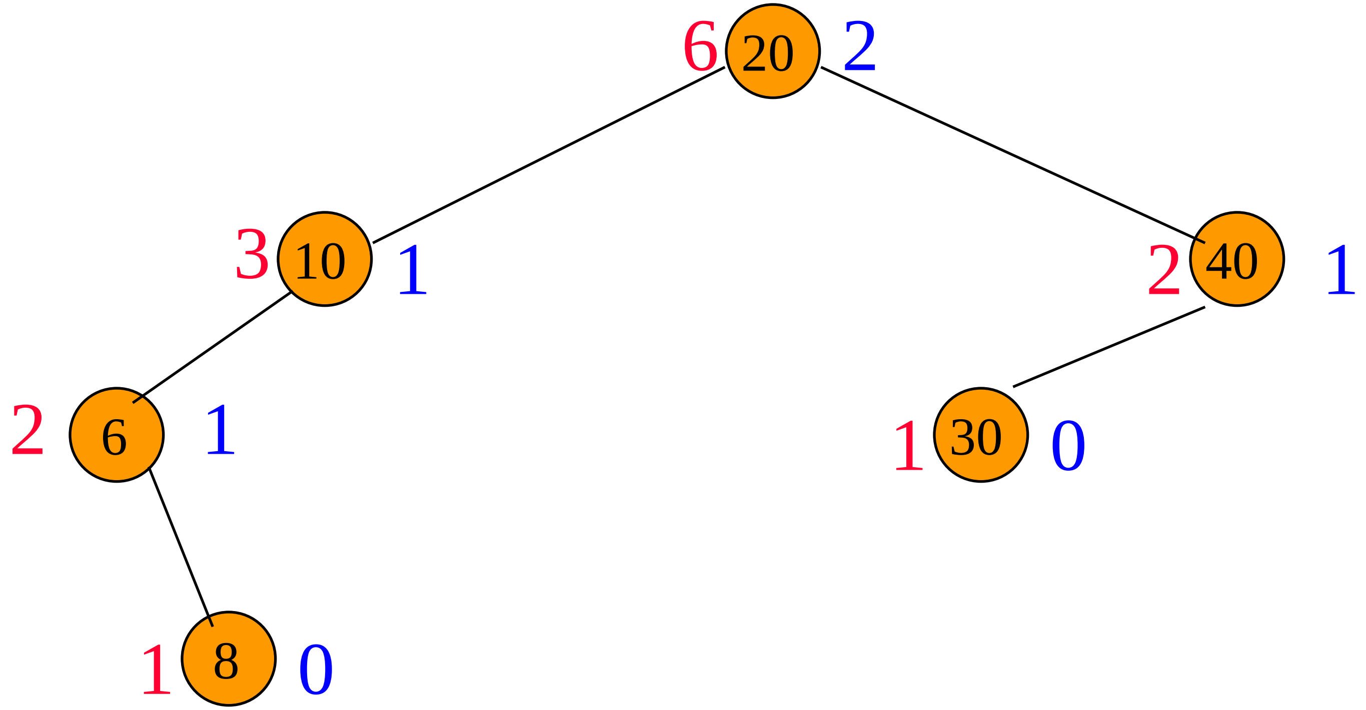
Bottom-Up Splay Trees–Analysis

- Actual and amortized complexity of join is O(1).
- Amortized complexity of search, insert, delete, and split is O(log n).
- Actual complexity of each splay tree operation is the same as that of the associated splay.
- Sufficient to show that the amortized complexity of the splay operation is O(log n).

Potential Function

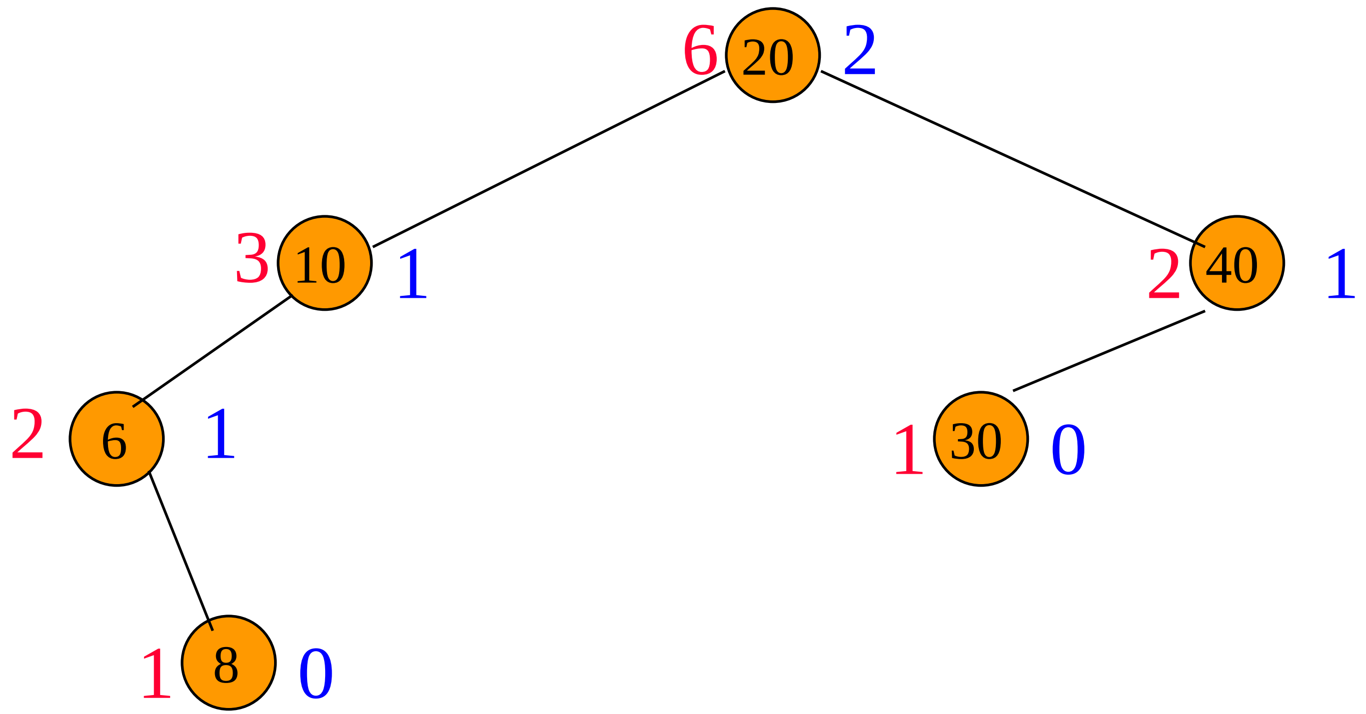
- size(x) = #nodes in subtree whose root is x.
- rank(x) = floor($\log_2$ size(x)).
- P(i) = $\Sigma_{\text{x is a tree node}}$ rank(x).
  - P(i) is potential after i'th operation.
  - size(x) and rank(x) are computed after i'th operation.
  - P(0) = 0.
- When join and split operations are done, number of splay trees > 1 at times.
  - P(i) is obtained by summing over all nodes in all trees.

Example



- size(x) is in red.
- rank(x) is in blue.
- Potential = 5.

Example



- rank(root) = floor($\log_2$ n).
- When you insert, potential may increase by floor($\log_2$ n)+1.

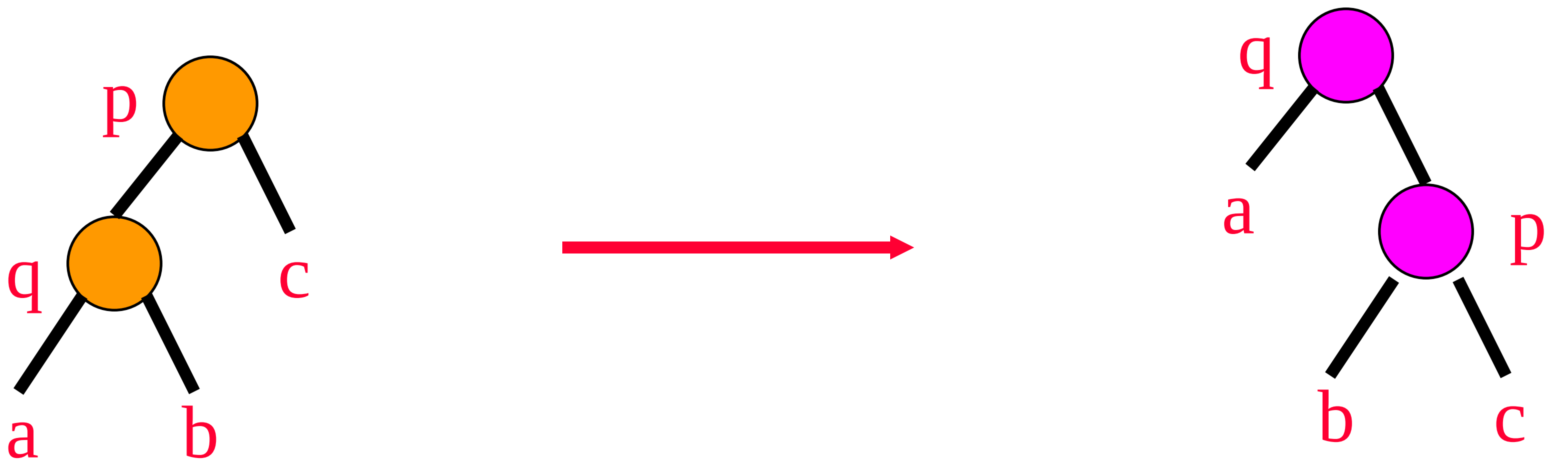
Splay Step Amortized Cost

- If q = null or q is the root, do nothing (splay is over).
- $\Delta P = 0$.
- amortized cost = actual cost + ΔP

  = 0.

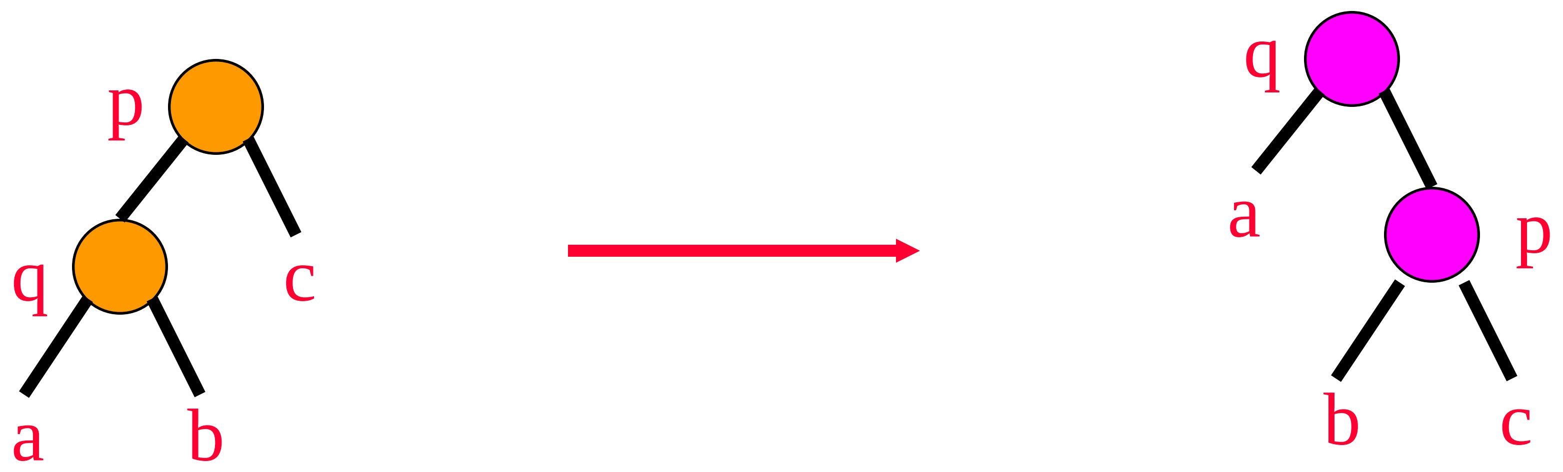
Splay Step Amortized Cost

- If q is at level 2, do a one-level move and terminate the splay operation.



- r(x) = rank of x before splay step.
- r'(x) = rank of x after splay step.

Splay Step Amortized Cost



- $\Delta P = r'(p) + r'(q) - r(p) - r(q)$

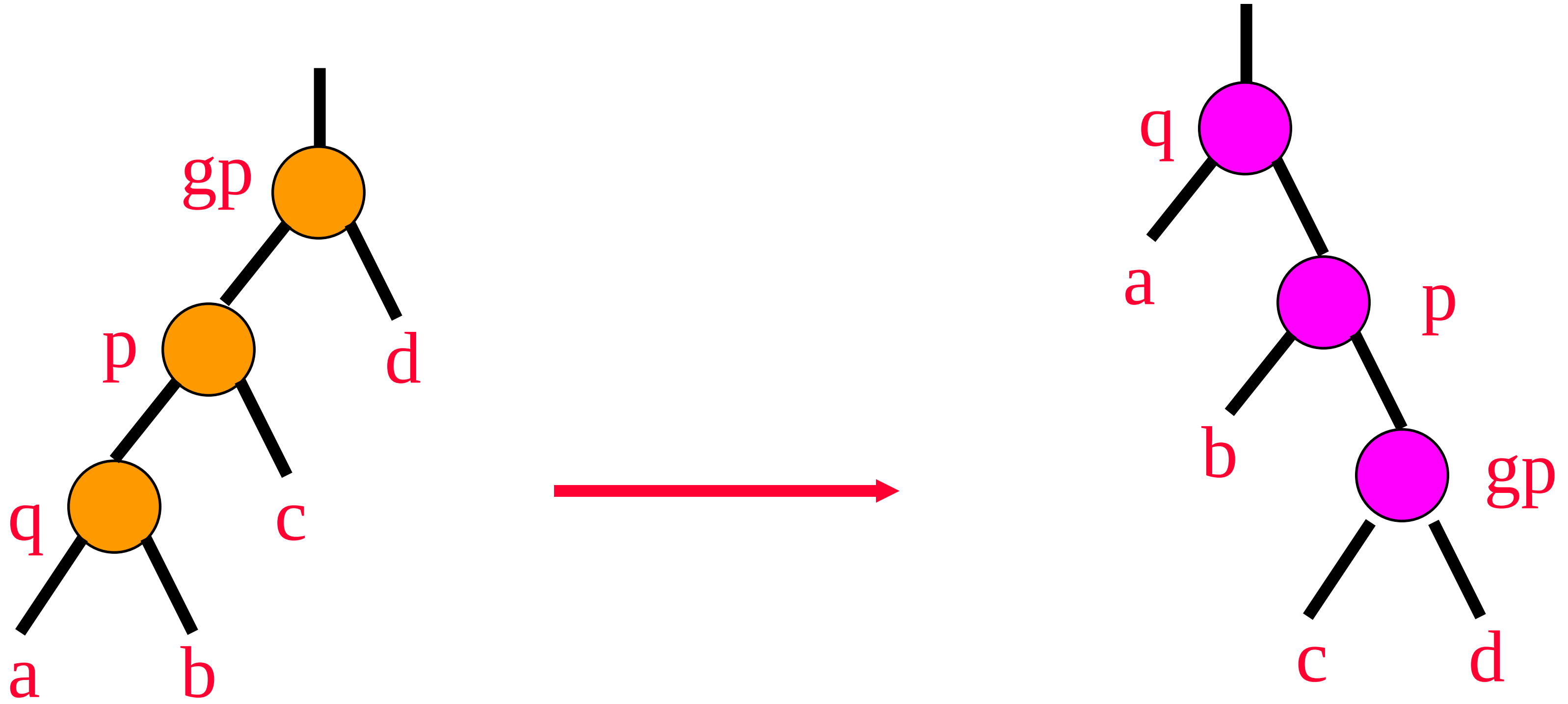
  $\quad <= r'(q) - r(q).$
- amortized cost = actual cost + ΔP

  $\quad <= 1 + r'(q) - r(q).$

2-Level Move (case 1)

- $\Delta P = r'(gp) + r'(p) + r'(q) - r(gp) - r(p) - r(q)$

2-Level Move (case 1)



- r'(q) = r(gp)
- r'(p) <= r'(q)
- r'(gp) <= r'(q)
- r (q) <= r(p)

2-Level Move (case 1)

- $\Delta P = r'(gp) + r'(p) + \cancel{r'(q)} - \cancel{r(gp)} - r(p) - r(q)$
- $r'(q) = r(gp)$
- $r'(gp) \le r'(q)$
- $r'(p) \le r'(q)$
- $r(q) \le r(p)$
- $\Delta P \le r'(q) + r'(q) - r(q) - r(q)$

$$= 2(r'(q) - r(q))$$

2-Level Move (case 1)

A more careful analysis reveals that

$\Delta P <= 3(r'(q) - r(q)) - 1$ (see text for proof)

2-Level Move (case 1)

- amortized cost = actual cost + ΔP

$$\le 1 + 3(r'(q) - r(q)) - 1$$

$$= 3(r'(q) - r(q))$$

2-Level Move (case 2)

- Similar to Case 1.

Splay Operation

- When q != null and q is not the root, zero or more 2-level splay steps followed by zero or one 1-level splay step.
- Let r''(q) be rank of q just after last 2-level splay step.
- Let r'''(q) be rank of q just after 1-level splay step.

Splay Operation

- Amortized cost of all 2-level splay steps is $<= 3(r''(q) - r(q))$
- Amortized cost of splay operation

$<= 1 + r'''(q) - r''(q) + 3(r''(q) - r(q))$

$<= 1 + 3(r'''(q) - r''(q)) + 3(r''(q) - r(q))$

$= 1 + 3(r'''(q) - r(q))$

$<= 3(\mathrm{floor}(\log_2 n) - r(q)) + 1$

Actual Cost Of Operation Sequence

- Actual cost of an n operation sequence

  $= O$(actual cost of the associated n splays).
- $actual\_cost\_splay(i) = amortized\_cost\_splay(i) - \Delta P$

  $\le 3(floor(\log_2 i) - r(q)) + 1 + P'(i) - P(i)$
- $P'(i)$ = potential just before i'th splay.
- $P(i)$ = potential just after i'th splay.
- $P'(i) \le P(i - 1) + floor(\log_2 i)$

Actual Cost Of Operation Sequence

- actual_cost_splay(i) = amortized_cost_splay(i) – ΔP

  <= 3(floor($\log_2$i) – r(q)) + 1 + P'(i) – P(i)

  <= 3 * floor($\log_2$i) + 1 + P'(i) – P(i)

  <= 4 * floor($\log_2$i) + 1 + P(i – 1) – P(i)
- P(0) = 0 and P(n) >= 0.
- Σ_i actual_cost_splay(i)

  <= 4n * floor($\log_2$n) + n + P(0) – P(n)

  <= 5n * floor($\log_2$n)

  = O(n log n)