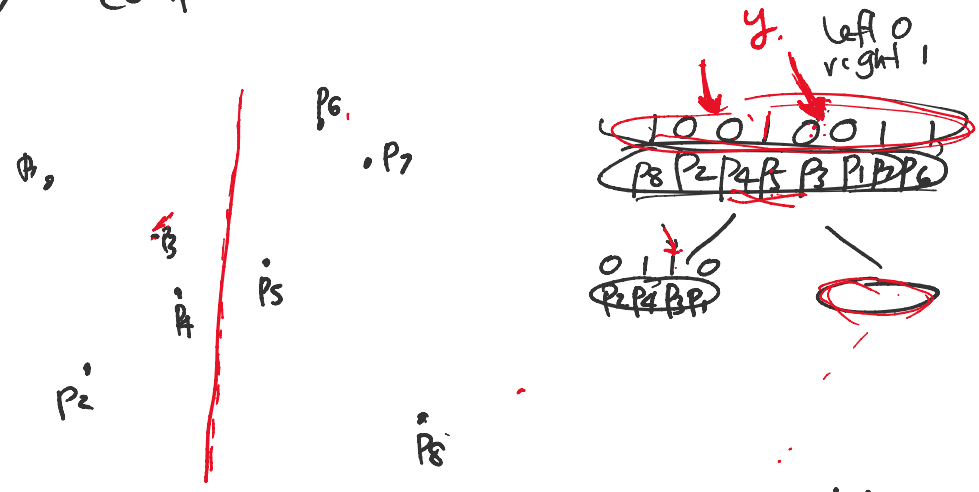
at each node,
replace y-sorted list of points/pointers
by list of bits
⇒ "compact range tree"



assumption - standard word RAM model
where each word is w-bit long
with $w \geq \log n$ (s.t. ptr/index fits in a word)

Subproblem store string s of n bits
s.t. we can answer rank queries:
given i, compute $\text{rank}_0(i)$ = # 0's in s[1..i]

1 0 1 1 0 1 1 1 0 1 1 0 1 1 0

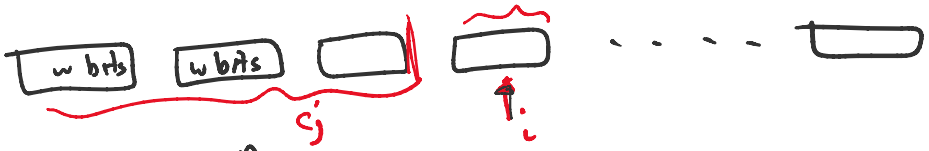
$\text{rank}_0(9) = 3$

trivial: O(n) space, O(1) time

Q: O(n) bits of space? Yes... next time...

Lemma rank query can be answered in O(1) time by a data structure using O(n) bits of space, i.e. $O(\frac{n}{w})$ words of space

Pf: divide string into $\frac{n}{w}$ words



for $j = 0, \dots, \frac{n}{w}$,
precompute c_j = # 0's in first j words
$\leftarrow O(\frac{n}{w})$ words

to compute $rank_0(i)$:
$j = \lfloor i/w \rfloor$
return c_{j-1} + (# 0's in the j^{th} word)
in $O(1)$ time
by one word op.)
nonstandard

(if $w = \log n$, word op can be simulated by table lookup with $O(2^w) = O(n)$ space) □

→ Rmk - "Succinct rank/select data structure"
↳ reduce space to $1 \cdot \frac{n}{w} + o(\frac{n}{w})$

Space: $S(n) = 2S(\frac{n}{2}) + O(\frac{n}{w})$
$\Rightarrow O(\frac{n}{w} \log n) = \boxed{O(n)}$ words
since $w \geq \log n$

Query: if y is at position i in parent list,
$\Rightarrow$ y is at position $rank_0(i)$ in left child's list
$rank_1(i)$ in right child's list

$\Rightarrow$ query time $\boxed{O(\log n)}$ for counting

Rmks - "wavelet tree"

- reporting: to "decode" each output pt, need $O(\log n)$ time by going downward to a leaf

$\Rightarrow$ $O(k \log n)$ time.

↑ improve?

A Different Method in 2D:

"Recursive Grid" (Alstrup, Brodal, Rauhe '00)

idea - $T(n) = T(n/2) + O(1) \Rightarrow O(\log n)$

→ $T(n) = T(\sqrt{n}) + O(1) \Rightarrow O(\log \log n)$

$n \to \sqrt{n} \to n^{1/4} \to n^{1/8} \to \cdots$

$n^{\frac{1}{2^k}} = \Theta(1) \Rightarrow \frac{1}{2^k} \log n = \Theta(1)$

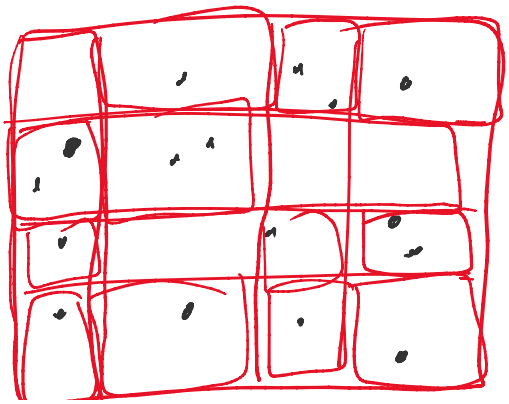
$\Rightarrow 2^k = \Theta(\log n)$

$\Rightarrow k = \Theta(\log \log n)$

- $\sqrt{n}$-way divide & conquer

Form $\sqrt{\frac{n}{c}} \times \sqrt{\frac{n}{c}}$ grid

s.t. each column has $\sqrt{cn}$ pts & each row has $\sqrt{cn}$ pts



1. recurse in each row
   & recurse in each column
2. store data structure for 3-sided queries in each row/column with $O(n)$ space & $O(\log n + k)$ time
   (by priority search tree or Cartesian tree)

3. store list L of all nonempty grid cells $|L| \le \frac{n}{c}$
   in known data structure with $O(|L| \log |L|)$ space, $O(\log |L| + k)$ query time
   $\le O(\frac{n}{c} \log n) = O(n)$
   set $c = \log n$

Analysis of space:

let $s(n)$ = space per pt

$$s(n) \le 2\, s(\sqrt{n}) + O(1)$$

"pretend" $c = 1$

$$\Rightarrow \quad s(n) \le O(2^{\log \log n}) = O(\log n)$$

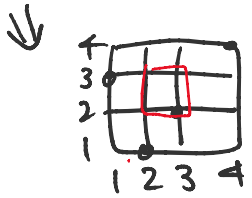
$\Rightarrow$ total space $O(n \log n)$ bad!

idea - bit packing + "reduction to rank space"

idea - bit packing + "reduction to rank space"
(replace coords with their ranks)
↑
$O(\log n)$ bits

$$s(n) \le 2\, s(\sqrt{cn}) + O\left(\frac{\log n}{w}\right)$$

pretend $c=1$



$$\Rightarrow s(n) = O\left(\frac{\log n}{w} + \frac{2\log\sqrt{n}}{w} + \frac{4\log n^{1/4}}{w} + \dots\right)$$

$$= O\left(\frac{\log n}{w} + \frac{\log n}{w} + \frac{\log n}{w} + \dots\right)$$

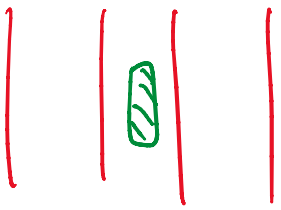
$$= O\left(\frac{\log n}{w}\log\log n\right)$$

$$= O(\log\log n)$$

$$\Rightarrow \boxed{O(n \log\log n)} \text{ space}$$

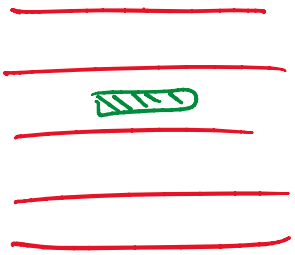
Query time: for emptiness

Case 1.



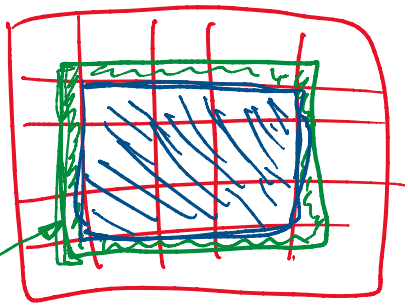
good

Case 2.



good

Case 3.



$$Q(n) \le \begin{cases} 1 \cdot Q(\sqrt{cn}) + O(\log n) & \text{if Case 1 \& 2} \\ O(\log n) & \text{if Case 3} \end{cases}$$

pred search to locate column/row

$O(\log n)$: 4 3-sided queries + 1 query in L

pretend $c = 1$ $\Rightarrow$

$$\begin{aligned} Q(n) &= O(\log n + \log \sqrt{n} + \log n^{1/4} + \dots) \\ &= O(\log n + \tfrac{1}{2}\log n + \tfrac{1}{4}\log n + \dots) \\ &= \boxed{O(\log n)} \end{aligned}$$