Amortized analysis of path compression

Observations

① $\forall u$, rank$(u) <$ rank (parent (u))

② Every root node of rank k has at least 2^k nodes in its subtree

③ There are at most $n/2^k$ nodes of rank $> k$

↳ ranks do not correspond to depths anymore, but all the statements above hold true

– rank of a node remains unchanged once it becomes a non-root node

Accounting analysis

divide the ranks into buckets as follows:

$\{1\}$, $\{2, 3, 4\}$, $\{5, 6, .., 16\}$, $\{17, 18, .., 2^{16}\}$, ...

$\{k, \ldots, 2^k\}$

How many buckets when there are n elements? } $\log^* n$

- when a node becomes a non-root and its rank lies in the bucket $\{k, .., 2^k\}$ the node pays 2^k to the bank

Accounting the cost of Find operation:

Cost of Find = # of edges from the node to the root

2 types of edges: ① Edges between nodes in the same bucket

$\log^* n$ such edges ← { ② Edges between nodes in different buckets

Observation: Each time Find(u) is performed the rank of parent(u) increases unless u is directly connected to the root

Total amount collected as credit

$\leq$ # of buckets $\times$ credit per bucket

credit-per-bucket $\leq \left(\frac{n}{2^{k+1}} + \frac{n}{2^{k+2}} + \cdots\right) 2^k$

$\leq n$

total credit $= O(n \log^* n)$

- For every find operation, account only for cost edges going across buckets $(O(\log^* n))$

  rest paid from the credit

  ↳ why is this sufficient?

  - Each time a unit is taken from u's credit rank(p(u)) increases
  - After 2^k find ops, rank(p(u)) falls in a different bucket

Cost of m Find/Union operations:

$m \log^* n + n \log^* n$