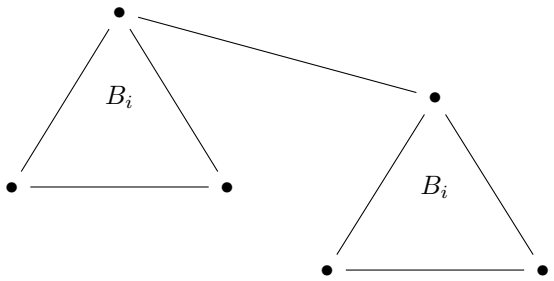
send ~~memes~~ comments to tomachello3@gmail.com

amortised cost example. you're given an n bit register, and the operation $inc : x \mapsto x + 1$. it's reasonable to say $\text{cost}(inc, x) = \#\{\text{lsb bits} = 1\}$. so at worst cost $= n$. on the other hand, $\mathbb{E}\text{cost} = \sum k/2^k$ is small. but also note the following: if you had cost $= k$ then it will take a exponential time for cost to equal k again. one costly operation for a bunch of cheap ones.

Tarjan's potential method let $\phi : S \mapsto \mathbb{Z}_{\geq 0}$ assign structures a non-negative integer. if $a = t + \Delta\phi$ where t is the time it takes to make the change ΔS. then $\sum a + \phi(S_0) \geq t_{\text{total}}$.
above example contd. let $\phi =$ total #1's in register. if cost $= k$ then $t = k$ but $\Delta\phi = 1 - k$ (or $-k$ if $k = n$) so that $a = 1$ and $t_{\text{total}} \leq \#\text{operations} + n$.

credit method give each item some dollars. if an operation costs t, it has to be payed for by the items. if we can make this work, $t_{\text{total}} \leq$starting total money.

binomial heap $B_0 = \bullet$ and $B_{i+1} = (1+r)B_i =$



in B_n we have 2^n vertices and height n. the root has n edges and there's $\binom{n}{k}$ nodes of depth k. say we have $N = \sum a_j 2^j$ keys. we keep a B_j if $a_j = 1$. it costs 1 to make two B_i's into one B_{i+1} so it costs $\log_2 N$ to meld two heaps of size at most N or insert a new key. it costs $\log_2 N$ to find min, and also to delete it (how?).

algo2 notes https://www.cs.technion.ac.il/ hamilis/Algorithms2.pdf
data structures 2 notes https://www.cs.technion.ac.il/ itai/Courses/ds2/lectures/lecture.html