Recursive Alg:

- Small prob: solve directly
- Otherwise, reduce the prob into smaller instances.

Ex: Compute max:

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

Ex: Sorting:

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

Ex: Comput # of bit strings with no '11'.

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

Announcements:

- PA1 is due next Mon
- GA1 is posted.

Recursive Alg:

- Small prob: solve directly BC
- Otherwise, reduce the prob into smaller instances.

Ex: Compute max:

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

Ex: Sorting:

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

Ex: Comput # of bit strings with no '11'.

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

Pf by induction

Prove $P(n)$:

- if n is small, prove directly. BC
- ASSUME that $P(k)$ is true for $k < N$
- Prove that $P(N)$ holds

Induction step (IS)

Induction Hypothesis (IH)

Pf by induction

Prove $P(n)$:

- if n is small, prove directly.
- ASSUME that $P(k)$ is true for $k < N$
- Prove that $P(N)$ holds

Base Case (BC)

Induction Hypothesis (IH)

Induction step (IS)

Recursive Alg:

- Small prob: solve directly
- Otherwise, reduce the prob into smaller instances.

BC

Ex:

Show $1+2+\cdots+n = \dfrac{n(n+1)}{2}$, for all $n \geq 1$

BC: $n=1$: $1 = \dfrac{(1)(1+1)}{2} = 1$ ✓

IH: for any $k < N$:

$$1+2+\cdots+k = \frac{k(k+1)}{2}$$

IS: to show:

$$\underbrace{1+2+\cdots+(N-1)}+N = \frac{N(N+1)}{2}$$

$$1+2+\cdots+(N-1)+N \underset{\text{IH}}{=} \frac{(N-1)N}{2}+N$$

$$= N\left(\frac{N-1}{2}+\frac{2}{2}\right) = N\left(\frac{N+1}{2}\right)$$

Ex: Prove that for any $n > 23$, n can be written as a sum of 5's and 7's

BC: $k = 24, 25, 26, 27, 28$

$k = 24 = 2 \times 5 + 2 \times 7$

$k = 25 = 5 \times 5$

$k = 26 = 5 + 3 \times 7$

$k = 27 = 4 \times 5 + 7$

$k = 28 = 4 \times 7$

IH: for any $23 < k < N$: k is a sum of 5's and 7's

IS: Any $N > 23$ is a sum of 5's and 7's.

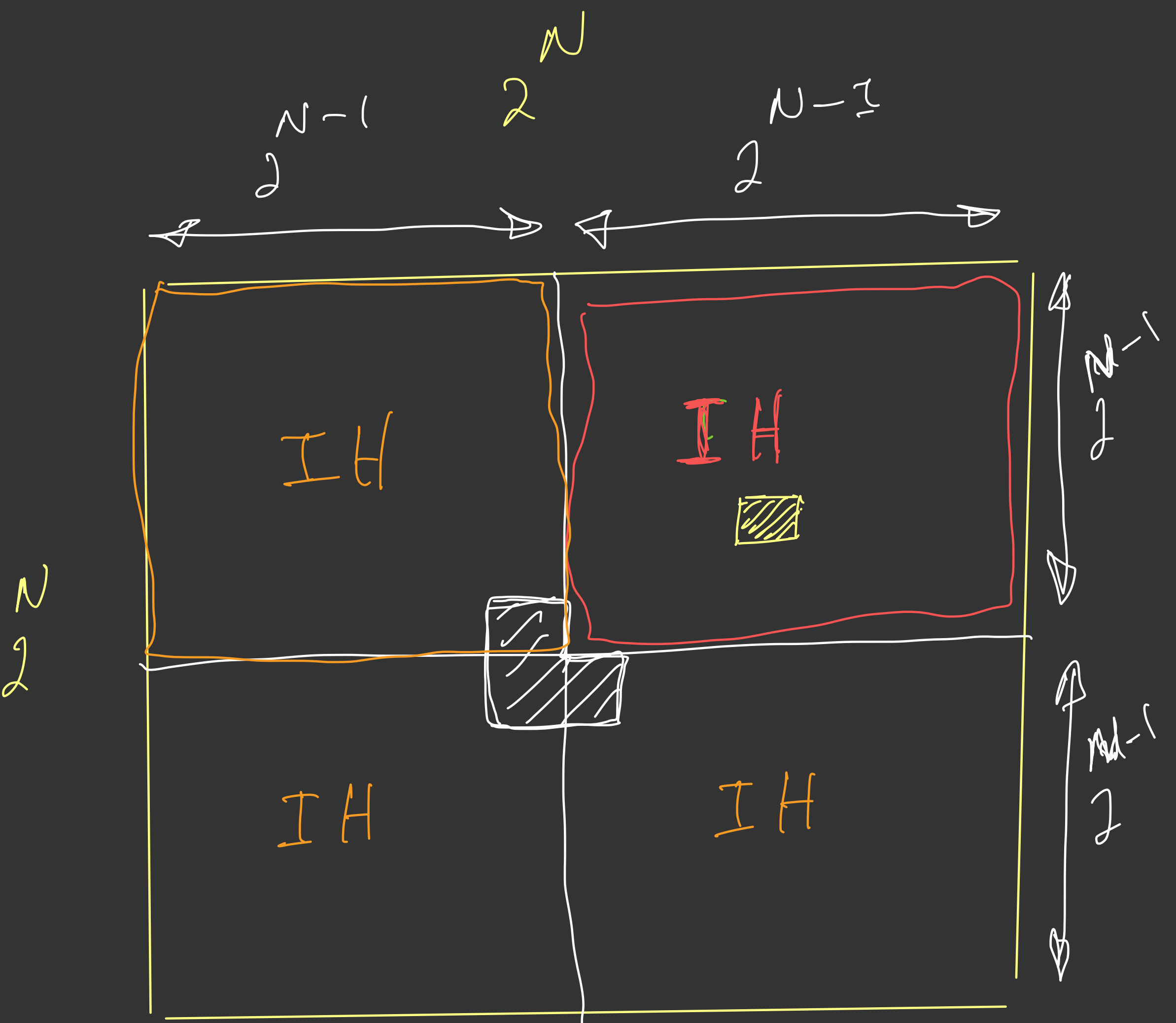
if $N > 28$; then $N - 5 > 23$

$\Rightarrow$ $N-5$ can be written as sum of 5's and 7's

$\Rightarrow$ N can be written as sum of 5's and 7's.

```
Break (N):   # Assume N>23
    if N <= 28
        .....
    else:
        Break(N-5)
        Print('+5')
```

Ex: It is possible to tile a $2^n \times 2^n$ chessboard that misses one square with



BC:

$n = 0$:

do nothing.

IH: for $k < N$:
a $2^k \times 2^k$ board with a missing square can be tiled by 's.

IS: a $2^N \times 2^N$ board with a missing □ can be tiled by 's.

(1) Break the board to 4 $2^{N-1} \times 2^{N-1}$ boards

(2) Use IH to solve the one of four with a missing □

(3) add an in the middle to miss one □ in the other 3.

Use IH for the other 3.

Recall our max finder:

```
0  arr_max(A[1..n]):
1      if n=1: return A[1]
2  ->  B = arr_max(A[1..n-1])
3      if B > A[n]:
4          return B
5      else: return A[n]
```

Ex: Show that arr_max correctly computes max of an array.

Pf:

BC: if $n=1$ the alg returns the only num in A, which is $\max(A)$

IH: for any $k < N$, arr_max$(A[1..k])$ returns max of $A[1..k]$.

IS: to show arr_max$(A[1..N])$ finds $\max(A[1..N])$

1. line 2: by IH, B will contain $\max(A[1..N-1])$
2. (i) max of A is B,
   (ii) max of A is $A[n]$
3. Alg returns $\max(i, ii)$