طراحی الگوریتم ها

مبحث سوم:
حل با روش جایگذاری

سجاد شیرعلی شهرضا

بهار 1402

یکشنبه، 23 بهمن 1401

Based on Stanford CS161 Slides from Summer 2020 by Karey Shi

اطلاع رسانی

- بخش مرتبط کتاب برای این جلسه: 4.3

حل با روش جایگذاری

الگوریتم، اثبات درستی، زمان اجرا

SUBSTITUTION METHOD

1. Guess what the answer is (expand for a few iterations)
2. Prove your guess is correct (using induction)

Let's try it on some example recurrences...

SUBSTITUTION METHOD: EXAMPLE

$$T(n) = 2 \cdot T(n/2) + n$$

$$T(1) = 1$$

STEP 1: guess what the answer is!

You can "unravel" the recursion a few steps & follow the pattern to get a closed form expression!

SUBSTITUTION METHOD: EXAMPLE

$$T(n) = 2 \cdot T(n/2) + n$$

$$T(1) = 1$$

STEP 1: guess what the answer is!

You can "unravel" the recursion a few steps & follow the pattern to get a closed form expression!

$$\begin{aligned} T(n) &= 2T(n/2) + n \\ &= 2(2T(n/4) + n/2) + n \\ &= 4T(n/4) + 2n \\ &= 4(2(T(n/8) + n/4)) + 2n \\ &= 8(T(n/8)) + 3n \\ &= \cdots \end{aligned}$$

SUBSTITUTION METHOD: EXAMPLE

$$T(n) = 2 \cdot T(n/2) + n$$

$$T(1) = 1$$

STEP 1: guess what the answer is!

You can “unravel” the recursion a few steps & follow the pattern to get a closed form expression!

$$\begin{aligned} T(n) &= 2T(n/2) + n \\ &= 2(2T(n/4) + n/2) + n \\ &= 4T(n/4) + 2n \\ &= 4(2(T(n/8) + n/4)) + 2n \\ &= 8(T(n/8)) + 3n \\ &= \cdots \end{aligned}$$

$$\begin{aligned} \mathbf{T(n)} &= \cdots \\ &= nT(n/n) + (\log n)n \\ &= nT(1) + n \log n \\ &= n \log n + n \end{aligned}$$

SUBSTITUTION METHOD: EXAMPLE

$$T(n) = 2 \cdot T(n/2) + n$$

$$T(1) = 1$$

STEP 1: guess what the answer is!

You can "unravel" the recursion a few steps & follow the pattern to get a closed form expression!

$$\begin{aligned} T(n) &= 2T(n/2) + n \\ &= 2(2T(n/4) + n/2) + n \\ &= 4T(n/4) + 2n \\ &= 4(2(T(n/8) + n/4)) + 2n \\ &= 8(T(n/8)) + 3n \\ &= \cdots \end{aligned}$$

$$\begin{aligned} \mathbf{T(n)} &= \cdots \\ &= nT(n/n) + (\log n)n \\ &= nT(1) + n \log n \\ &= n \log n + n \end{aligned}$$

let's guess that $T(n) = n \log n + n$ and try to prove it!

SUBSTITUTION METHOD: EXAMPLE

$$T(n) = 2 \cdot T(n/2) + n$$

$$T(1) = 1$$

STEP 1: guess what the answer is!

You can “unravel” the recursion a few steps & follow the pattern to get a closed form expression!

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

$$= 2(2 \ldots$$

$$= 8(T(n/8)) \ldots$$

$$= \cdots$$

THIS IS A JOB FOR:

PROOF BY INDUCTION!

let’s guess that
T(n) = n log n + n
and try to prove it!

4 INGREDIENTS OF INDUCTION

INDUCTIVE HYPOTHESIS (IH)

This is a statement that's basically what you're trying to prove, except it's written in terms of some variable (e.g. **i**). We need to set up the inductive hypothesis clearly, and our goal in the next three steps is to prove that the IH holds for a whole *range* of values for **i**.

4 INGREDIENTS OF INDUCTION

INDUCTIVE HYPOTHESIS (IH)

This is a statement that's basically what you're trying to prove, except it's written in terms of some variable (e.g. **i**). We need to set up the inductive hypothesis clearly, and our goal in the next three steps is to prove that the IH holds for a whole *range* of values for **i**.

BASE CASE

First establish that the inductive hypothesis holds for some base case value(s) of **i**.

4 INGREDIENTS OF INDUCTION

INDUCTIVE HYPOTHESIS (IH)

This is a statement that's basically what you're trying to prove, except it's written in terms of some variable (e.g. **i**). We need to set up the inductive hypothesis clearly, and our goal in the next three steps is to prove that the IH holds for a whole *range* of values for **i**.

BASE CASE

First establish that the inductive hypothesis holds for some base case value(s) of **i**.

INDUCTIVE STEP *(weak induction version)*

Next, assume that the inductive hypothesis holds when **i** takes on some value **k**.
Now prove that the IH holds as well when **i** takes on the value **k+1**.

4 INGREDIENTS OF INDUCTION

INDUCTIVE HYPOTHESIS (IH)

This is a statement that's basically what you're trying to prove, except it's written in terms of some variable (e.g. **i**). We need to set up the inductive hypothesis clearly, and our goal in the next three steps is to prove that the IH holds for a whole *range* of values for **i**.

BASE CASE

First establish that the inductive hypothesis holds for some base case value(s) of **i**.

INDUCTIVE STEP *(weak induction version)*

Next, assume that the inductive hypothesis holds when **i** takes on some value **k**.
Now prove that the IH holds as well when **i** takes on the value **k+1**.

CONCLUSION

By induction, conclude that the IH holds across the range of **i** you're dealing with.

4 INGREDIENTS OF INDUCTION

INDUCTIVE HYPOTHESIS (IH)

This is a statement that's basically what you're trying to prove, except it's written in terms of some variable (e.g. **i**). We need to set up the inductive hypothesis clearly, and our goal in the next three steps is to prove that the IH holds for a whole *range* of values for **i**.

BASE CASE

First establish that the inductive hypothesis holds for some base case value(s) of **i**.

INDUCTIVE STEP *(strong/complete induction version)*

Next, assume that the IH holds when **i** takes on any value *between [base case value(s)] and some number* **k**. Now prove that the IH holds as well when **i** takes on the value **k+1**.

CONCLUSION

By induction, conclude that the IH holds across the range of **i** you're dealing with.

SUBSTITUTION METHOD: EXAMPLE

$T(n) = 2 \cdot T(n/2) + n$

$T(1) = 1$

Our guess from Step 1:

$\mathbf{T(n) = n \log n + n}$

STEP 2: Try to prove your guess!

SUBSTITUTION METHOD: EXAMPLE

$T(n) = 2 \cdot T(n/2) + n$

$T(1) = 1$

Our guess from Step 1:

$T(n) = n \log n + n$

STEP 2: Try to prove your guess!

- **Inductive Hypothesis**: $T(n) = n \log n + n$

SUBSTITUTION METHOD: EXAMPLE

$T(n) = 2 \cdot T(n/2) + n$

$T(1) = 1$

Our guess from Step 1:

$T(n) = n \log n + n$

STEP 2: Try to prove your guess!

- **Inductive Hypothesis**: $T(n) = n \log n + n$
- **Base case**: Prove IH holds for $n = 1$. $T(1) = 1 = 1 \log 1 + 1$.

SUBSTITUTION METHOD: EXAMPLE

$T(n) = 2 \cdot T(n/2) + n$

$T(1) = 1$

Our guess from Step 1:

$T(n) = n \log n + n$

STEP 2: Try to prove your guess!

- **Inductive Hypothesis**: $T(n) = n \log n + n$
- **Base case**: Prove IH holds for $n = 1$. $T(1) = 1 = 1 \log 1 + 1$.
- **Inductive step**:
  - Let $k > 1$. Assume that the IH holds for all n such that $1 \leq n < k$.
  - $$\begin{aligned} T(k) &= 2 \cdot T(k/2) + k \\ &= 2 \cdot ((k/2)(\log (k/2)) + (k/2)) + k \\ &= 2 \cdot ((k/2)(\log k - 1 + 1)) + k \\ &= 2 \cdot (k/2)(\log k) + k \\ &= k \log k + k \end{aligned}$$

SUBSTITUTION METHOD: EXAMPLE

$T(n) = 2 \cdot T(n/2) + n$

$T(1) = 1$

Our guess from Step 1:

$T(n) = n \log n + n$

STEP 2: Try to prove your guess!

- **Inductive Hypothesis**: $T(n) = n \log n + n$
- **Base case**: Prove IH holds for $n = 1$. $T(1) = 1 = 1 \log 1 + 1$.
- **Inductive step**:
  - Let $k > 1$. Assume that the IH holds for all n such that $1 \leq n < k$.
  - $$\begin{aligned} T(k) &= 2 \cdot T(k/2) + k \\ &= 2 \cdot ((k/2)(\log (k/2)) + (k/2)) + k \\ &= 2 \cdot ((k/2)(\log k - 1 + 1)) + k \\ &= 2 \cdot (k/2)(\log k) + k \\ &= k \log k + k \end{aligned}$$
- **Conclusion:** By induction, $T(n) \leq n \log n + n$ for all $n > 0$.

This satisfies the Big-O definition for $O(n \log n)$ (imagine choosing $c = 2$, $n_0 = 1$)

SUBSTITUTION METHOD: EXAMPLE 1

$T(n) \le n \log n + n$ for all $n > 0$ $\Rightarrow$ **$T(n) = O(n \log n)$**

1. Guess what the answer is (expand for a few iterations)
2. Prove your guess is correct (using induction)

SUBSTITUTION METHOD: EXAMPLE 1

$$T(n) \le n \log n + n \text{ for all } n > 0 \quad \Rightarrow \quad T(n) = O(n \log n)$$

1. Guess what the answer is (expand for a few iterations)
2. Prove your guess is correct (using induction)

But sometimes expanding gets complicated...

سوال؟

SUBSTITUTION METHOD: EXAMPLE 2

$$T(n) = T(n/5) + T(7n/10) + n$$

$$T(n) = 1 \text{ when } 1 \le n \le 10$$

Note:

While Example 1 could have also been solved with the Master Theorem, this one has differently sized subproblems, so the Master Theorem won't apply.

So… Time to use the Substitution Method!

SUBSTITUTION METHOD: EXAMPLE 2

$$T(n) = T(n/5) + T(7n/10) + n$$

$$T(n) = 1 \quad \text{when} \quad 1 \le n \le 10$$

STEP 1: guess what the answer is!

Unraveling this expression gets ugly… (feel free to try it!).

You can also make a semi-educated guess and just hope for the best.

SUBSTITUTION METHOD: EXAMPLE 2

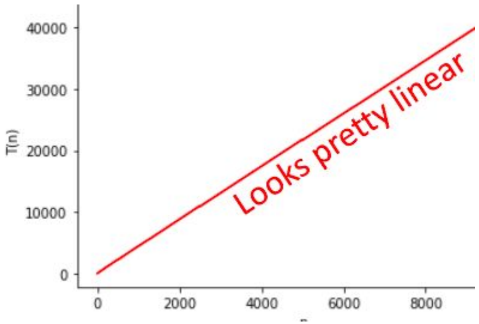
$$T(n) = T(n/5) + T(7n/10) + n$$

$$T(n) = 1 \quad \text{when} \quad 1 \le n \le 10$$

STEP 1: guess what the answer is!

Unraveling this expression gets ugly… (feel free to try it!).

You can also make a semi-educated guess and just hope for the best.



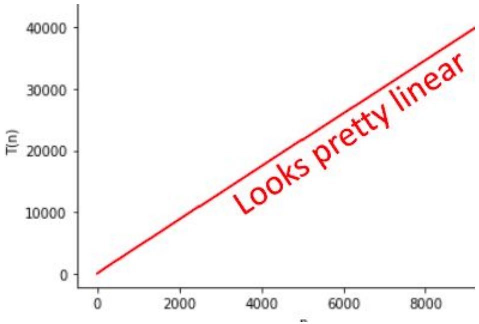
SUBSTITUTION METHOD: EXAMPLE 2

$$T(n) = T(n/5) + T(7n/10) + n$$

$$T(n) = 1 \text{ when } 1 \leq n \leq 10$$

STEP 1: guess what the answer is!

Unraveling this expression gets ugly… (feel free to try it!).
You can also make a semi-educated guess and just hope for the best.



It also feels like it could be better than $2T(n/2) + n$, which we know to be $O(n \log n)$...

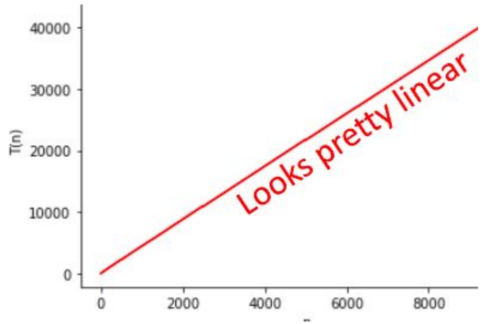
SUBSTITUTION METHOD: EXAMPLE 2

$$T(n) = T(n/5) + T(7n/10) + n$$

$$T(n) = 1 \text{ when } 1 \le n \le 10$$

STEP 1: guess what the answer is!

Unraveling this expression gets ugly… (feel free to try it!).
You can also make a semi-educated guess and just hope for the best.



It also feels like it could be better than 2T(n/2) + n, which we know to be O(n log n)...

Let's guess O(n)

SUBSTITUTION METHOD: EXAMPLE 2

$T(n) = T(n/5) + T(7n/10) + n$

$T(n) = 1$ when $1 \leq n \leq 10$

Our guess from Step 1:

T(n) is O(n)

STEP 2: Prove it!

SUBSTITUTION METHOD: EXAMPLE 2

$T(n) = T(n/5) + T(7n/10) + n$

$T(n) = 1$ when $1 \le n \le 10$

Our guess from Step 1:

$T(n)$ is $O(n)$

STEP 2: Prove it!

WARNING:

You might be tempted to prove this with the inductive hypothesis "$T(n) = O(n)$"

But that doesn't make sense! Formally, this is what your IH would be saying:

"There is some $n_0 > 0$ and some $C > 0$ such that for all $n \ge n_0$, $T(n) \le C \cdot n$"

Your IH is supposed to hold for a *specific* n, not an unbounded *range* of n!

SUBSTITUTION METHOD: EXAMPLE 2

T(n) = T(n/5) + T(7n/10) + n

T(n) = 1 when $1 \le n \le 10$

Our guess from Step 1:

T(n) is O(n)

STEP 2: Prove it!

You … esis

But t … ing:

“There is … n_0 … for all $n \ge n_0$, … $\le C \cdot n$”

Instead, we need to pick a C first,
and have our inductive hypothesis be

$T(n) \le C \cdot n$

Your IH is supposed to hold for a *specific* n, not an unbounded *range* of n!

SUBSTITUTION METHOD: EXAMPLE 2

$T(n) = T(n/5) + T(7n/10) + n$

$T(n) = 1$ when $1 \le n \le 10$

Our guess from Step 1:

T(n) is O(n)

STEP 2: Prove it!

Use a placeholder **C** constant in the big-O proof. We don't know what C should be yet, but let's go through the proof leaving it as C and then figure out what works.

SUBSTITUTION METHOD: EXAMPLE 2

$T(n) = T(n/5) + T(7n/10) + n$

$T(n) = 1$ when $1 \leq n \leq 10$

Our guess from Step 1:

T(n) is O(n)

STEP 2: Prove it!

Use a placeholder **C** constant in the big-O proof. We don't know what C should be yet, but let's go through the proof leaving it as C and then figure out what works.

- **Inductive Hypothesis**: $T(n) \leq \mathbf{C}n$
- **Base case**: Prove IH holds for $1 \leq n \leq 10$. $T(n) = 1 \leq \mathbf{C}n$

SUBSTITUTION METHOD: EXAMPLE 2

$T(n) = T(n/5) + T(7n/10) + n$

$T(n) = 1$ when $1 \leq n \leq 10$

Our guess from Step 1:

T(n) is O(n)

STEP 2: Prove it!

Use a placeholder **C** constant in the big-O proof. We don't know what C should be yet, but let's go through the proof leaving it as C and then figure out what works.

- **Inductive Hypothesis**: $T(n) \leq Cn$
- **Base case**: Prove IH holds for $1 \leq n \leq 10$. $T(n) = 1 \leq Cn$

Whatever we choose C to be, we know C needs to be at least 1

SUBSTITUTION METHOD: EXAMPLE 2

$T(n) = T(n/5) + T(7n/10) + n$

$T(n) = 1$ when $1 \le n \le 10$

Our guess from Step 1:

T(n) is O(n)

STEP 2: Prove it!

Use a placeholder **C** constant in the big-O proof. We don't know what C should be yet, but let's go through the proof leaving it as C and then figure out what works.

- **Inductive Hypothesis**: $T(n) \le \mathbf{C}n$
- **Base case**: Prove IH holds for $1 \le n \le 10$. $T(n) = 1 \le \mathbf{C}n$
- **Inductive step**:
  - Let $k > 10$. Assume that the IH holds for all n such that $1 \le n < k$.

Whatever we choose C to be, we know C needs to be at least 1

SUBSTITUTION METHOD: EXAMPLE 2

$T(n) = T(n/5) + T(7n/10) + n$

$T(n) = 1$ when $1 \le n \le 10$

Our guess from Step 1:

T(n) is O(n)

STEP 2: Prove it!

Use a placeholder **C** constant in the big-O proof. We don't know what C should be yet, but let's go through the proof leaving it as C and then figure out what works.

- **Inductive Hypothesis**: $T(n) \le \mathbf{C}n$
- **Base case**: Prove IH holds for $1 \le n \le 10$. $T(n) = 1 \le \mathbf{C}n$
- **Inductive step**:
  - Let $k > 10$. Assume that the IH holds for all n such that $1 \le n < k$.
  - $$\begin{aligned} T(k) &= k + T(k/5) + T(7k/10) \\ &\le k + \mathbf{C} \cdot (k/5) + \mathbf{C} \cdot (7k/10) \\ &= k \cdot (1 + \mathbf{C}/5 + 7\mathbf{C}/10) \\ &\le \mathbf{C}k \quad ??? \end{aligned}$$
  - (If we find the right C, then we've shown IH holds for n = k)

Whatever we choose C to be, we know C needs to be at least 1

SUBSTITUTION METHOD: EXAMPLE 2

$T(n) = T(n/5) + T(7n/10) + n$

$T(n) = 1$ when $1 \le n \le 10$

Our guess from Step 1:

T(n) is O(n)

STEP 2: Prove it!

Use a placeholder **C** constant in the big-O proof. We don't know what C should be yet, but let's go through the proof leaving it as C and then figure out what works.

- **Inductive Hypothesis**: $T(n) \le Cn$
- **Base case**: Prove IH holds for $1 \le n \le 10$. $T(n) = 1 \le Cn$
- **Inductive step**:
  - Let $k > 10$. Assume that the IH holds for all n such that $1 \le n < k$.
  - $$\begin{aligned} T(k) &= k + T(k/5) + T(7k/10) \\ &\le k + C \cdot (k/5) + C \cdot (7k/10) \\ &= k \cdot (1 + C/5 + 7C/10) \\ &\le Ck \quad ??? \end{aligned}$$
  - (If we find the right C, then we've shown IH holds for n = k)

Whatever we choose C to be, we know C needs to be at least 1

We can just solve for C:

$$1 + C/5 + 7C/10 \le C$$
$$1 + 9C/10 \le C$$
$$1 \le C/10$$

So let's choose C = 10!

SUBSTITUTION METHOD: EXAMPLE 2

$T(n) = T(n/5) + T(7n/10) + n$
$T(n) = 1$ when $1 \le n \le 10$

Our guess from Step 1:
T(n) is O(n)

STEP 2: Prove it!

We can choose C = 10!

- **Inductive Hypothesis**: $T(n) \le \mathbf{10}n$
- **Base case**: Prove IH holds for $1 \le n \le 10$. $T(n) = 1 \le \mathbf{10}n$
- **Inductive step**:
  - Let $k > 10$. Assume that the IH holds for all n such that $1 \le n < k$.
  - $$\begin{aligned} T(k) &= k + T(k/5) + T(7k/10) \\ &\le k + \mathbf{10} \cdot (k/5) + \mathbf{10} \cdot (7k/10) \\ &= k + 2k + 7k \\ &= \mathbf{10}k \end{aligned}$$
  - Thus, the IH holds for $n = k$
- **Conclusion:** With $C = 10$ and $n_0 = 1$, $T(n) \le Cn$ for all $n \ge n_0$. By the Big-O definition, $T(n) = O(n)$.

SUBSTITUTION METHOD: EXAMPLE 2

$T(n) = T(n/5) + T(7n/10) + n$

$T(n) = 1$ when $1 \le n \le 10$

Our guess from Step 1:

T(n) is O(n)

STEP 2: Prove it!

We can choose C = 10!

- **Induct**
- **Base**
- **Induc**
  -
  -

$= \mathbf{10}k$

  - Thus, the IH holds for n = k
- **Conclusion:** With C = 10 and $n_0 = 1$, $T(n) \le Cn$ for all $n \ge n_0$. By the Big-O definition, $T(n) = O(n)$.

Yay! Our guess worked!

But what if you make a bad guess?

سوال؟

WHAT IF YOU MAKE A BAD GUESS?

$T(n) = 2 \cdot T(n/2) + n$

$T(1) = 1$

Bad guess:

$T(n) = O(n)$

WHAT IF YOU MAKE A BAD GUESS?

$T(n) = 2 \cdot T(n/2) + n$

$T(1) = 1$

Bad guess:

$T(n) = O(n)$

STEP 2: Prove it!

Use a placeholder C constant in the big-O proof. We don't know what C should be yet, but let's go through the proof leaving it as C and see if we run into any trouble.

- Inductive Hypothesis: $T(n) \le Cn$
- Base case: $1 = T(n) \le Cn$ for $n = 1$.
- Inductive step:
  - Let $k > 1$. Assume that the IH holds for all n such that $1 \le n < k$.
  - $$\begin{aligned} T(k) &= 2 \cdot T(k/2) + k \\ &\le 2 \cdot C(k/2) + k \\ &= Ck + k \\ &\le Ck??? \end{aligned}$$

WHAT IF YOU MAKE A BAD GUESS?

$T(n) = 2 \cdot T(n/2) + n$

$T(1) = 1$

Bad guess:

$T(n) = O(n)$

STEP 2: Prove it!

Use a placeholder C constant in the big-O proof. We don't know what C should be yet, but let's go through the proof leaving it as C and see if we run into any trouble.

- Inductive Hypothesis: $T(n) \leq Cn$
- Base case: $1 = T(n) \leq Cn$ for $n = 1$.
- Inductive step:
  - Let $k > 1$. Assume that the IH holds for all n such that $1 \leq n < k$.
  - $$\begin{aligned} T(k) &= 2 \cdot T(k/2) + k \\ &\leq 2 \cdot C(k/2) + k \\ &= Ck + k \\ &\leq Ck??? \end{aligned}$$

We need this inequality to hold for the Inductive Step to be complete. However, no choice of C could ever make $Ck + k \leq Ck$!

WHAT IF YOU MAKE A BAD GUESS?

$T(n) = 2 \cdot T(n/2) + n$

$T(1) = 1$

Bad guess:

$T(n) = O(n)$

STEP 2: Prove it!

Use a placeholder C constant in the big-O proof. We don't know what C should be yet, but let's go

A few tips:

If you stumble across impossible inequalities, then your guess was too small! If you end up with an inequality that seems too loose (e.g. $k \le k^2$, $\log k \le k$), maybe try a smaller guess.

$= \quad Ck + k$

$\le \quad Ck???$

hold for the inductive step to be complete. However, no choice of C could ever make Ck+ k ≤ Ck!

SO WHAT HAVE WE LEARNED?

- The substitution method can work when the master theorem doesn't
  - E.g. with different-sized sub-problems

1. **Guess what the answer is (expand for a few iterations)**
2. **Prove your guess is correct (using induction)**

In your final proof, pretend like you didn't do Steps 1 & 2 - no need to say how you unraveled the expression or why you made your guess. Just make sure your proof checks out!

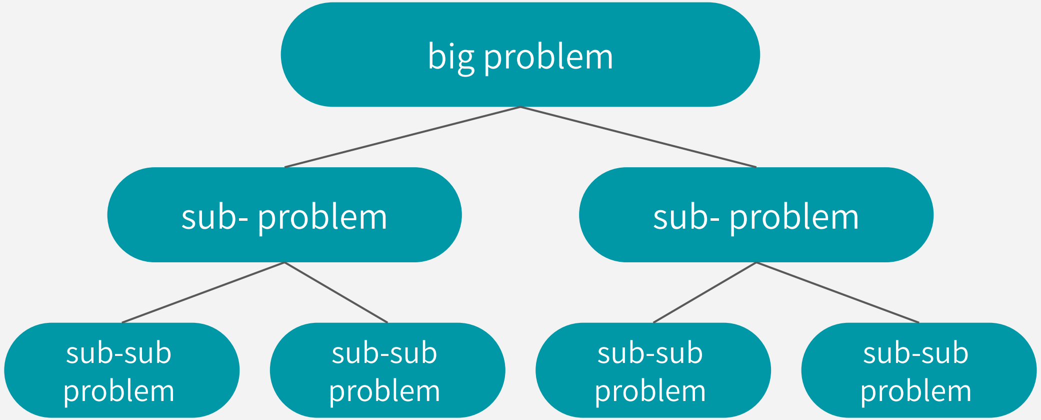
سوال؟

یادآوری روش تقسیم و حل و مرتب سازی ادغامی

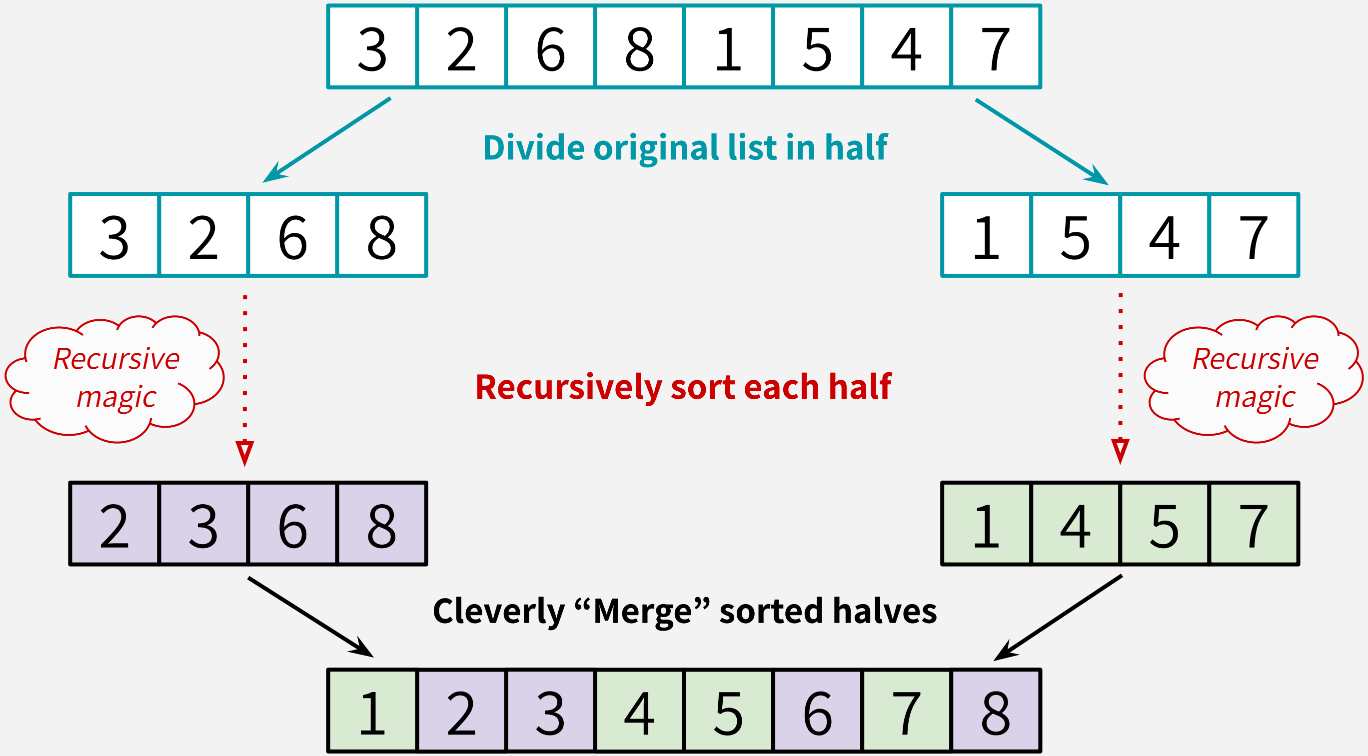
MERGESORT

FROM DATA STRUCTURE COURSE!

- **DIVIDE-AND-CONQUER: an algorithm design paradigm**
  1. break up a problem into smaller subproblems
  2. solve those subproblems *recursively*
  3. combine the results of those subproblems to get the overall answer



MERGESORT


3 2 6 8 1 5 4 7
Divide original list in half
3 2 6 8
1 5 4 7
Recursive magic
Recursively sort each half
Recursive magic
2 3 6 8
1 4 5 7
Cleverly “Merge” sorted halves
1 2 3 4 5 6 7 8

MERGESORT: PSEUDOCODE

Intuition: Divide and Conquer. If you sort your left and right halves, it's easier to "Merge" them into a sorted list.

```
MERGESORT(A):
    n = len(A)
    if n <= 1:
        return A
    L = MERGESORT(A[0:n/2])
    R = MERGESORT(A[n/2:n])
    return MERGE(L,R)
```

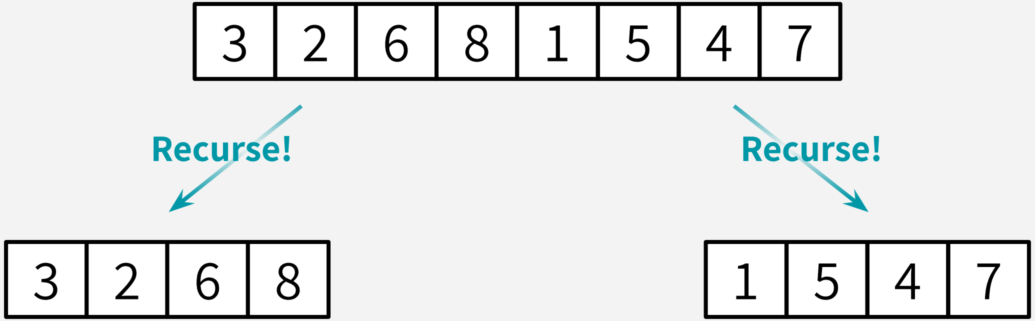
```
MERGE*(L,R):
    result = length n array
    i = 0, j = 0
    for k in [0,...,n-1]:
        if L[i] < R[j]:
            result[k] = L[i]
            i += 1
        else:
            result[k] = R[j]
            j += 1
    return result
```

* Not complete! Some corner cases are missing.

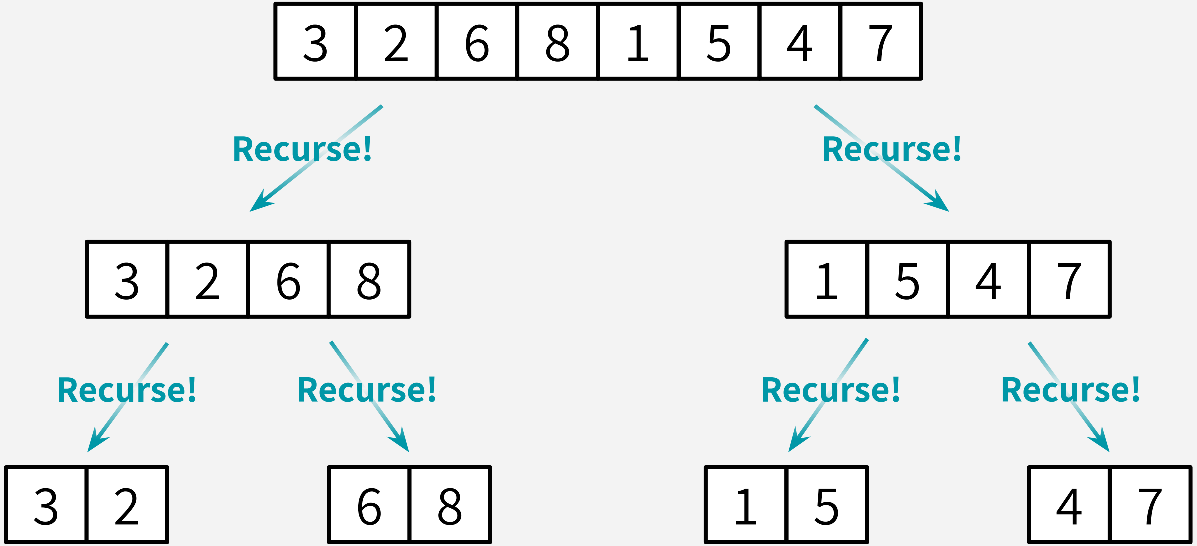
MERGESORT: RECURSIVE CALLS

3	2	6	8	1	5	4	7

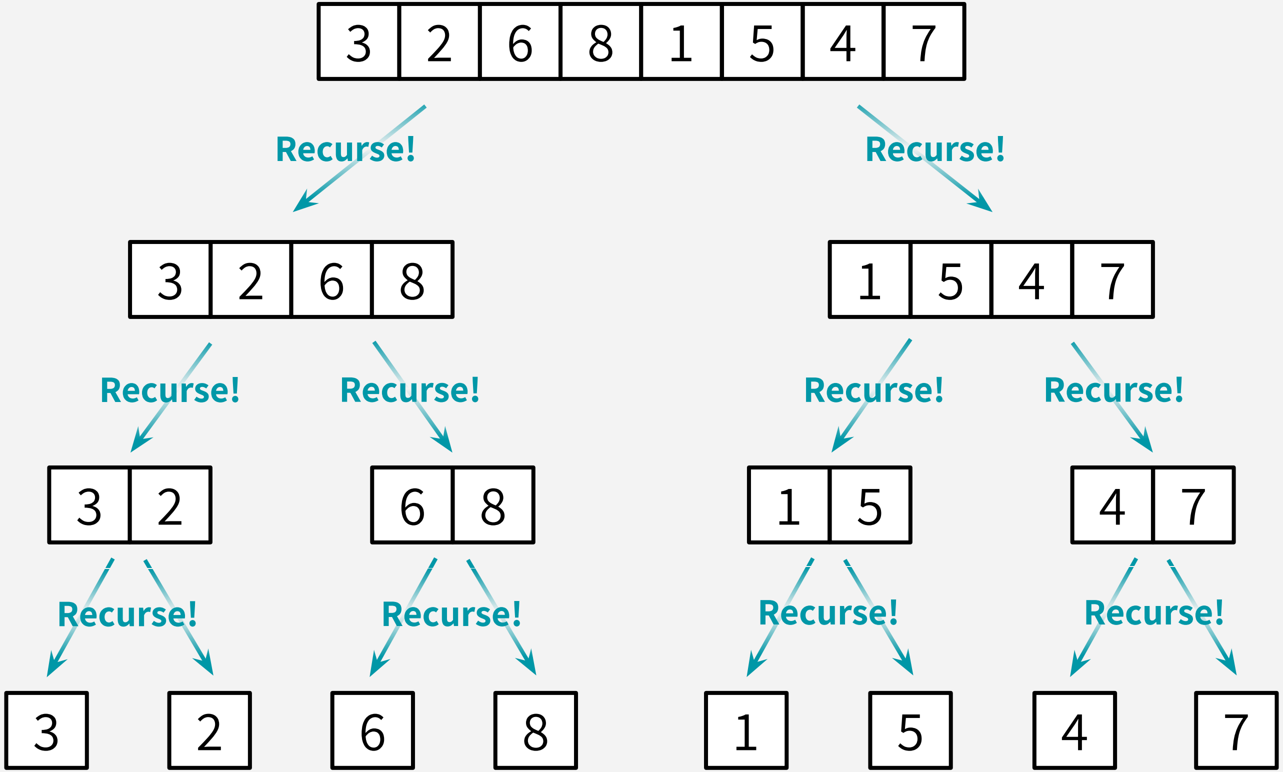
MERGESORT: RECURSIVE CALLS



MERGESORT: RECURSIVE CALLS



MERGESORT: RECURSIVE CALLS

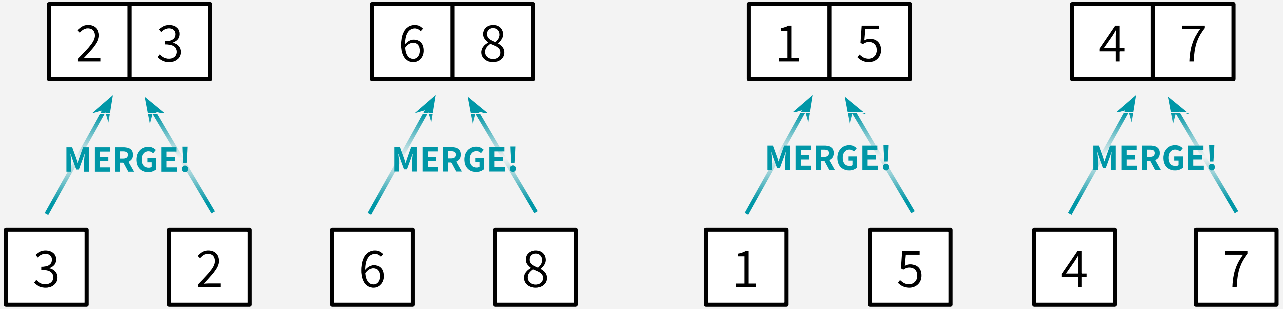


This is where we hit our base case!

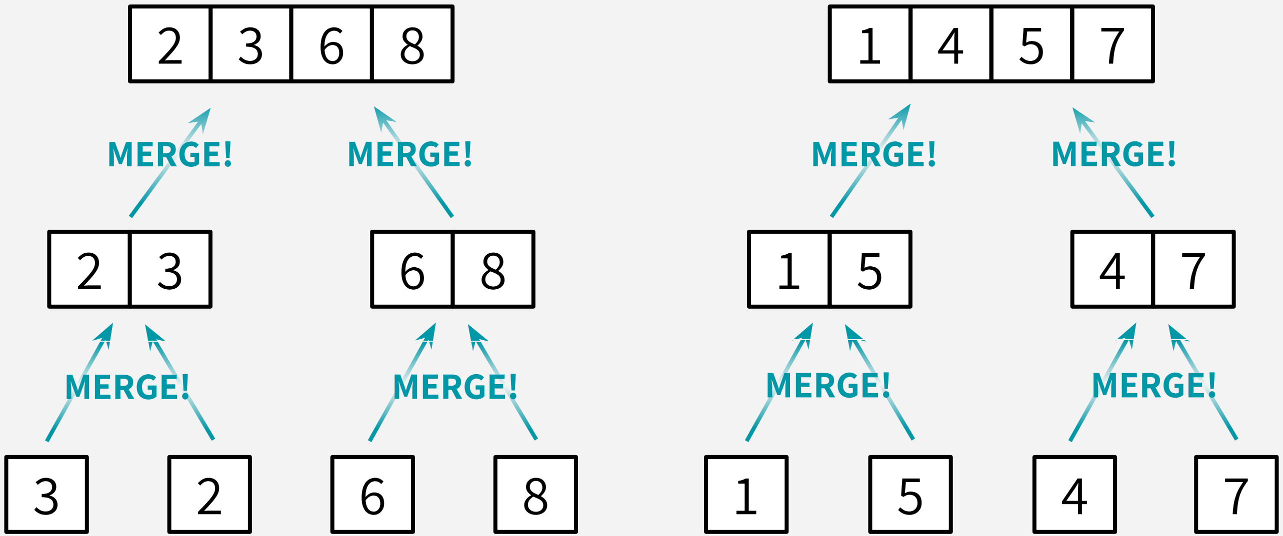
MERGESORT: MERGE STEPS

3	2	6	8	1	5	4	7

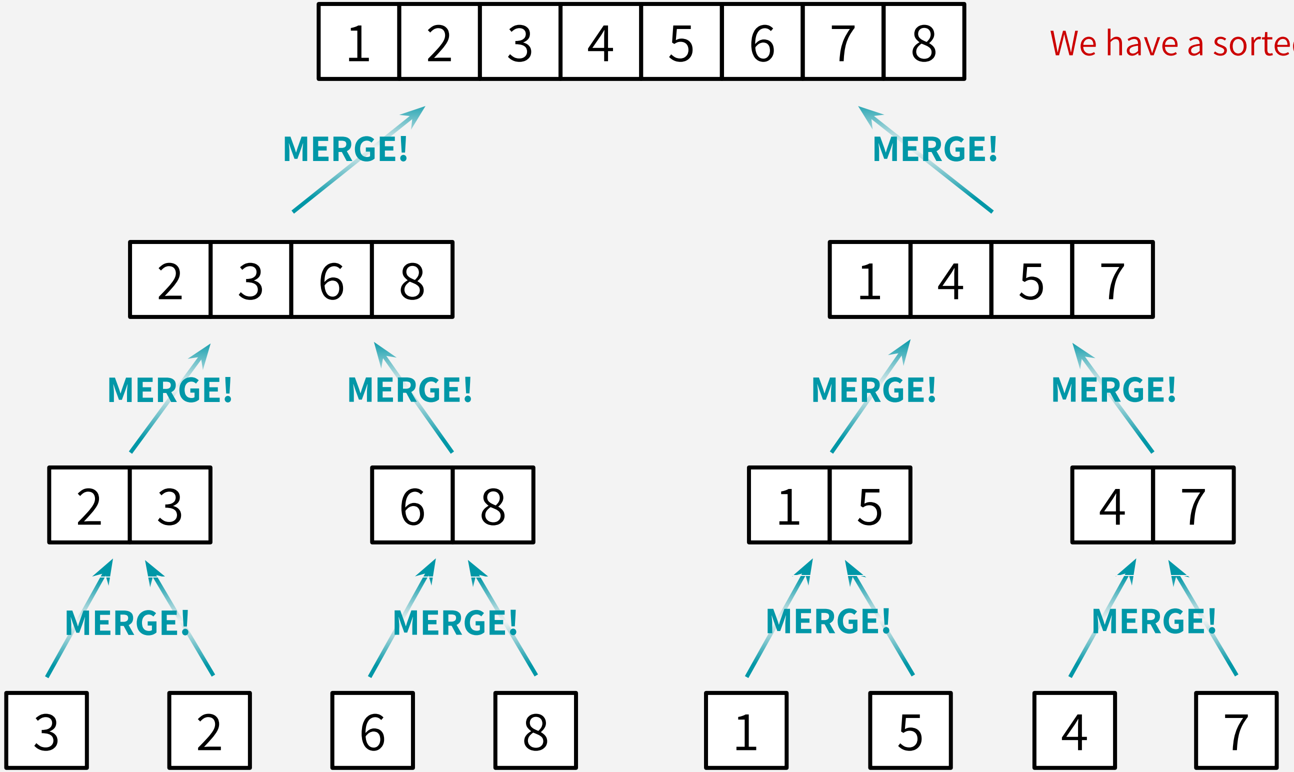
MERGESORT: MERGE STEPS



MERGESORT: MERGE STEPS



MERGESORT: MERGE STEPS



We have a sorted sequence!

سوال؟