Practice Question: Induction

- Suppose you have an unlimited supply of 3 and 7 cent coins, prove by induction that you can make any amount $n \geq 12$.

BASE: $H(12) =$ 4 3 cent coins ✓

$H(13) =$ 2 3 cent coins + 1 7 cent coin ✓

$H(14) =$ 2 7 cent coins ✓

Ind: $H(12) \wedge \ldots \wedge H(K-1) \rightarrow H(K)$ $(K > 14)$ important

Consider $K-3$. By the IH, $H(K-3)$ is true.

Thus, we can take the solution for $K-3$ and add one 3 cent coin. $\square$

Practice Question: Asymptotics

- Put these functions in order so that $f_i = O(f_{i+1})$
  - f_2 $n^{\log_2 7}$ $\rightarrow n^{2.x}$ $\left(\log_2 4 = 2 < \overline{\log_2 7} < \log_2 8 = 3\right)$
  - f_3 $8^{\log_2 n}$ $\rightarrow 2^{\log_2 8 \cdot \log_2 n} \rightarrow n^3$
  - f_4 $2^{3.1 \log_2 n}$ $\rightarrow 2^{(\log_2 n) \cdot 3.1} \rightarrow n^{3.1}$
  - f_6 $2^{(\log_2 n)^2}$ $\rightarrow \underbrace{2^{\log_2 n}}\,^{\cdot \log_2 n} \rightarrow n^{\log_2 n}$
  - f_5 $\underline{n^2 \sum_{i=1}^{n} i}$ $\rightarrow n^2 \cdot c \cdot n^2 \rightarrow \cancel{c} \cdot n^4$
  - f_1 $n^2 \log_2 n$ $\rightarrow n^2 \cdot \log n$

(See HW1 for more examples)

$n^{2.x}$ vs. $\cancel{n} \log n$

$\underbrace{n^{0.x}}$ vs. $\log n$

Practice Question: Asymptotics

- Suppose $f_1(n) = O(g(n))$ and $\cancel{f_2}$ $f_2(n)$ $= O(g(n))$. Prove that $f_1(n) + 4f_2(n) = O(g(n))$.

$$f_1(n) \le c_1\, g(n) \quad \forall n \ge n_0^1$$

$$f_2(n) \le c_2\, g(n) \quad \forall n \ge n_0^2$$

$$\hookrightarrow 4f_2(n) \le 4c_2\, g(n) \quad \forall n \ge n_0^2$$

$$f_1(n) + 4f_2(n) \le c_1\, g(n) + 4c_2\, g(n) = (c_1 + 4c_2)(g(n))$$

$$c = c_1 + 4c_2\,, \quad n_0 = \max(n_0^1, n_0^2)$$

Practice Question: Recurrences

```
F(n):  if n < 1 → return
  For i = 1,…,n²: Print "Hi"
  For i = 1,…,3: F(n/3)
```

- Write a recurrence for the running time of this algorithm. ✓
  Write the asymptotic running time given by the recurrence. ✓

$T(1) = c$

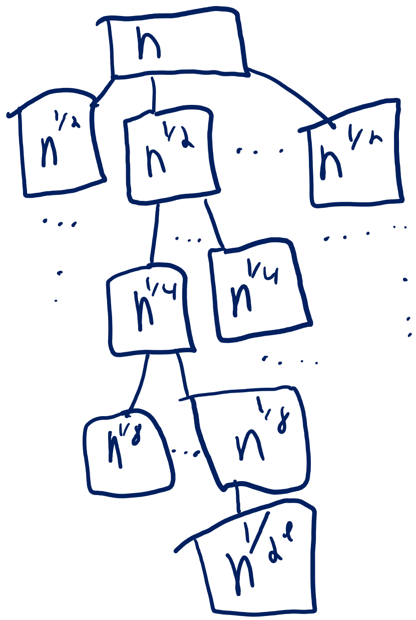
$T(n) = 3T(n/3) + n^2$

$a = 3 \quad b = 3 \quad d = 2$

$3/3^2 < 1 \quad \Theta(n^2)$

Topics: Recurrence Trees

- Consder the recurrence $T(n) = \sqrt{n} \cdot T(\sqrt{n}) + n$ with $T(2) = 1$. Solve using a recursion tree.



level	#	work
0	1	n
1	$n^{1/2}$	$n^{1/2} \cdot n^{1/2} = n$
2	$n^{1/2} \cdot n^{1/4}$	$n^{3/4} \cdot n^{1/4} = n$
3	$n^{3/4} \cdot n^{1/8}$	$n^{7/8} \cdot n^{1/8} = n$
ℓ	$n^{\frac{2^\ell - 1}{2^\ell}}$	$n^{\frac{2^\ell-1}{2^\ell}} \cdot n^{1/2^\ell} = n^{2^\ell/2^\ell} = n$

$$\sum_{i=0}^{\log \log n} n = (\log(\log(n)) + 1) n = \Theta(n \log \log(n))$$

$\square \leftarrow 2 = n^{1/2^\ell} \rightarrow \log_2 2 = \frac{1}{2^\ell} \log_2 n$

$2^\ell = \log_2 n$

$\ell = \log_2 \log_2 n$

Practice Question: Stable Matching

- Give an example of 3 doctors and 3 hospitals such that there exists a stable matching in which every hospital gets it last choice of doctor

	1st	2nd	3rd
h_1			d_1
h_2			d_1
h_3			d_2

	1st	2nd	3rd
d_1	h_2		
d_2	h_3		
d_3	h_1		

Consider: (d_1, h_2), (d_2, h_3), (d_3, h_1)

Stable because no doctor could be happier!

Practice Question: Divide-and-Conquer

You are babysitting your niece and before she will go to bed she insists on playing the following guessing game:

1. She picks a number x in $1, 2, \ldots, n$.
2. You make a guess y_1, and she simply says *correct* or *incorrect*.
3. You make a sequence of guesses $y_2, y_3, \ldots$. If your guess $y_i = x$ then your neice says *correct* and goes to bed. If your guess y_i is closer to x than the previous guess y_{i-1}, then she says *warmer* and if y_{i+1} is farther than the previous guess, then she says *colder*.

Your goal is to find x with as few guesses as possible so that your niece will go to bed. Design a divide-and-conquer algorithm that guesses your niece's number using $O(\log n)$ guesses.[1]

- Two versions:

  1. If your guess y_i is the same distance as guess y_{i-1}, your niece stays up forever and you lose

  2. If your guess y_i is the same distance as guess y_{i-1}, your niece says "same" and you win

Practice Question: Divide-and-Conquer

$\ell \downarrow \quad r \downarrow$

$1, \ 2, \ 3, \ \ldots, \ n-1, \ n$

- Describe your algorithm in pseudocode

Game(n)
  ~~Guess~~ FindX$(1, n)$

FindX(ℓ, r)
  guess r
  if "correct" $\to 2^{2^2}$
  guess ℓ
  if "correct" $\to 2^{2^2}$
  else if "same" $\to$ guess $\ell + \frac{r-\ell}{2} \to$ "correct" 2^{2^2}
  else if "colder" $\to$ FindX$\left(\ell + \left\lfloor \frac{r-\ell}{2} \right\rfloor, r\right)$
  else $\to$ FindX$\left(\ell, \ell + \left\lceil \frac{r-\ell}{2} \right\rceil\right)$

Practice Question: Divide-and-Conquer

- Analyze your algorithm's running time by writing a recurrence

$$T(1) = C$$

$$T(n) = T(n/2) + C$$

$a = 1 \quad b = 2 \quad d = 0$

$$O(n^0 \log n = \log n)$$

Practice Question: Divide-and-Conquer

- Prove by induction that the algorithm is correct

$H(n)$: our alg. finds x when there are n numbers in the range of possibilities

Base: $H(1) \rightarrow l = r \rightarrow x = l$, we guess l ✓

$H(2) \rightarrow l = r-1 \rightarrow x = l$ or $x = r$, we guess l and r ✓

Ind: $H(1) \wedge H(2) \wedge \ldots \wedge H(k-1) \rightarrow H(k)$

if $x = l \rightarrow$ we win ✓, if $x = r \rightarrow$ we win ✓

if x is equidistant from l and r $\rightarrow$ we win ✓

Otherwise we recurse $\leq k/2 + 1$ elements

$k/2 + 1 < k$ $(\forall k > 2)$ $\therefore$ by IH, the rec. call correctly finds x $\rightarrow 2^{2^2}$ □

Practice Question: Alg. Design

- Design an $O(n)$-time algorithm that takes an array $A[1:n]$ and returns a sorted array containing the smallest $\sqrt{n}$ elements of A

① $v = \text{Select}(A, \sqrt{n})$ $\quad O(n)$

② Partition around v $\quad O(n)$

↳ L = smallest $\sqrt{n}$ elems,

③ Mergesort(L)

$c \cdot \sqrt{n} \log(\sqrt{n}) = c \cdot n^{1/2} \cdot \frac{1}{2} \cdot \log(n)$

$n^{1/2} \log n < n^{1/2} \cdot n^{1/2}$

$= O(n)$

Practice Question: Recurrence Analysis

- Consider the following sorting algorithm

```
A[1:n] is a global array
SillySort(1,n):
  if (n <= 3): put A in order
  else:
   SillySort(1,2n/3)
   SillySort(n/3,n)
   SillySort(1,2n/3)
```

- Write a recurrence for the running time of this algorithm. $T(1) = c, \; T(n) = 3T(2n/3) + c$
- Write the asymptotic running time given by the recurrence. $a = 3 \quad b = 3/2 \quad d = 0$

$3/1.5^0 > 1 \rightarrow \Theta(n^{\log_{1.5} 3} = n^{2.71})$