How much memory is needed

1 Problem

How much memory is needed to construct BWT of n characters using Hayashi's method?

2 Leaf

First we analyze leaf case. The flow of the algorithm for building BWT for range $[a, b)$ is:

	bits	lifetime
build suffix array	$(b-a)\log n$	0
build BWT based on the suffix array	$(b-a)\log \sigma$	2
sample the suffix array	$S \log n$	1
build character occurrence array	$\sigma \log n$	1
build wavelet matrix	$(b-a)\log \sigma$	1

note:

- lifetime column indicates whether the bits are required:
  - (0): only temporarily; can be freed at some point during constructing the BWT of this range
  - (1): to represent the result of this range; can be freed after it is merged into its parent range
  - (2): to represent the final result; cannot be freed until the BWT of the whole range has been constructed
- S can be chosen; in practice, S will be $\in O(n/\log n)$, so that this part will take $\in O(n)$; this is a part of our augmented BWT. In theory, we should choose $S \in O(n/\log n)$, to make the space for the sampled array $\in O(n)$. In practice, we choose $S = n/64 + 2$.

3 Merge

Let's say we are merging two ranges $[a_1, b_1)$ and $[a_2, b_2)$. The overall structure of merge is this.

	bits	lifetime
build gap array	$(b_1 - a_1 + 1)(A + (1/B + 1/C)\log n)$	0
sort right samples	$S \log n$	0
scan the right BWT to fill the gap array	-	-
prefix sum the gap array	-	-
build BWT based on the suffix array (output)	$(b_2 - a_1)\log\sigma$	0
build BWT based on the suffix array (result)	$(b_2 - a_1)\log\sigma$	2
resample sampled arrays	$S \log n$	1
build character occurrence array	$\sigma \log n$	1
build wavelet matrix	$(b_2 - a_1)\log\sigma$	1

note:

- A, B, and C can be chosen.
- A is typically 8, meaning a single byte is used to maintain a non-overflowing counts in a gap array
- $1/B$ term is required to maintain overflowing counters in a gap array. it is chosen large enough to accommodate the worst-case number of overflows; that is, since the total counts put into the gap array is $(b_2 - a_2)$, and a counter overflows at $2^A - 2$[1], the overflow array must have at least $(b_2 - a_2)/(2^A - 1)$ entries. so in practice, assuming $A = 8$, we use $B = 128$. (they are currently hardcoded).
- $1/C$ term is required to maintain a prefix sum of the gap array, to quickly find the place in left to insert each right element into. We do not have a luxury of the full prefix array as it would need $\Omega(n \log n)$ memory. In theory, we should choose $C \in \Omega(\log n)$. In practice, we choose $C = 128$ (`gap_sum_gran`).

[1] we reserve $(2^A - 1)$ to indicate an overflowed entry