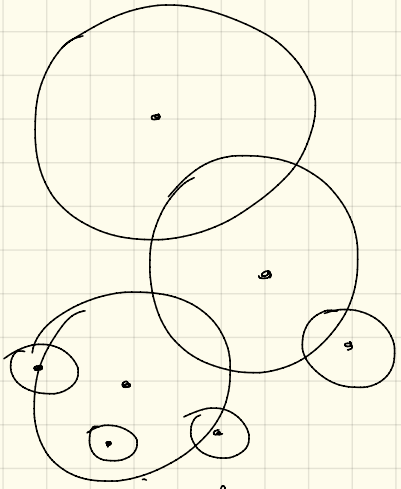
# COMP 252 - ASSIGNMENT 5

## OUTLINES OF SOLUTIONS

## Exercise 1. (Circle overlap)



Given: $(x_i, y_i, r_i)$, $1 \le i \le n$ : $\begin{cases} (x_i, y_i) \text{ center} \\ r_i \text{ radius} \end{cases}$

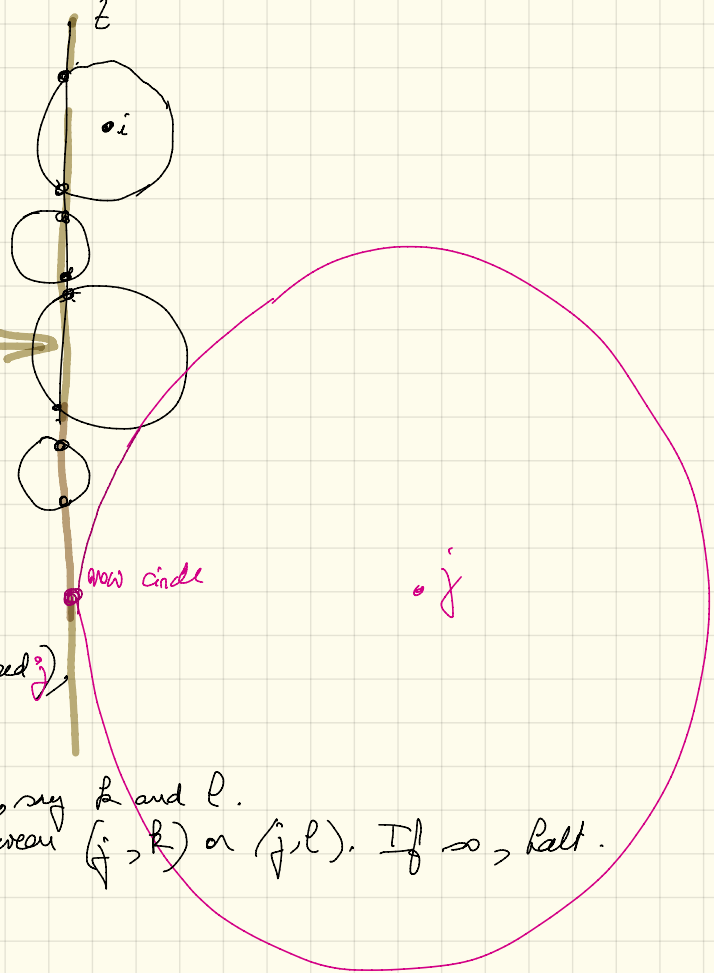
Wanted: Determine in $O(n \log n)$ time if two circles overlap.

Strategy:

We will use a sweepline algorithm but let the time corresponding to the sweepline jump only discretely, as in discrete event simulation (DES). For this we need a priority queue (PQ) with key "time", or "$x$-coordinate".

We place all pairs $(i, x_i - r_i)$ in the PQ, with key $x_i - r_i$ (birth of the $i$-th circle), and all $(i, x_i + r_i)$ for deaths. Also, a red-black tree of $y_i$'s for those $i$ that are alive is maintained.

Example:




When a new circle arrives (red),
its $y$-value (in case, $y_j$)
is added to the rd-Bst tree.
If two neighbors are determined, say $k$ and $\ell$.
We check if there is overlap between $(j,k)$ or $(j,\ell)$. If so, Fail.

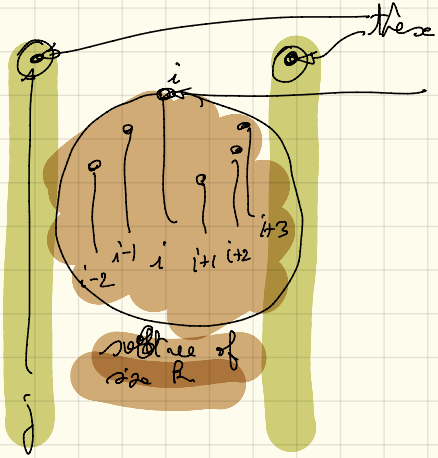
Otherwise, we do nothing.

When a circle dies, say $i$, then its two neighbors, $R$ and $L$, could possibly overlap, as they become neighbors. So, overlap between $R$ and $L$ is decided.

Check that the total time is $O(n \log n)$.

Exercise 2. (Treaps.)

P[ $(i, T_i)$ has a subtree of size $k$ ]

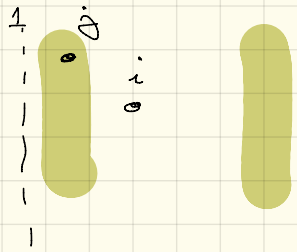



The chance of this is $\frac{2}{k(k+1)(k+2)}$, if the position of $j$ is fixed. Now $j$ has $k$ possible positions $i-k \le j \le i-1$.

So, if $i \ge k+1$, and $n+1-i \ge k+1$, so that we have no border effects, then the probability is

$$\frac{2k}{k(k+1)(k+2)} = \frac{2}{(k+1)(k+2)}.$$

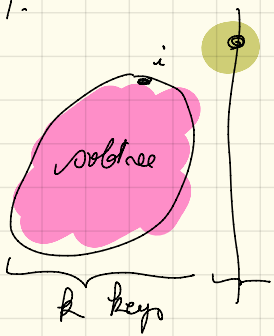
If $1 \le i \le k$, then we have two cases:




① Choose $j$ between $1$ and $i-1$ inclusive and argue as before. Then the probability is

$$\frac{2}{k(k+1)(k+2)} * (i-1).$$

② There is no $j$: We only have $\rightarrow$ Then the probability is $\frac{1}{(k+1)k}$.




The sum is

$$\frac{1}{k(k+1)} \cdot \left\{ 1 + \frac{2i-2}{k+2} \right\}$$

$$= \frac{1}{k(k+1)(k+2)} \cdot (2i+k).$$

Argue by symmetry when $1 \le n+1-i \le k$, and replacing $i$ by $n+1-i$ in the formula.

For (ii), set $i = n/3$, $j = 2n/3$. Let $(l, T_l)$ be the LCA of $i$ and $j$. The range is clearly

$$i \le l \le j.$$

Let $D_l$ = depth of the LCA. If we consider the nodes in order of $T_i$, from small to large, then

$$D_l \le \text{index of first node with } \alpha\text{-index in } [i, j].$$

Now, $(1, T_1), \ldots, (n, T_n)$ can be rewritten as $(\sigma_1, 1), \ldots, (\sigma_n, n)$, the inverse permutation, with $(\sigma_1, \ldots, \sigma_n)$ again a random permutation. So,

$$D_l \le \text{first index } k \text{ such that } \frac{n}{3} \le \sigma_k \le \frac{2n}{3}.$$

So, $P\{D_l > k\} = P\{\sigma_1, \ldots, \sigma_k \text{ do not have values in } [\frac{n}{3}, \frac{2n}{3}]\}$

$$= \frac{2n/3}{n} \times \frac{2n/3 - 1}{n-1} \times \cdots \times \frac{2n/3 - (k-1)}{n-k+1} \le \left(\frac{2}{3}\right)^k.$$

Therefore, $E\{D_l\} = \sum_{k=0}^{\infty} P\{D_l > k\} \le \sum_{k=0}^{\infty} \left(\frac{2}{3}\right)^k = \frac{1}{1 - \frac{2}{3}} = 3.$

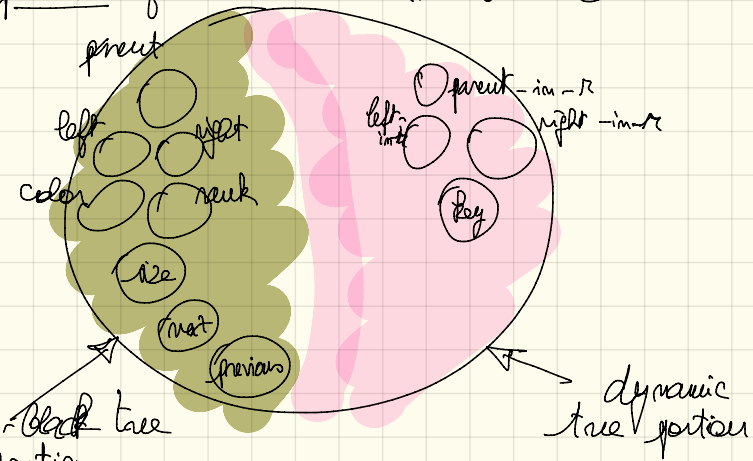
## Exercise 3. (Augmentation)

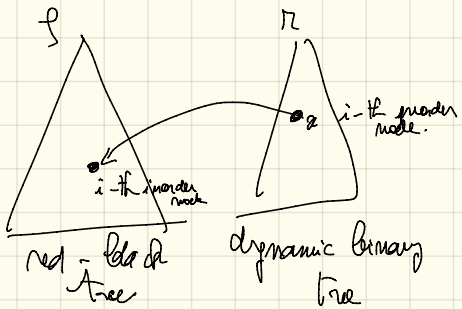
Let us maintain two trees:

(1) The **dynamic binary tree** with root $r$, storing, as usual, a key, parent, left child and right child.

(2) A **red-black tree** where each node corresponds to one node in $r$. The inorder traversal must at all points in time give the preorder of $r$. We do not store the preorder numbers, but only subtree sizes. The root of the red-black tree is $s$. Next and previous fields are also maintained

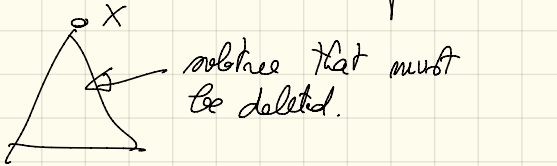
A cell thus looks like this:



**Operation 1**: Find the $i$-th preorder node. This is like SELECT$(i, \rho)$ in red-black trees, and takes time $O(\log n)$. It returns a pointer to a cell.

**Operation 2**: Replace $(i, t)$ is split into $x \leftarrow$ SELECT$(i, \rho)$, so that $x$ is the subtree that must be replaced. In $t$, we set $x$ aside




Then, via a preorder traversal of $x$, we do:

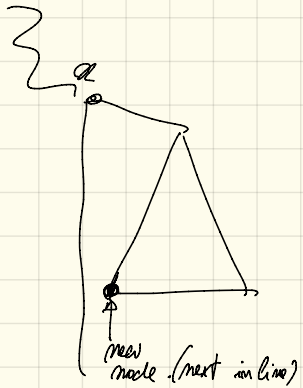
For all nodes $v$ in $x$ do:

- if $v \neq x$: delete$(v, \rho)$ [delete the node from the red-black tree] [this updates the size fields].
- if $v = x$:
  - left$[x] \leftarrow$ left$[t]$ (in $\rho$)
  - right$[x] \leftarrow$ right$[t]$
  - key$[x] \leftarrow$ key$[t]$
  - Establish all $t$-part information in the cells for all nodes of $t$.

Finally, again by preorder traversal of $t$:

$V \leftarrow x$
for all nodes $v$ of $t$ in preorder do:
  if $v \neq x$ then { insert $v$ as a next node of $V$ in $\rho$
  $V \leftarrow v$

(this adds node in "inorder" fashion in $\rho$)




red-black tree $\rho$

The time of each insert and delete in $S$ is $O(\log n)$. Now, each node ever born is inserted once and deleted at most once, and not more than $10n$ nodes are ever born.
So, the complexity is $O(n \log n)$.

The size operation is easily dealt with as well by also maintaining a size_of_subtree field in $x$.