CS 515

Amir Nayyeri
(OH, after class until 4pm)

Will Maxwell
(Mon, 2pm)

---

canvas, slack

Prereq: CS 325 (or equiv)
DP, Recursion, Proof
RT, data structures,
graphs.

This is NOT a first course
in alg. [CS 325 or CS 599]

Text book: Jeff Erickson

Evaluation

HW (40%):
all written
design/analysis
pf of correctness
groups of pref 3
people.

Midterm (30%) } closed
final (30%) } book

Academic Integrity:

H/W:

(1) cite whatever source you use

(2) never copy/paste.

Exams:

closed book.

IDK (I don't know) Rule:

you can write IDK for any prob in HW or exam and recieve 25% of the points.

Topics

Recursion

Dynamic Prog

Divide and Conquere

Greedy Alg*

Randomized Alg

Basic Randomization

quick sort

Random ds

Hashing ...

Network flows

Linear Prog Alg

Apx Alg*

Recursion

Idea:

to solve problem X on input of size n

Base Case
(1) if n is 'small' solve it directly

(2) otherwise, reduce the problem to smaller size problem of same type.

Pf: by induction:

(1) BC are correct.
(2) IH: Alg is correct for $n < k$
(3) IS: Alg is correct for k

```
max(A[1..n]): # Assume n ≥ 1
    if n = 1:
        return A[1]
    else
        maxOfTwo(
            max(A[1,...,n-1]),
            A[n]
        )
```

$T(n)$: ruing time of 'max' for n numbers.

$T(1) = O(1)$

$T(n) = T(n-1) + O(1)$

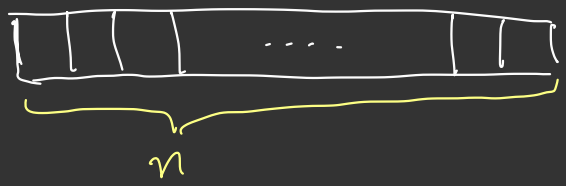
$= (T(n-2) + O(1)) + O(1)$

$\vdots$

$= \underbrace{O(1) + \cdots + O(1)}_{n \text{ times}} = O(n)$

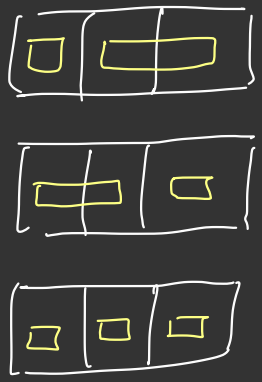
EX:

In how many diff ways I can tile



with □ and □□

$n=1$ there is 1 way

$n=3$  there are 3 ways.

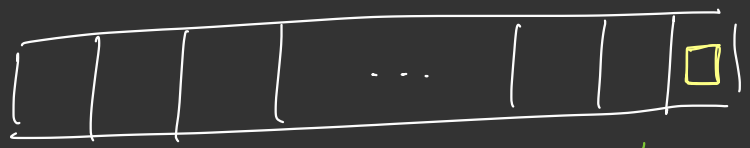
$T(n)$: RT of Count(n)

$T(1) = O(1)$, $T(2) = O(1)$

$T(n) = T(n-1) + T(n-2) + O(1)$

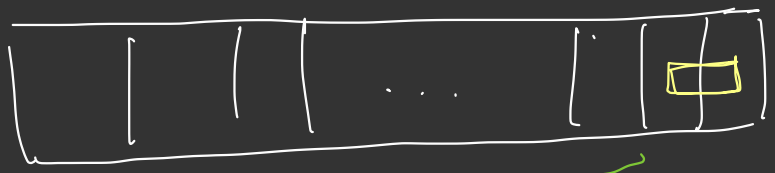
---

①



# of ways to tile $1 \times (n-1)$ grid

②



# of ways to tile $1 \times (n-2)$ grid

```
Count(n):
    if n = 1 return 1
    if n = 2 return 2
    return Count(n-1) + Count(n-2)
```

$T(n)$: RT of Count(n)

$T(1) = O(1)$, $T(2) = O(1)$

$T(n) = T(n-1) + T(n-2) + O(1)$

$$T(n) \leq T(n-1) + T(n-2) + O(1)$$
$$\leq 2T(n-1) + O(1)$$
$$\vdots$$
$$= O(2^n)$$

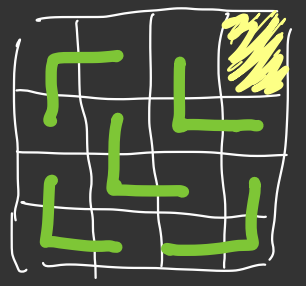
$$T(n) \geq 2T(n-2) + O(1)$$
$$\geq 2T(n-2)$$
$$\geq 2(2T(n-4))$$
$$\geq 2(2(2T(n-6)))$$
$$= O(2^{n/2})$$

* Tile a $2^n \times 2^n$ grid with one missing square with

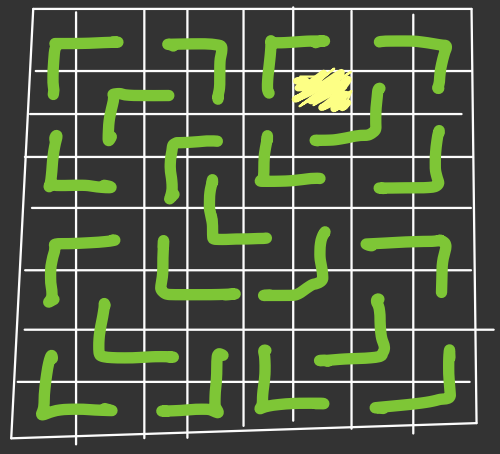
Ex: $n = 0$

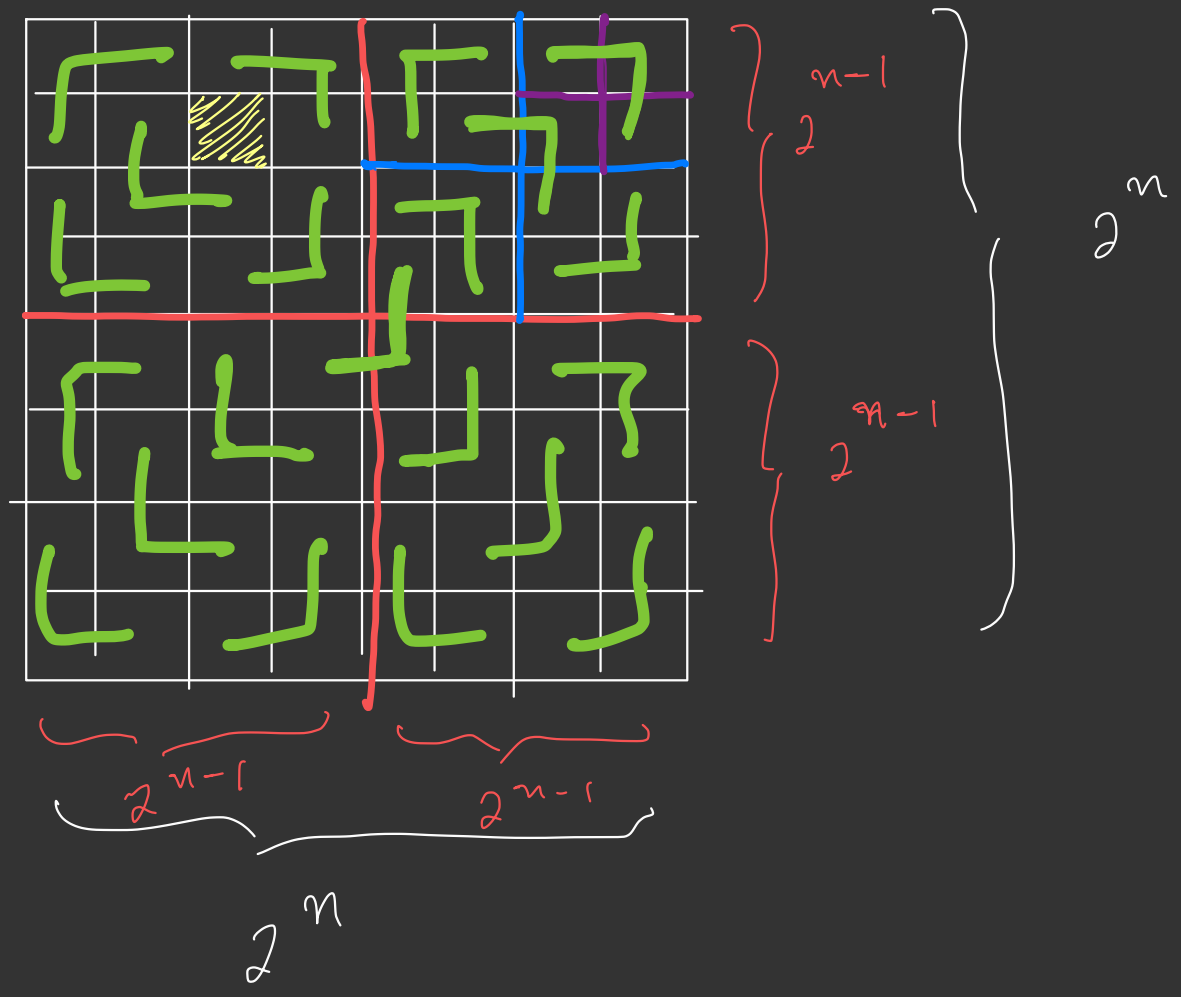
$n = 1$

$n = 2$
missing
$(1,4)$



$n = 3$
missing
$(2,6)$





Alg:

#BC:

if $n=0$ do nothing

otherwise

* divide the grid into four $2^{n-1} \times 2^{n-1}$ grids.
* place a ⊞ to intersect those $2^{n-1} \times 2^{n-1}$ grids that don't contain the missing square
* recursively tile the rest