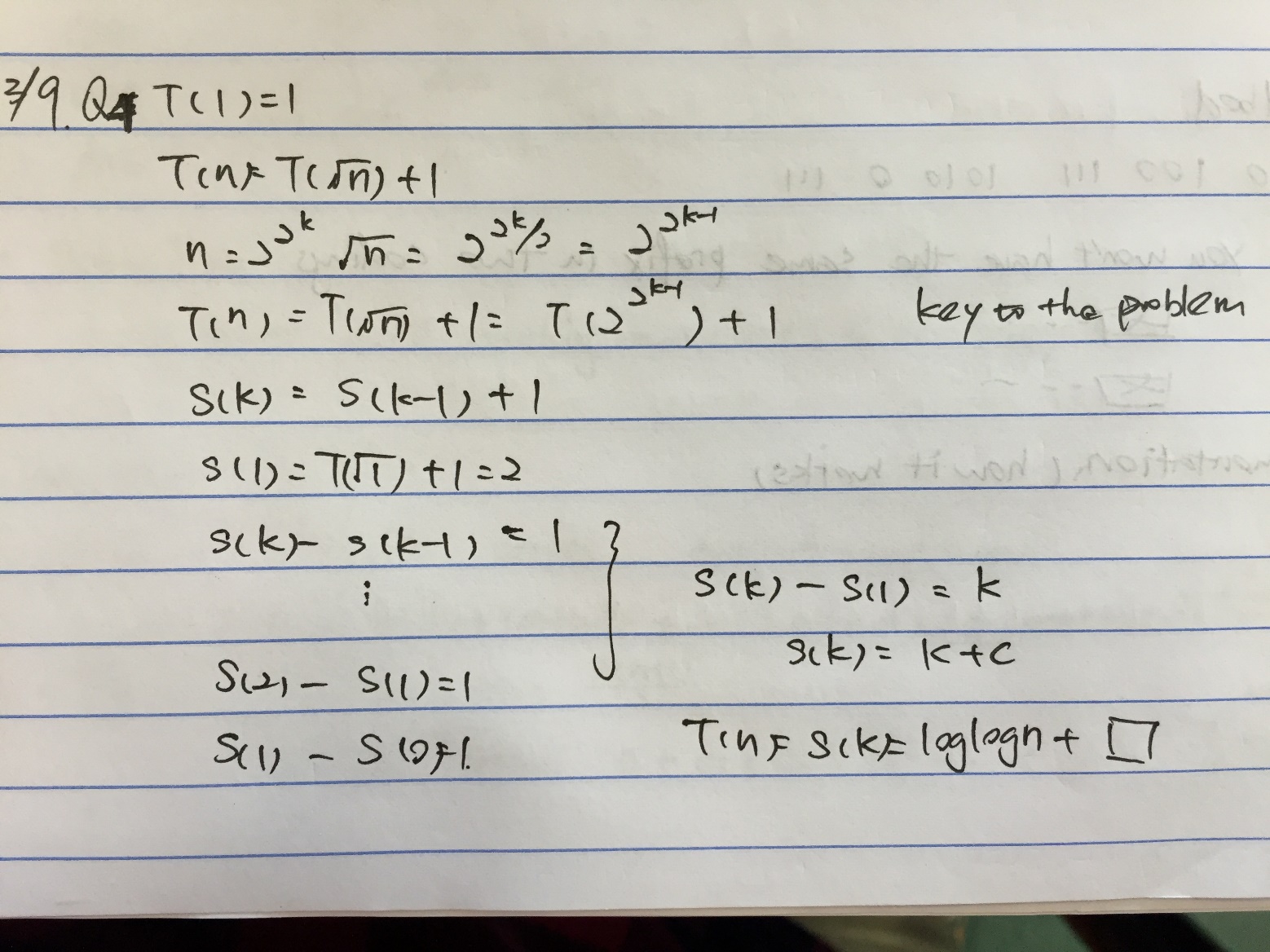
**Analysis of Algorithms – CS 700/32**

**Lecture#6 – March 9 2016**

**Notes by: Yuqian Zhang**

**Section 1: Homework**

****

**Section 2:**

**Huffman Coding**

It is one of data compression, but the best compression.

It is an example of greedy algorithm (like **Fraction Knapsack** mentioned before)

**Greedy algorithms** is where you make locally optimal choice in global optimal choice.

(Uses small grained, or local minimal/maximal choices in attempt to result in a global minimum/maximum. At each step, the algorithm makes the near choice that appears to lead toward the goal in the long-term.)

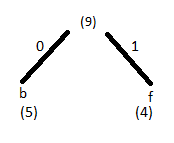
If you want to represent a letter how much space need?

Old way🡪ASCII 8 bits

But all things take 8 bits, can we do better than that?

Yes🡪find relative frequency

Example: in the text there are 100 characters, and only contains following letters

a: 10

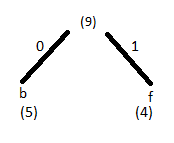
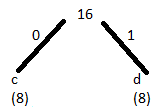
b: 5

c: 8 [take 2 smallest and combine them]🡪

d: 8

e: 25

f: 4

a:10

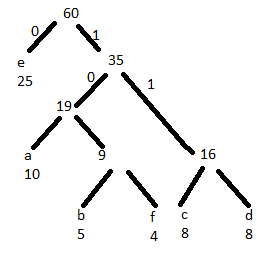
9 [continue to take 2 smallest]🡪

c:8

d:8

e:25

……finally we got:

So, if we want to encode ‘badbed’, it is:

(The length of encoding would be the depth of the leaf)

b- 1010

a- 100

d- 111

b- 1010

e- 0

d- 111

Comparing to ASCII, which using fixed 8 bits to check, Huffman Tree do not implement fixed length. When you are out of leaf, you are in the end of character. The beauty of Huffman coding is no 2 can have 2 different codings have the same prefix and 1 has something more. Thus, that allows you to decode compressed text. (About space, just take it as one of relative frequency)

**Most greedy part:** taking nest 2 smallest numbers on the list and combine them into sum and go into the bottom.

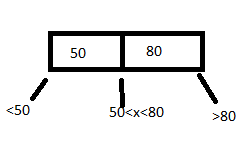
Q: How does it work (implement) in real coding?

The data structure: **heap**. (Always gets the min on the top) [O(nlgn)]

**Binary Search Tree**

* **2-3 Tree**

Hopcraft 1970 O(logn)

A 2–3 tree is a tree data structure, where every node with children (internal node) has either two children (2-node) and one data element or three children (3-nodes) and two data elements.

* **AVL**

AV= Adelson- Velsky

L = Landis 1967

Abs(height of (leftSubTree)-height of (RightSubTree))<= 1

It keep tracking the node height

* **Red-Black**

Guibas Sedgewick 1978

Not perfect balanced, but balanced enough, not breaking (lgn)

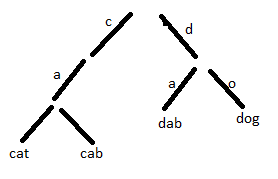
* **B-tree**

Bayer McCreight 1972

A large amount of data

* **B+ tree**

Like B- tree, but has more children

* **Trie (Tree/Try)**

R.de la Brandais 1959

Also called prefix-tree

Example: cat cab dog dab

Time O(m)

Maximum length of word

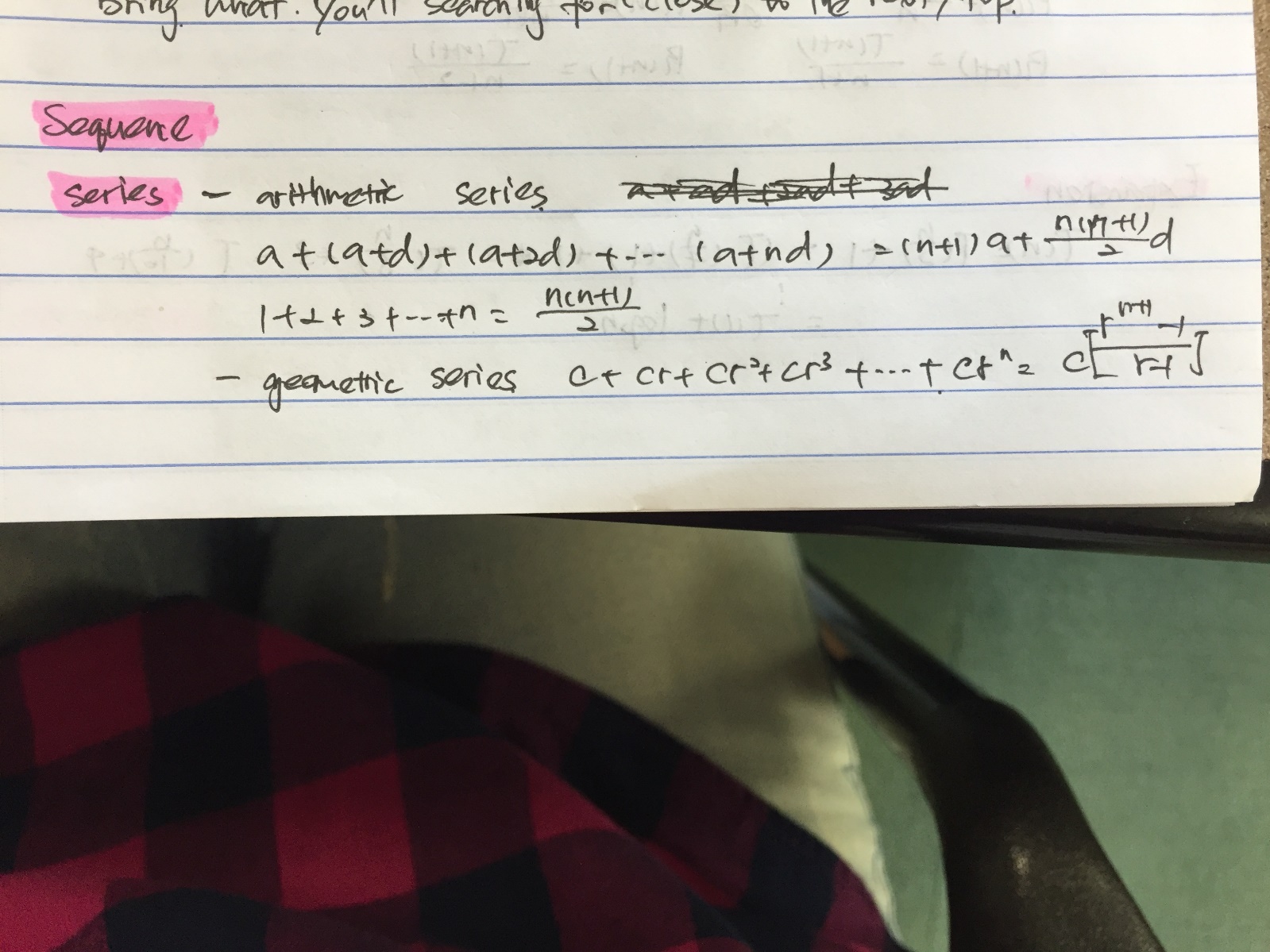
* **Splay Tree**

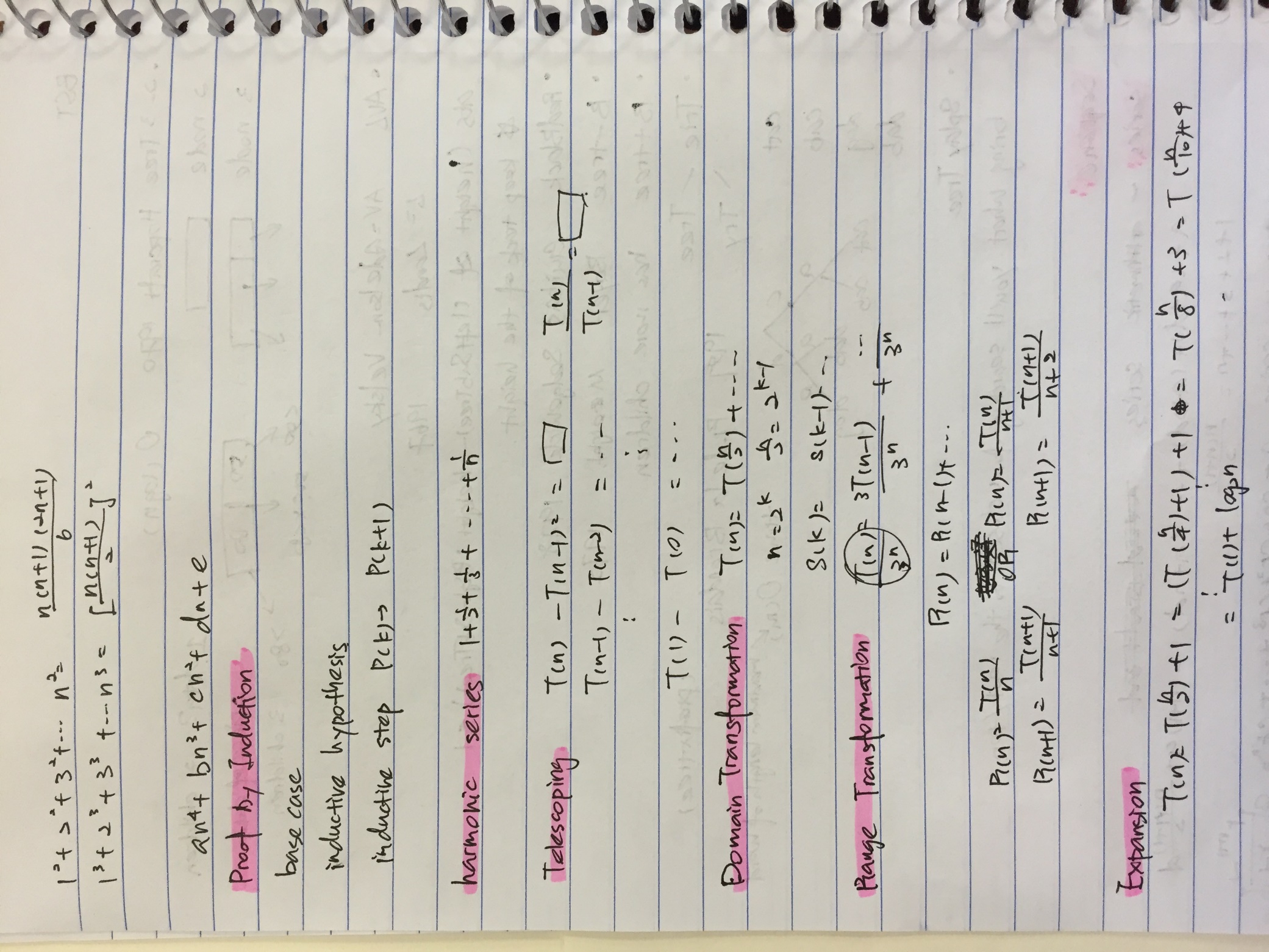
A splay tree is a self-adjusting binary search tree with the additional property that recently accessed elements are quick to access again. It performs basic operations such as insertion, look-up and removal in O(log n) amortized time.

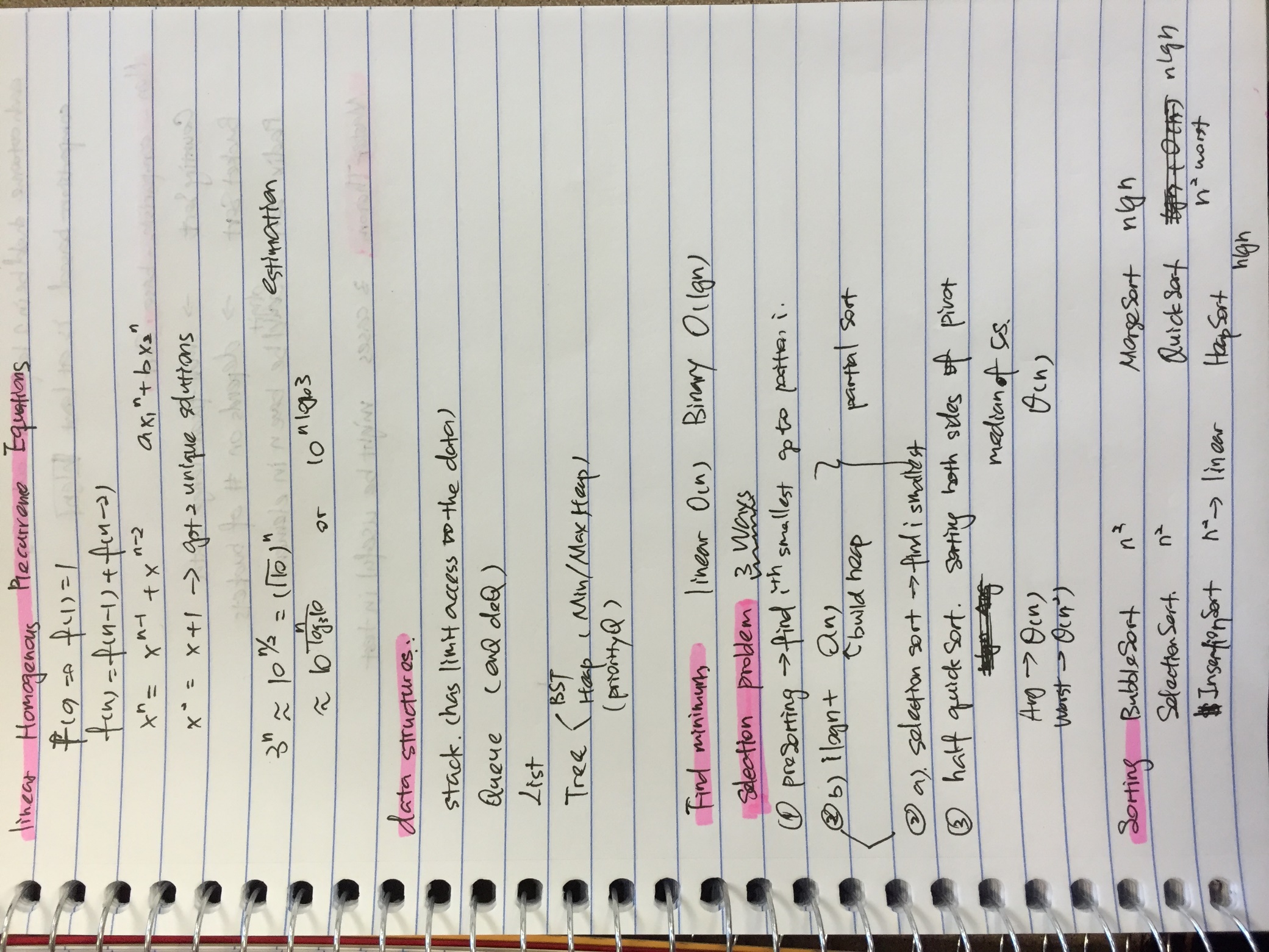
Bring what you’ll searching for (close) to the root/top

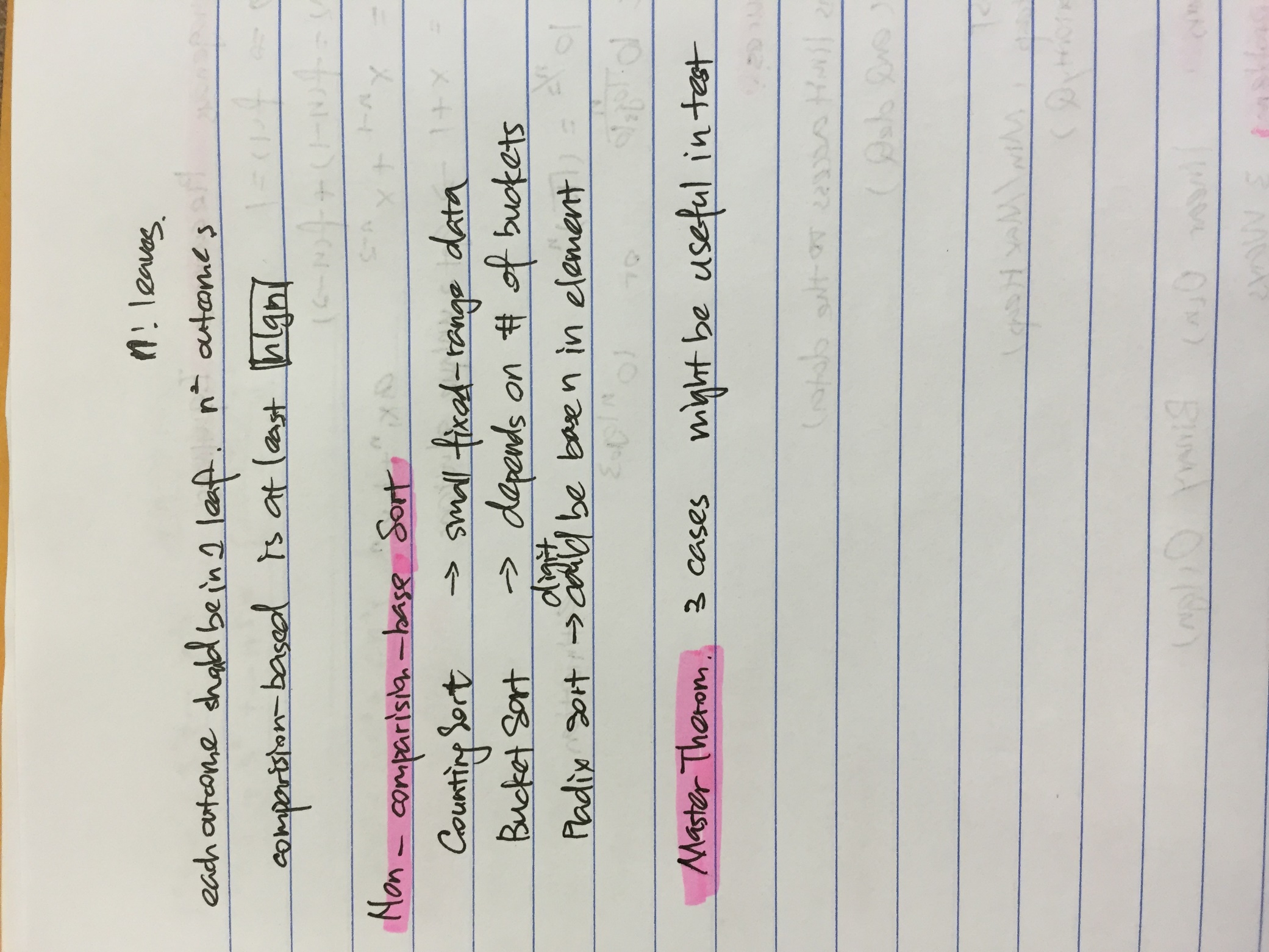
**Section 3: Review**

In Order

****

****

****

****

Each outcome should be in 1 leaf. There are n^2 outcomes cause there are n! leaves.