PSO 2

Question 1

(Recursion Tree) Give a big-O closed form for each of the following recurrences. (Assume that $T(x) = 1$ for any $x \leq 1$.)

(1) $T(n) = 2T(n/4) + \sqrt{n}$

(2) $T(n) = T(n/2) + T(n/3) + T(n/6) + n$

Question 1

(Recursion Tree) Give a big-O closed form for each of the following recurrences. (Assume that $T(x) = 1$ for any $x \leq 1$.)

(1) $T(n) = 2T(n/4) + \sqrt{n}$

(2) $T(n) = T(n/2) + T(n/3) + T(n/6) + n$

Warning: Solving this $T(n)$ using iterations is a bad idea!

… kind of, we will see that trees help us organize better!

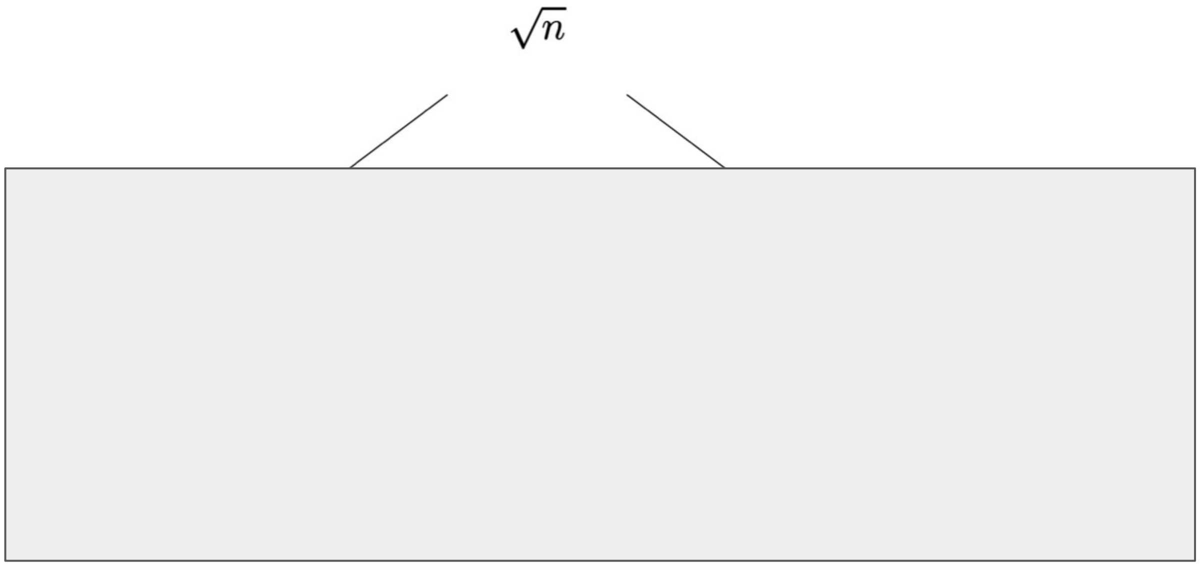
1. Draw out the tree
2. Find the cost at the ith level and the number of levels
3. Derive the sum and closed form

Question 1

(Recursion Tree) Give a big-O closed form for each of the following recurrences. (Assume that $T(x) = 1$ for any $x \leq 1$.)

(1) $T(n) = 2T(n/4) + \sqrt{n}$

(2) $T(n) = T(n/2) + T(n/3) + T(n/6) + n$



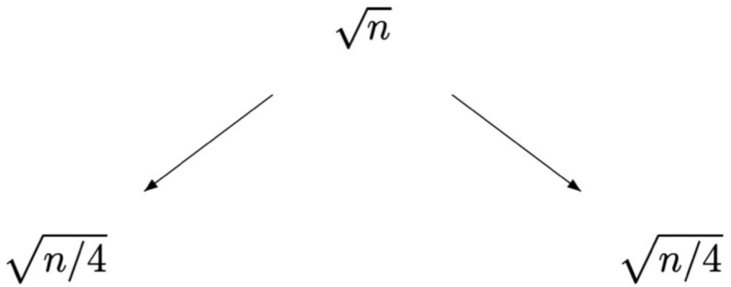
1. Draw out the tree
2. Find the cost at the ith level and the number of levels
3. Derive the sum and closed form

Question 1

(Recursion Tree) Give a big-O closed form for each of the following recurrences. (Assume that $T(x) = 1$ for any $x \leq 1$.)

(1) $T(n) = 2T(n/4) + \sqrt{n}$

(2) $T(n) = T(n/2) + T(n/3) + T(n/6) + n$



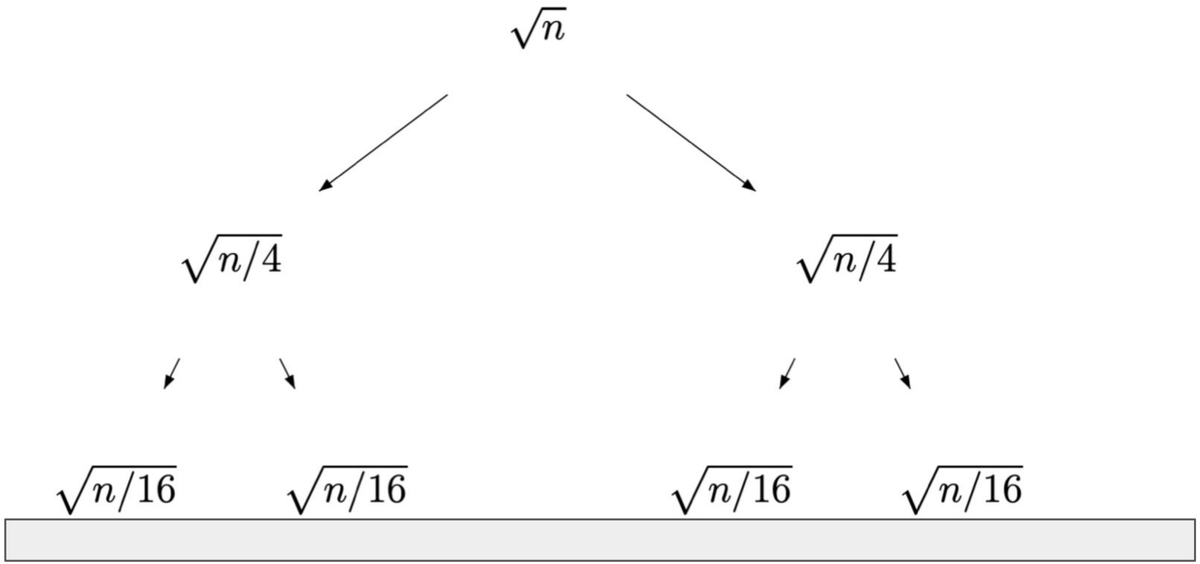
1. Draw out the tree
2. Find the cost at the ith level and the number of levels
3. Derive the sum and closed form

Question 1

(Recursion Tree) Give a big-O closed form for each of the following recurrences. (Assume that $T(x) = 1$ for any $x \leq 1$.)

(1) $T(n) = 2T(n/4) + \sqrt{n}$

(2) $T(n) = T(n/2) + T(n/3) + T(n/6) + n$



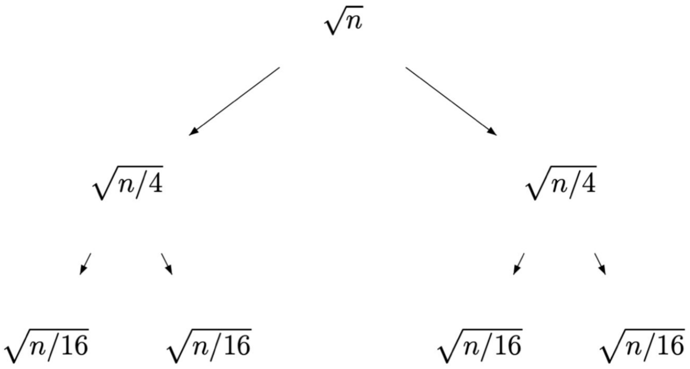
1. Draw out the tree
2. Find the cost at the ith level and the number of levels
3. Derive the sum and closed form

Question 1

(Recursion Tree) Give a big-O closed form for each of the following recurrences. (Assume that $T(x) = 1$ for any $x \leq 1$.)

(1) $T(n) = 2T(n/4) + \sqrt{n}$

(2) $T(n) = T(n/2) + T(n/3) + T(n/6) + n$



1. Draw out the tree
2. Find the cost at the ith level and the number of levels
3. Derive the sum and closed form

Cost at first level:
Cost at second level:
Cost at ith level:

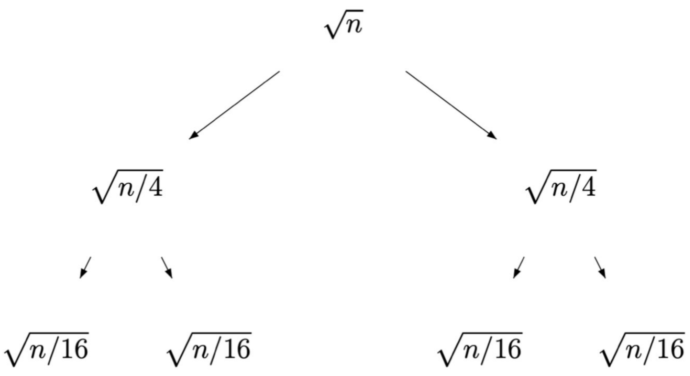
\# levels:

Question 1

(Recursion Tree) Give a big-O closed form for each of the following recurrences. (Assume that $T(x) = 1$ for any $x \leq 1$.)

(1) $T(n) = 2T(n/4) + \sqrt{n}$

(2) $T(n) = T(n/2) + T(n/3) + T(n/6) + n$

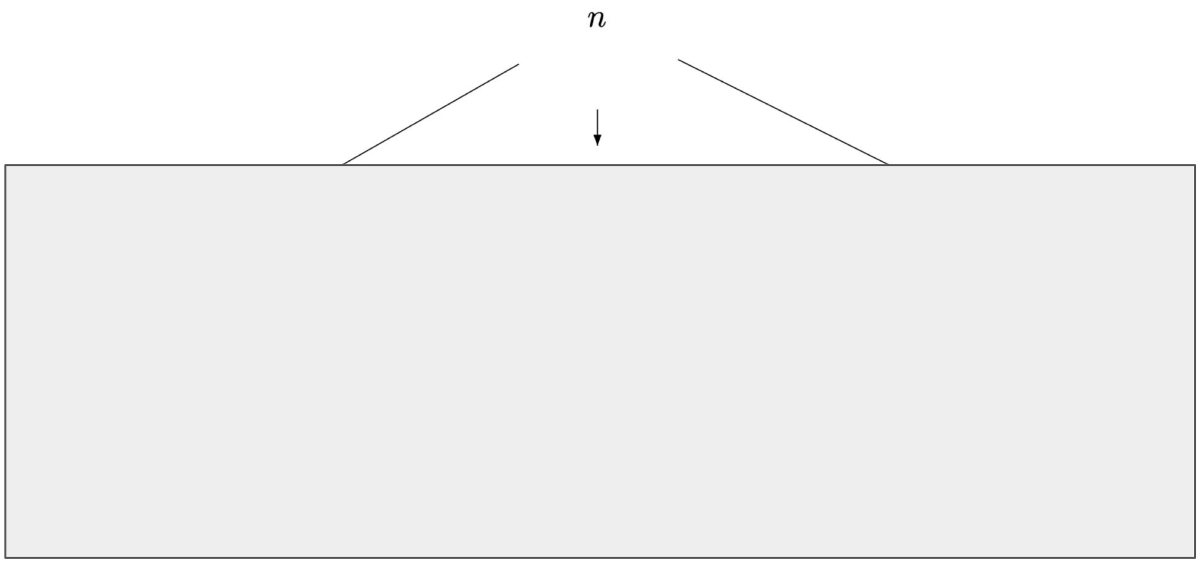


Cost at ith level: $\sqrt{n}$

Number of levels: $\log_4 n$

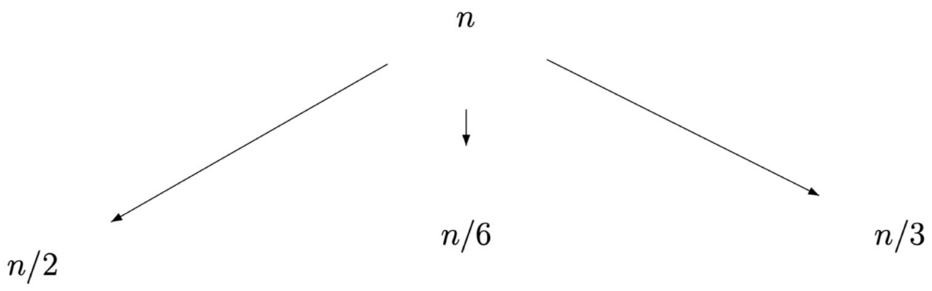
1. Draw out the tree
2. Find the cost at the ith level and the number of levels
3. Derive the sum and closed form

(2) $T(n) = T(n/2) + T(n/3) + T(n/6) + n$



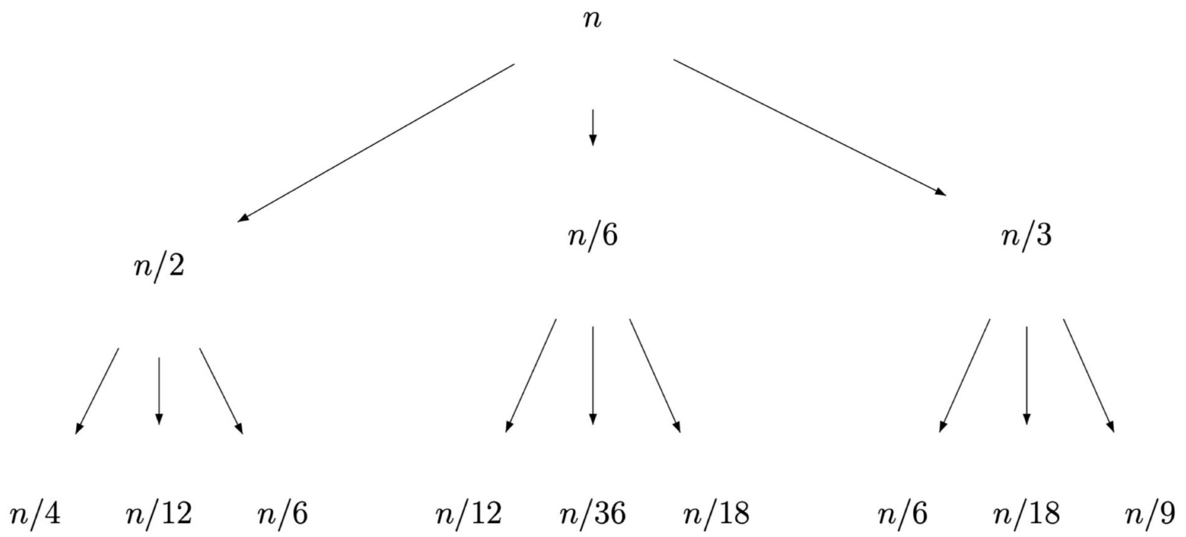
1. Draw out the tree
2. Find the cost at the ith level and the number of levels
3. Derive the sum and closed form

$$(2)\ T(n) = T(n/2) + T(n/3) + T(n/6) + n$$



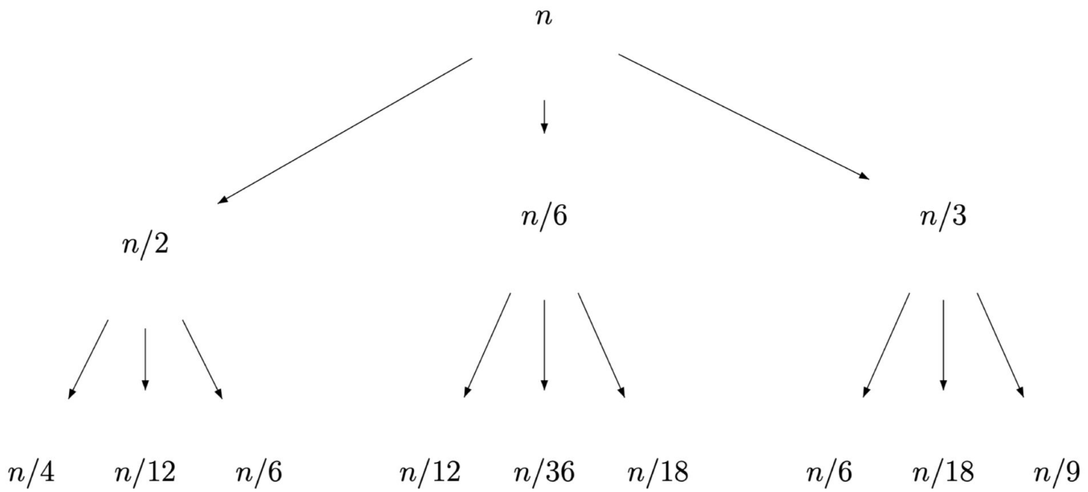
1. Draw out the tree
2. Find the cost at the ith level and the number of levels
3. Derive the sum and closed form

(2) $T(n) = T(n/2) + T(n/3) + T(n/6) + n$



1. Draw out the tree
2. Find the cost at the ith level and the number of levels
3. Derive the sum and closed form

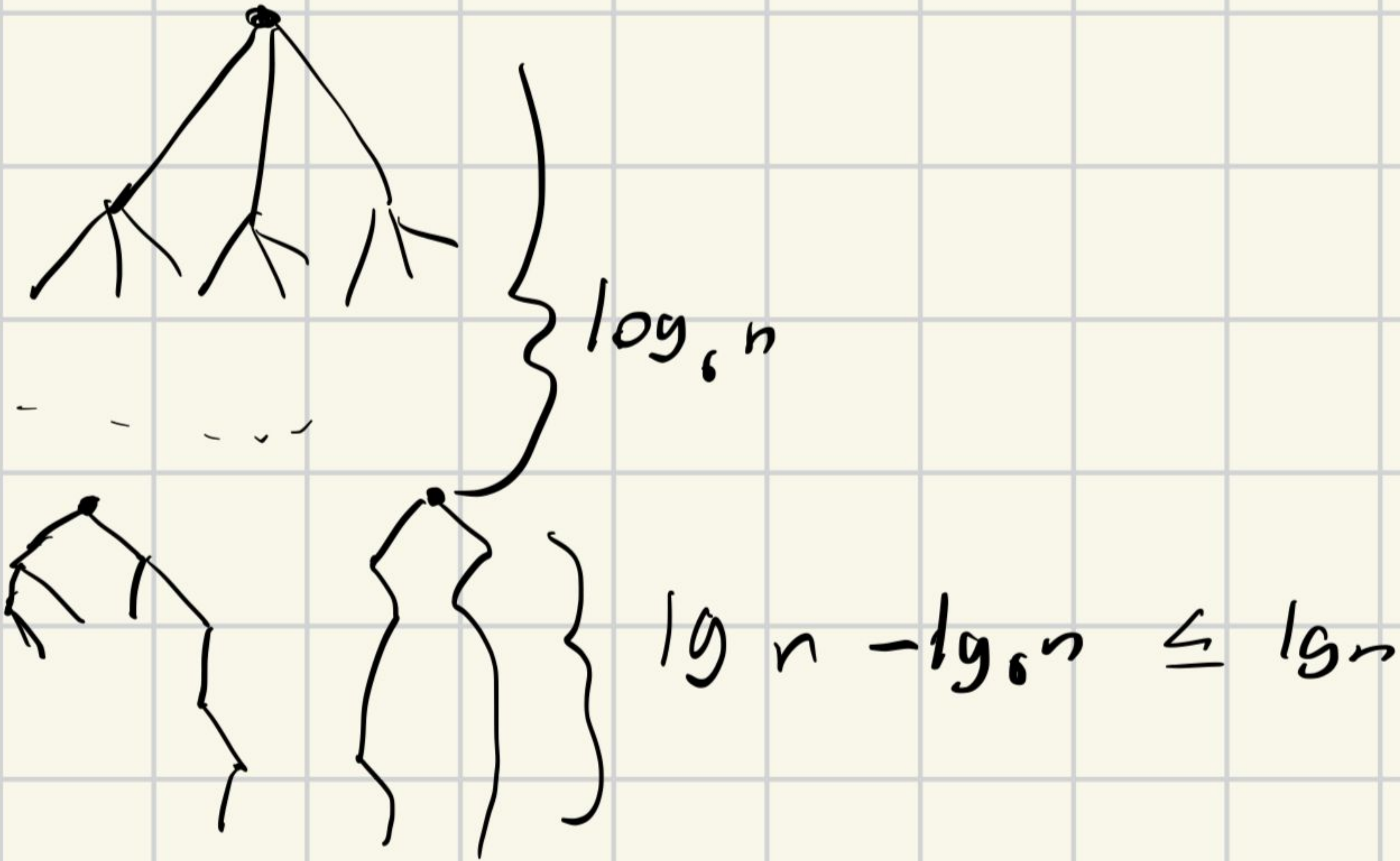
(2) $T(n) = T(n/2) + T(n/3) + T(n/6) + n$



1. Draw out the tree
2. Find the cost at the ith level and the number of levels
3. Derive the sum and closed form

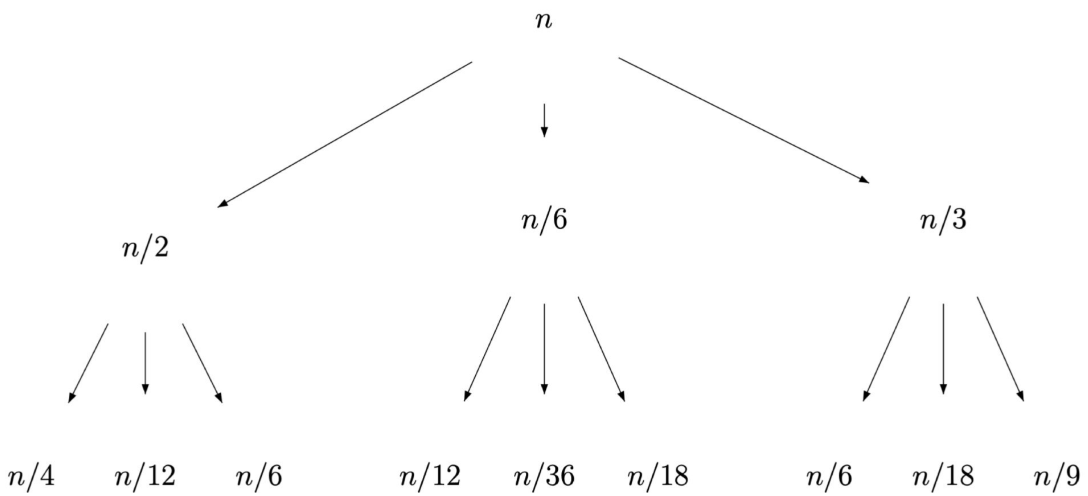
Cost at first level:
Cost at second level:
Cost at ith level:

levels:



log₆ n
lg n − lg₆ n ≤ lg n

(2) $T(n) = T(n/2) + T(n/3) + T(n/6) + n$



1. Draw out the tree
2. Find the cost at the ith level and the number of levels
3. **Derive the sum and closed form**

Cost at ith level: n

Number of levels: $O(\lg n)$

Question 2

(Change a Variable) Give a big-O closed form for the following recurrence.

$$T(n) = 2T(\sqrt{n}) + \log n$$

Question 2

(Change a Variable) Give a big-O closed form for the following recurrence.

$$T(n) = 2T(\sqrt{n}) + \log n$$

What is the problem with a tree?

Question 2

(Change a Variable) Give a big-O closed form for the following recurrence.

$$T(n) = 2T(\sqrt{n}) + \log n$$

We usually like recurrences of this form

$$S(n) = \alpha S(n/\beta) + f(n),$$

E.g question 1 recurrences

Solution: Variable change! But to what value?

Question 2

(Change a Variable) Give a big-O closed form for the following recurrence.

$$T(n) = 2T(\sqrt{n}) + \log n$$

We usually like recurrences of this form $S(n) = \alpha S(n/\beta) + f(n)$,

Change variable: m =

Question 2

(Change a Variable) Give a big-O closed form for the following recurrence.

$$T(n) = 2T(\sqrt{n}) + \log n$$

We usually like recurrences of this form

$$S(n) = \alpha S(n/\beta) + f(n),$$

Change variable: m = log n

$$T(2^m) = 2T(2^{m/2}) + m.$$

Question 2

(Change a Variable) Give a big-O closed form for the following recurrence.

$$T(n) = 2T(\sqrt{n}) + \log n$$

We usually like recurrences of this form

$$S(n) = \alpha S(n/\beta) + f(n),$$

Change variable: m = log n

$$T(2^m) = 2T(2^{m/2}) + m.$$

Change equation: S(m) =

Question 2

(Change a Variable) Give a big-O closed form for the following recurrence.

$$T(n) = 2T(\sqrt{n}) + \log n$$

We usually like recurrences of this form

$$S(n) = \alpha S(n/\beta) + f(n),$$

Change variable: m = log n

$$T(2^m) = 2T(2^{m/2}) + m.$$

Change equation: $S(m) = T(2^m)$

$$S(m) = 2S(m/2) + m,$$

Question 2

(Change a Variable) Give a big-O closed form for the following recurrence.

$$T(n) = 2T(\sqrt{n}) + \log n$$

We usually like recurrences of this form

$$S(n) = \alpha S(n/\beta) + f(n),$$

Change variable: m = log n

$$T(2^m) = 2T(2^{m/2}) + m.$$

Change equation: $S(m) = T(2^m)$

$$S(m) = 2S(m/2) + m,$$

This is just merge sort! O(mlogm) = O(log n * (log log n))

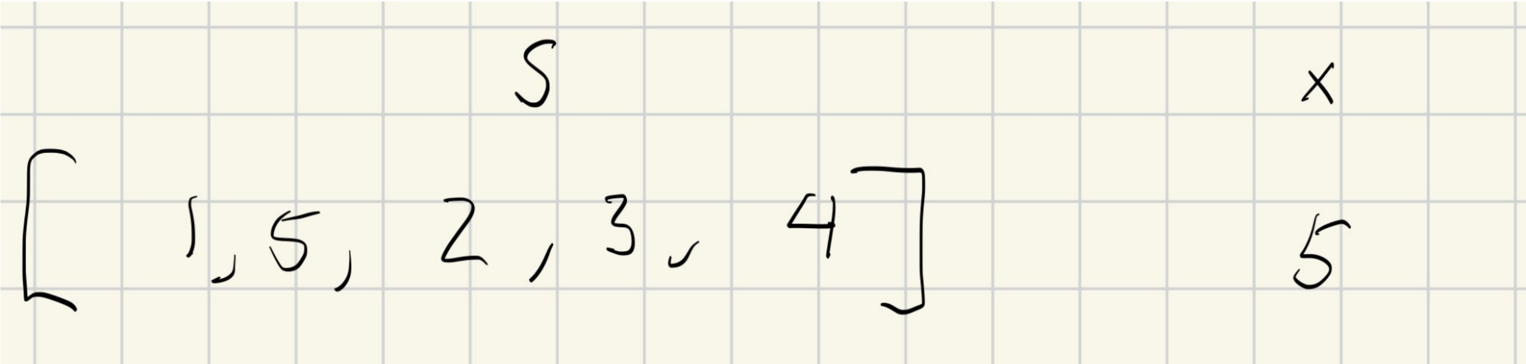
Question 3

(Algorithm Design) Describe a $\Theta(n \log n)$ algorithm that, given a set S of n integers and another integer x, determines whether or not there exist two elements in S whose sum is exactly x.

Question 3

(Algorithm Design) Describe a $\Theta(n \log n)$ algorithm that, given a set S of n integers and another integer x, determines whether or not there exist two elements in S whose sum is exactly x.

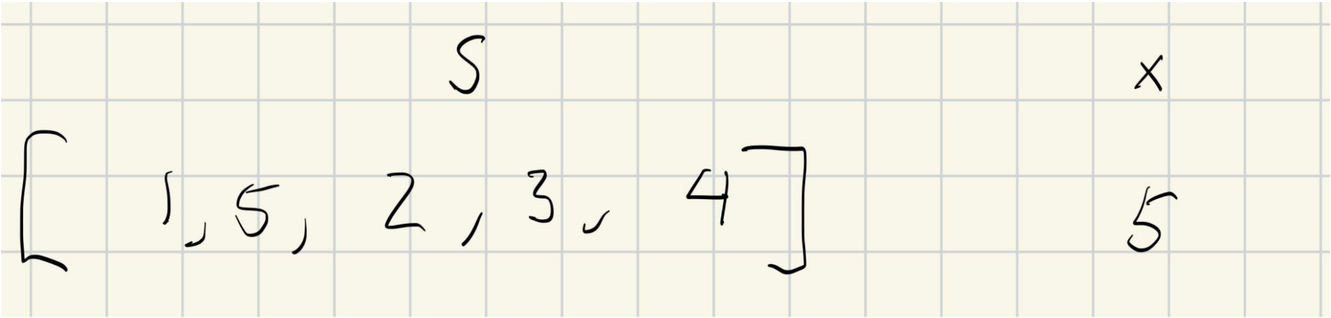
Naive n^2 strategy?

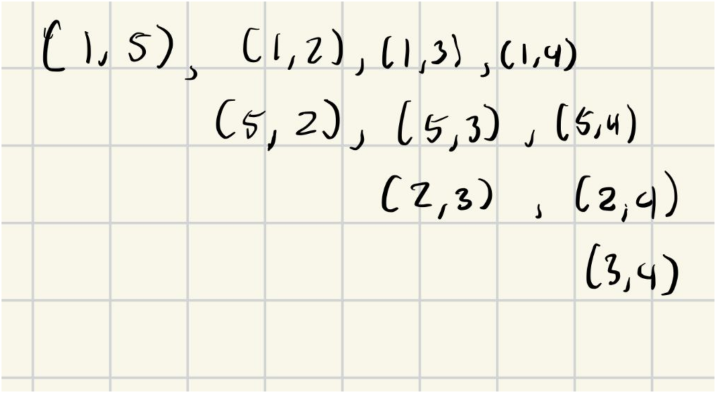


Question 3

(Algorithm Design) Describe a $\Theta(n \log n)$ algorithm that, given a set S of n integers and another integer x, determines whether or not there exist two elements in S whose sum is exactly x.

Naive n^2 strategy?

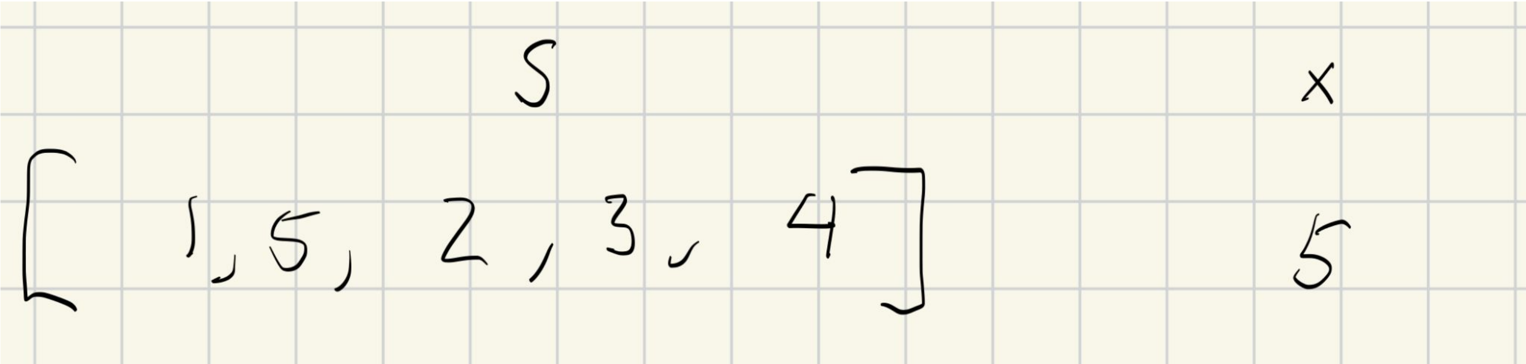




Question 3

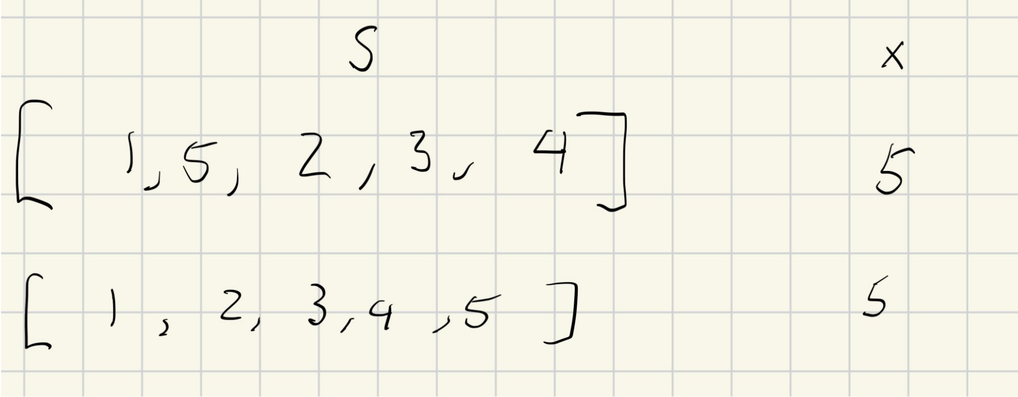
(Algorithm Design) Describe a $\Theta(n \log n)$ algorithm that, given a set S of n integers and another integer x, determines whether or not there exist two elements in S whose sum is exactly x.

What can we do in nlogn time?



Question 3

(Algorithm Design) Describe a $\Theta(n \log n)$ algorithm that, given a set S of n integers and another integer x, determines whether or not there exist two elements in S whose sum is exactly x.



What can we do in nlogn time? Sort

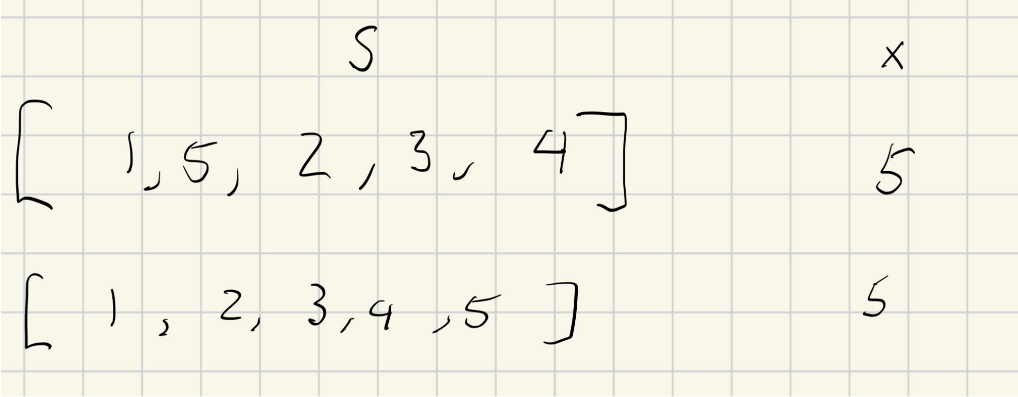
Ok now that elements are sorted,

What do we do?

Question 3

(Algorithm Design) Describe a $\Theta(n \log n)$ algorithm that, given a set S of n integers and another integer x, determines whether or not there exist two elements in S whose sum is exactly x.

What can we do in nlogn time? Sort



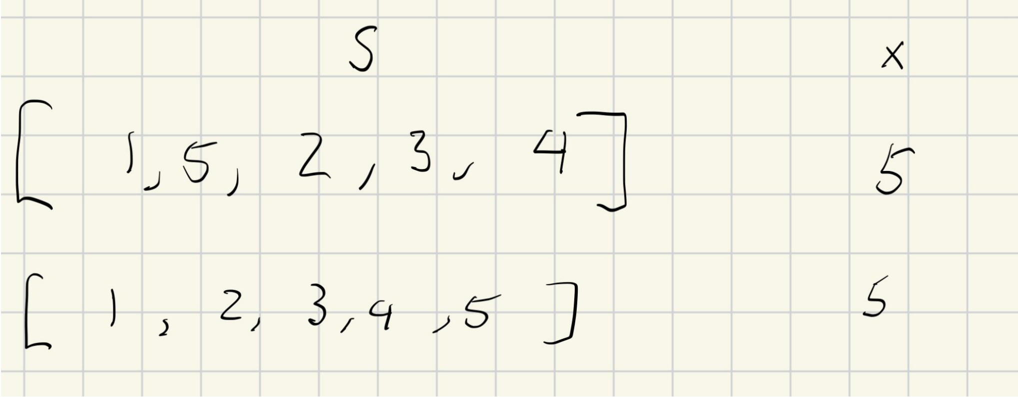
Ok now that elements are sorted,

What do we do?

<u>Idea</u>: find pairs smarter

Question 3

(Algorithm Design) Describe a $\Theta(n \log n)$ algorithm that, given a set S of n integers and another integer x, determines whether or not there exist two elements in S whose sum is exactly x.



What can we do in nlogn time? Sort

Ok now that elements are sorted,

What do we do?

Idea: find pairs smarter

Hold a left,right pointer, calculate sum

If sum > x: move right pointer

If sum < x: move left pointer

Question 4

(Linked List) Consider a sorted circular doubly linked list of N numbers where the head element points to the smallest element in the list. Provide the asymptotic complexity in big-Θ with a brief explanation (including assumptions and analysis for each case, if there is more than one) for the following operations.

1. Inserting an element in its sorted position.
2. Finding the smallest element in the list.
3. Finding the 3^{rd} - largest element in the list.
4. Finding the median in the list.