CS 173 Lecture 12a: Recursively Defined Functions/Objects

10a: Prove $\forall n \geq 2$, $\sum_{i=2}^{n} i2^i = (n-1)2^{n+1}$

Pf. ⋮

I.S. $\sum_{i=2}^{n} i2^i = n2^n + \sum_{i=2}^{n-1} i2^i$

Apply I.H. to smaller version of sum.

Let $S(n) = \sum_{i=2}^{n} i2^i$.

Then $S(n) = n2^n + S(n-1)$] recursion
w/ base case $S(2) = 8$.

Recursion is the practice of defining something in terms of (a smaller version of) itself.

Joke: To understand recursion, one must first understand recursion.

→ Googling "recursion"
"Did you mean 'recursion'".

other "silly" examples of recursion:

$S(n) = \sum_{i=1}^{n} i = n + \sum_{i=1}^{n-1} i$ → $S(n) = n + S(n-1)$, $S(0) = 0$

$F(n) = n! = n(n-1)!$ → $F(n) = nF(n-1)$, $F(0) = 1$

Closed Form: rewriting of the function w/ no summations or recursive calls.

$S(n) = n + S(n-1)$, $S(0) = 0$ → $S(n) = \frac{n(n+1)}{2}$.

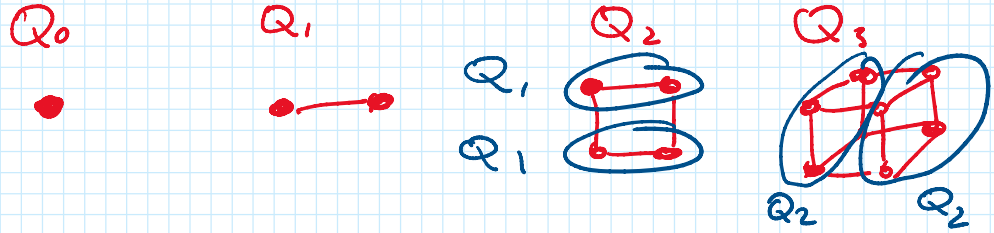
Applications of recursion: all over CS → see CS 374.

Recursive Algorithms like mergesort, binary search intimately tied to question of what can a computer

intimately tied to question of what can a computer do?

Recursively defined object:

Q_n : n-dimensional hypercube.



Q_n is obtained by taking two copies of Q_{n-1} and putting an edge between copies of the same vertex.

→ Recursively defined object

→ Recursively defined properties.

V_n = set of vertices in Q_n

E_n = set of edges in Q_n.

$|V_n| = 2|V_{n-1}|$, $|V_0| = 1$ ⇝ $v(n) = 2v(n-1)$, $v(0) = 1$.

$|E_n| = 2|E_{n-1}| + |V_{n-1}|$ → $v(n) = 2^n$

$e(n) = 2e(n-1) + 2^n$, $e(0) = 0$,

$e(n) = ?$