S. KIMMEL

Probability Review

All outcomes equally likely – unless you are really comfortable with probability – unlearn what you have learned!

$$P(E) = \frac{|E|}{|S|}$$

$|E|$ ← counting problem 1

$|S|$ ← counting problem 2

$|S|$ = Ways of choosing set of 10 out of set of 72, eleven times (use product rule)

$$\binom{72}{10}^{11}$$

$|E_3|$ = First choose set of 3 out of set of 11 possible weeks to test: $\binom{11}{3}$

Next: • For week where tested, number of ways of choosing other players $\binom{71}{9}$

- 71: size of set not including Eric
- 9: size of other players to choose

S. KIMMEL

Next: • For week not tested, number of ways of choosing players

$\binom{71}{10}$ — size of set not including Eric; size of other players to choose

1st week of testing: $\binom{71}{9}$ ways

2nd " $\binom{71}{9}$ ways $\Rightarrow \binom{71}{9}^3$ ways

3rd " $\binom{71}{9}$ ways

1st week of no testing $\binom{71}{10}$ ways

$\Rightarrow \binom{71}{10}^8$

⋮

8th week of no testing $\binom{71}{10}$ ways

Total: $\binom{11}{3}\binom{71}{10}^8\binom{71}{9}^3$

S. KIMMEL

Equivalence Relation + Equivalence Classes

$R \subseteq S \times S$

↑ pairs of elements

$C \subseteq S$

↑ No Pairs!

$R = \{(0,0),(0,1),(1,0),(1,1),$
$(0,2),(2,0),(1,2),(2,1)$
$(2,2),(3,3),(3,4),$
$(4,3),(4,4),(5,5)\}$

$C_1 = \{0,1,2\}$
$C_2 = \{3,4\}$
$C_3 = \{5\}$

$(0,1) \Rightarrow$ "0 is equivalent to 1"
$(1,2) \Rightarrow$ "1 is equivalent to 2"
} 0, 1, 2 all equivalent to each other

$S = \{f : f \text{ is a function } f: \mathbb{R} \to \mathbb{R}\}$

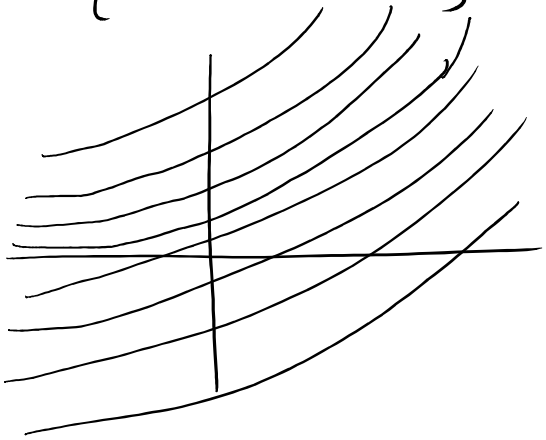
$R = \{(f,g) : \exists C \in \mathbb{Z} : \forall x \in \mathbb{R}, f(x) - g(x) = C\}$

What is an equivalence class?

S.KIMMEL

ex: $\{e^x, e^x+1, e^x-1, e^x+2, e^x-2 \cdots\}$

$\{e^x + C : C \in \mathbb{Z}\}$



S. KIMMEL

Tree Method: use trees to solve recurrence relations

max(A)

Input: Array A of length n

Output: Max value in array

```
if (length of A is 1) return A[1]
for i = 1 to n, do nothing
max1 = max(1st half of A)
max2 = max(2nd half of A)
return maximum{max1, max2}
```

Recurrence Relation for Time Complexity:

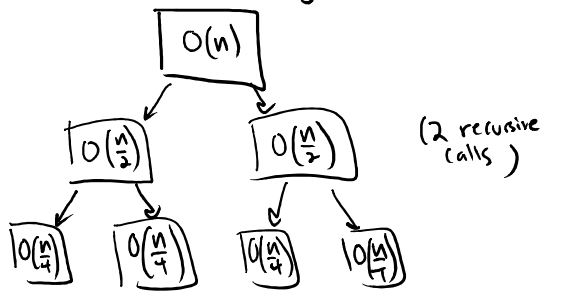
$T(n)$ = runtime on length n array

- Base case: $T(1) = O(1)$
- Recurrence: $T(n) = O(n) + 2T\left(\frac{n}{2}\right)$

$O(n)$: work done here and now

$2T\left(\frac{n}{2}\right)$: # of recursive calls; $T\left(\frac{n}{2}\right)$: runtime of recursive call

In box: amount of work done at this call not including work done by recursive calls



Size of input	
n	$T(n) = O(n) + 2T\left(\frac{n}{2}\right)$
$n/2$	$T\left(\frac{n}{2}\right) = O\left(\frac{n}{2}\right) + 2T\left(\frac{n}{4}\right)$
$n/4$	$T\left(\frac{n}{4}\right) = O\left(\frac{n}{4}\right) + 2T\left(\frac{n}{8}\right)$
⋮	⋮
1	

Idea: count all work done in all boxes... that will be all the work.