CS157 Lecture #13

UNION-FIND

UNION OF SETS: n sets, 1 elem each (tree structure) w/ rank 0

① determine which set an element is in by tracing up to its root (find)

② merge two sets using the following rule (union):
- * if rank(u) < rank(v), make u a child of v
- * if rank(u) = rank(v), make 1 a child of the other & increase rank by 1.

} seems to support logarithmic time → but we can do better!

FACTS: ① rank(v) ≤ rank(p(v))

② v has ≥ $2^{\text{rank}(v)}$ nodes in its tree (proof by induction on rank) (root)

③ ≤ $\frac{n}{2^k}$ elements of rank k (by 1 and 2)

every time you call find, make each node you encounter a child of the root

↳ "path-compression"

ranks no longer correspond with depth of the tree. only follows the rule:
- * if rank(u) < rank(v), make u a child of v
- * if rank(u) = rank(v), make one a child of the other and increment rank

→ analyze n operations

HOW DO WE ANALYZE THIS?

[PUNCHLINE: O(n log* n), where log* is the # times you need to take log to get to 0 or 1 (for all practical purposes, log* n ≤ 5]

* **CRUCIAL PROPERTY**: every time we process v, rank of p(v) increases [v not root, child of root]

* BUCKETS

b → $2^b - 1$

1	2,3	4,...,15	16,...,65535
log* = 0	log* = 1	log* = 2	log* = 3

* if a nodes' parent is in the next bucket

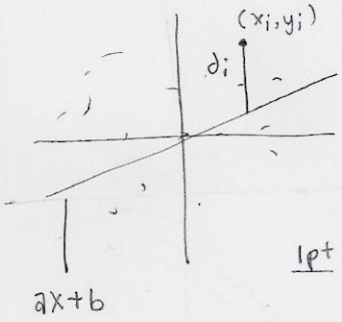
* otherwise, since the rank of its parent always increases, we can do at most $2^{b-1} - b = O(2^b)$ work

* $\sum_{i=b}^{2^b-1} \frac{n}{2^i} 2^b < \sum_{i=b}^{\infty} \frac{n}{2^i} 2^b = \frac{n}{2^{b+1}} 2^b + \frac{n}{2^{b+2}} 2^b + \cdots$

$= n + \frac{n}{2} + \frac{n}{4} + \cdots = 2^n$

↑ elem of max rank i ↑ parent upgrades

OPTIMIZATION

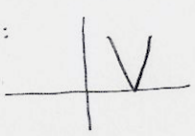


GOAL: minimize $\sum_i |d_i|$

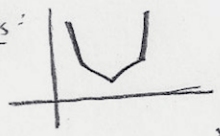
$|d_i| = |ax_i + b - y_i|$

* fix a. How do you find b?

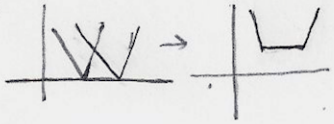
1pt:



3pts:



2 pts:



function decreases to a point and then increases

↳ binary search:
- * start at a point, and look at the point to the right. if that point has value < curr, recur on it, otherwise, work left

* what about 2 dimensions (a & b unknown)
- ↳ each col / row looks like previous case
- * choose a pt & one right & up