1. <https://leetcode.com/problems/validate-binary-search-tree/>
   1. Parcurgere in inordine O(n) si verificat daca numerele sunt in ordine crescatoare
      1. https://pastebin.com/FmMRzH5E
   2. Tinem minte pentru fiecare nod ce interval de numere poate retine (minim si maxim). Initial porninm cu -INF + INF, cand mergem in stanga mergem din (min, max) in (min, value\_radacina… care e noua limita superioara) cand mergem in stanga mergem in (value\_radacina, max). Daca parcurgerea nu gaseste contradictie atunci arborele e valid O(n)
      1. <https://pastebin.com/p6dfR1QE>
2. <https://leetcode.com/problems/balance-a-binary-search-tree/>
   1. Parcurgem in inordine obtinem numerele sortate
      1. Apoi luam mijlocul il punem radacina si apelam recursiv in stanga pe prima parte a vectorului si in dreapta pe a doua parte a vectorului
      2. 1 2 3 4 5 6 -> alegem **3**  radacina si recursiv 1 2 in stanga si 4 5 6 in dreapta
      3. <https://leetcode.com/problems/balance-a-binary-search-tree/discuss/1162834/Simple-C%2B%2B-%3A-Clean-Code>



1. <https://leetcode.com/problems/all-elements-in-two-binary-search-trees>
   1. Facem doua parcurgeri in inordine si facem interclasare O(n)
      1. https://pastebin.com/KjjhRYcS
   2. Facem doua parcurgeri in inordine concatenam si sortam O(n logn) dar usor de scris
      1. https://pastebin.com/mm2SJKBW
2. <https://leetcode.com/problems/number-of-ways-to-reorder-array-to-get-same-bst/>
   1. bonus