Compressed suffix arrays: [Grossi & Vitter – STOC 2000]

follow divide & conquer recursion of linear-time suffix tree/array construction, but 2-way not 3-way

- **base:** text $T_0 = T$
  size $n_0 = n = |T|$
  suffix array $SA_0 = SA$:
  $SA[i]$ = index in T of ith suffix of T in sorted order of suffixes of T

- **step:** text $T_{k+1} = \langle (T_k[2i], T_k[2i+1]) \text{ for } i = 0, 1, \ldots, n_k/2 \rangle$
  size $n_{k+1} = n_k/2 \Rightarrow n_k = n/2^k$
  suffix array SA_{k+1}
  = extract even entries from SA_k & divide by 2 ← compacted ← "compression"

– representing SA_k by SA_{k+1}:

– **even**-$succ_k(i) = \begin{cases} i & \text{if } SA_k[i] \text{ is even} \\ j & \text{where } \underbrace{SA_k[j]}_{\text{even}} = \underbrace{SA_k[i]}_{\text{odd}} + 1 \end{cases}$

– even-$rank_k(i)$ = # even values in $SA_k[:i]$
= # even suffixes preceding ith suffix

– $SA_k[i] = 2 \cdot SA_{k+1}[\text{even-rank}_k(\text{even-succ}_k(i))] - \{1 \text{ if } SA_k[i] \text{ is odd}\}$

- even-succ$_k(i)$: nearest even suffix ⇒ represented by SA_{k+1}
- even-rank$_k$(even-succ$_k(i)$): name of that even suffix in compacted form SA_{k+1}
- $SA_{k+1}[\text{even-rank}_k(\text{even-succ}_k(i))]$: index of that even suffix in T_{k+1}
- $2 \cdot SA_{k+1}[\text{even-rank}_k(\text{even-succ}_k(i))]$: index of that even suffix in T_k
- $2 \cdot SA_{k+1}[\text{even-rank}_k(\text{even-succ}_k(i))] - \{1 \text{ if } SA_k[i] \text{ is odd}\}$: index of ith suffix (even or odd) in T_k

– stop recursion at $d = \lg \lg n$ where $|T_d| = n_d = n/\lg n$
⇒ can afford naive $n_d \lg n_d \le n$-bit array encoding

⇒ $O(\lg \lg n)$ query to $SA[i]$ assuming even-succ & even-rank $O(1)$

– **Space:** even-rank$_k$ – $O\left(n_k \frac{\lg \lg n_k}{\lg n_k}\right)$ bits (geometric) ⇒ $O\left(n \frac{\lg \lg n}{\lg n}\right)$ total

n_k bit vector: ⟨is $SA_k[i]$ even or odd?⟩ ⇒ $2n$ total
even-succ$_k$ – $O(n)$ bits ⇒ $O(n \lg \lg n)$ bits total