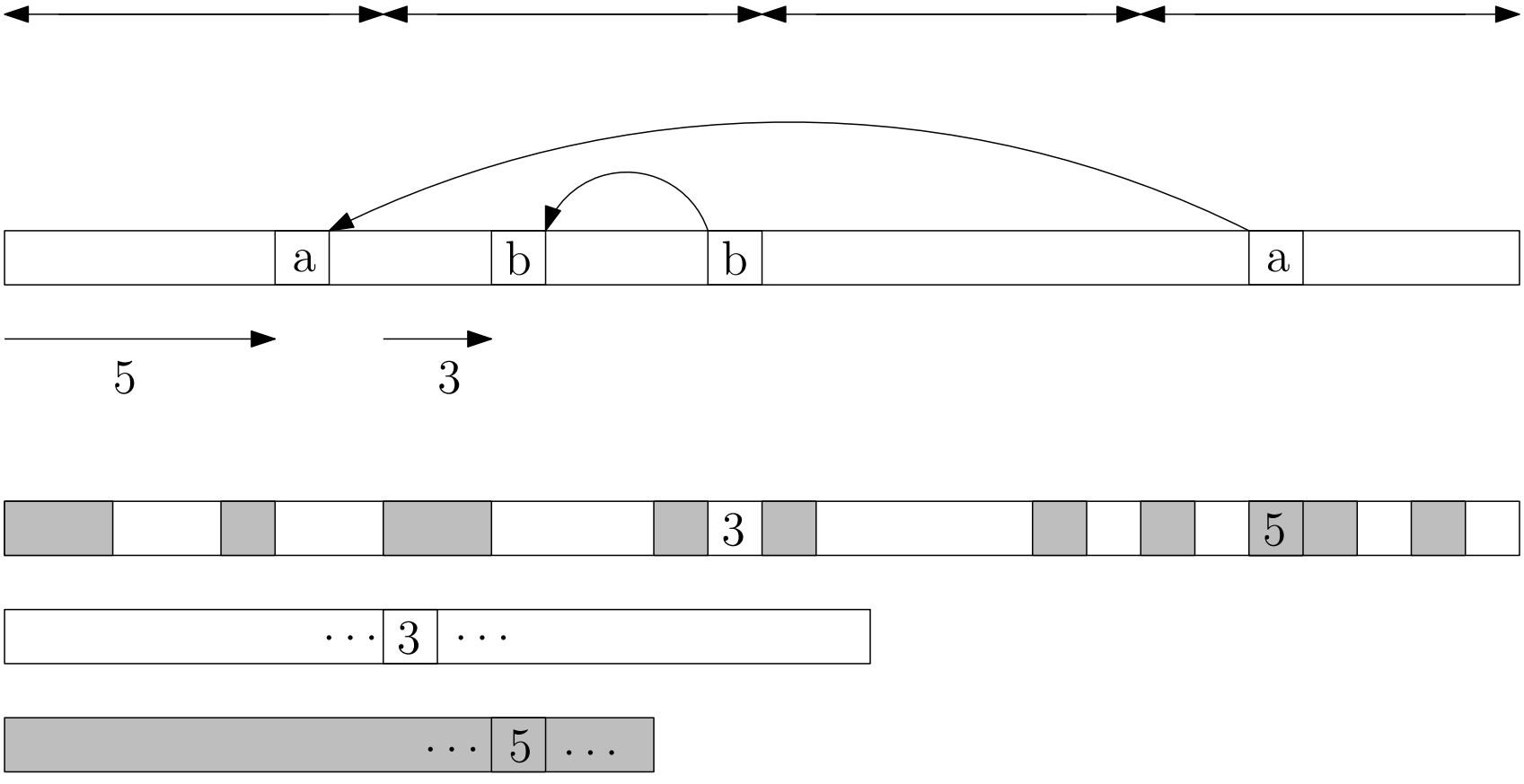
counting colours in compressed strings

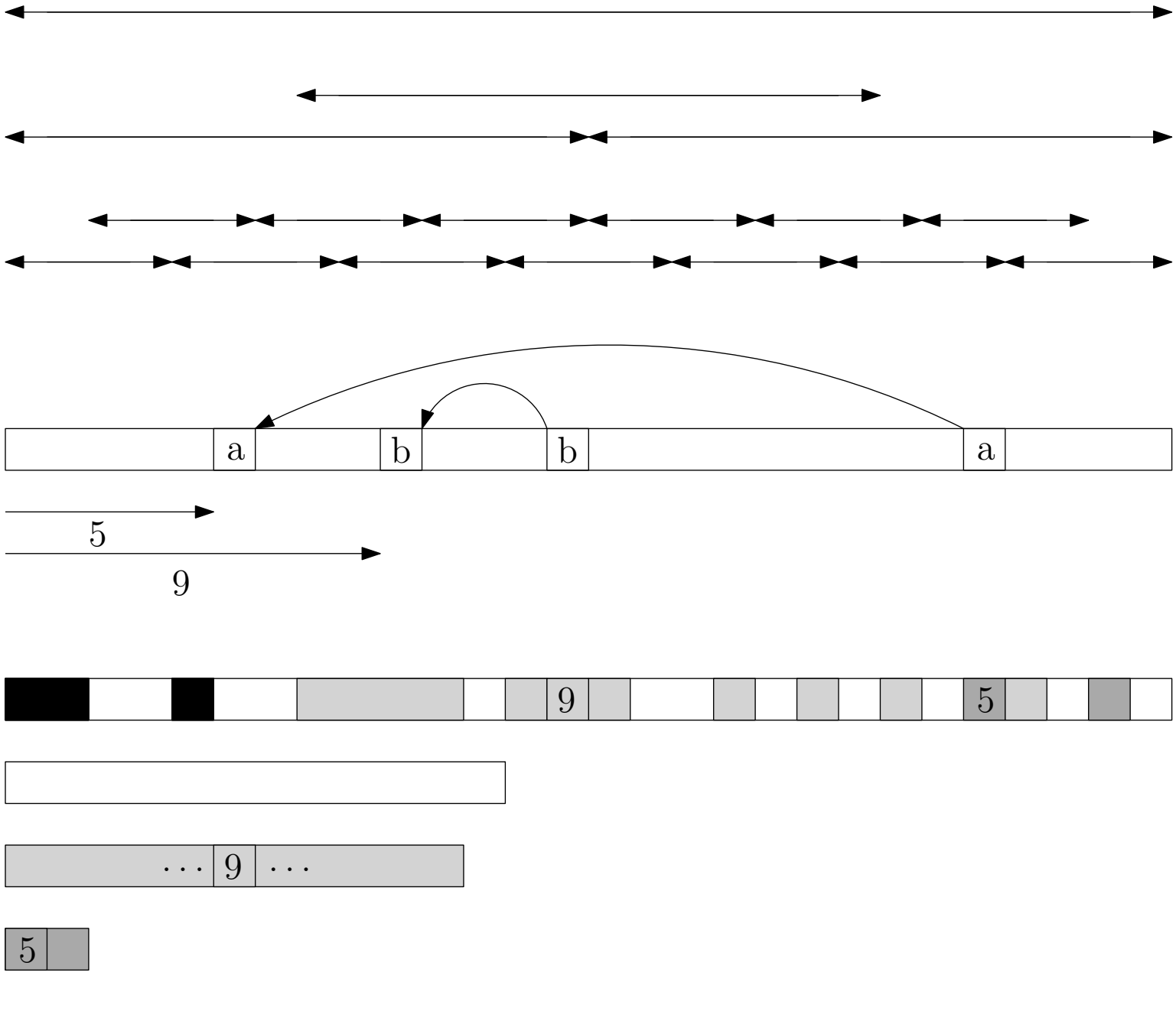
[c, o, u, n, t, i, n, g, c, o, l, o, u, r, s, i, n,
c, o, m, p, r, e, s, s, e, d, s, t, r, i, n, g, s]

[0, 0, 0, 0, 0, 0, 4, 0, 1, 2, 0, 10, 3, 0, 0, 6, 7,
9, 12, 0, 0, 14, 0, 15, 24, 23, 0, 25, 5, 22, 16, 17, 28]

source	space	time
BKM&T	$\mathcal{O}(n \log n)$	$\mathcal{O}(\log n)$
Muthu + WT	$n \log n + o(n \log n)$	$\mathcal{O}(\log n)$
GN&P	$n \log \sigma + \mathcal{O}(n \log \log n)$	$\mathcal{O}(\log n)$
this paper	$nH_0(s) + \mathcal{O}(n) + o(nH_0(s))$	$\mathcal{O}(\log \ell)$

source	space	time
BKM&T	$\mathcal{O}(n \log n)$	$\mathcal{O}(\log n)$
Muthu + WT	$n \log n + o(n \log n)$	$\mathcal{O}(\log n)$
GN&P	$n \log \sigma + \mathcal{O}(n \log \log n)$	$\mathcal{O}(\log n)$
this paper	$nH_0(s) + \mathcal{O}(n) + o(nH_0(s))$	$\mathcal{O}(\log \ell)$


a
b
b
a
5
3
3
5
. . .
3
. . .
. . .
5
. . .


a
b
b
a
5
9
9
5
. . .
9
. . .
5

Components:

- multiary wavelet tree assigning entries to blocks
- wavelet tree for each block (with a shared bitvector for each block size and depth)

Observations:

- if we use more block sizes, the C array becomes more like recency coding and compression is better (but queries take more time)
- if we use polylog(n) block sizes, then we can count the entries much bigger than ℓ in $\mathcal{O}(1)$ time using the multiary wavelet tree

Calculation:

- if we use block sizes

$$b_k = \begin{cases} 2 & k = 1 \\ 2^{\max\left(\prod_{h=1}^{k-1}(1+1/\alpha(b_h)),k\right)} & k > 1 \end{cases}$$

then we use a total of $nH_0(s) + \mathcal{O}(n) + o(nH_0(s))$ bits and $\mathcal{O}(\alpha(\ell) \log \ell \log\log(\ell + 1))$ query time

Observations:

- if a block B smaller than ℓ contains the beginning i of the interval, then it does not contain the end j
- we can count the entries $C[q] = p$ in B with $p < i \leq q$ by counting
  - all the entries in B (in $\mathcal{O}(1)$ time with the multiary wavelet tree)
  - all the entries in B with $q < i$ (in $\mathcal{O}(1)$ time with the multiary wavelet tree)
  - all the entries in B with $p \geq i$

Calculation:

- if we store pointers to the wavelet-tree nodes at height k, then we use $\mathcal{O}(n)$ more bits and can count all the entries in B with $p \geq i$ in $\mathcal{O}\big(\alpha(\ell)(\log\log(\ell+1))^2\big) \subseteq o(\log \ell)$ time

source	space	time
BKM&T	$\mathcal{O}(n \log n)$	$\mathcal{O}(\log n)$
Muthu + WT	$n \log n + o(n \log n)$	$\mathcal{O}(\log n)$
GN&P	$n \log \sigma + \mathcal{O}(n \log \log n)$	$\mathcal{O}(\log n)$
this paper	$nH_0(s) + \mathcal{O}(n) + o(nH_0(s))$	$\mathcal{O}(\log \ell)$