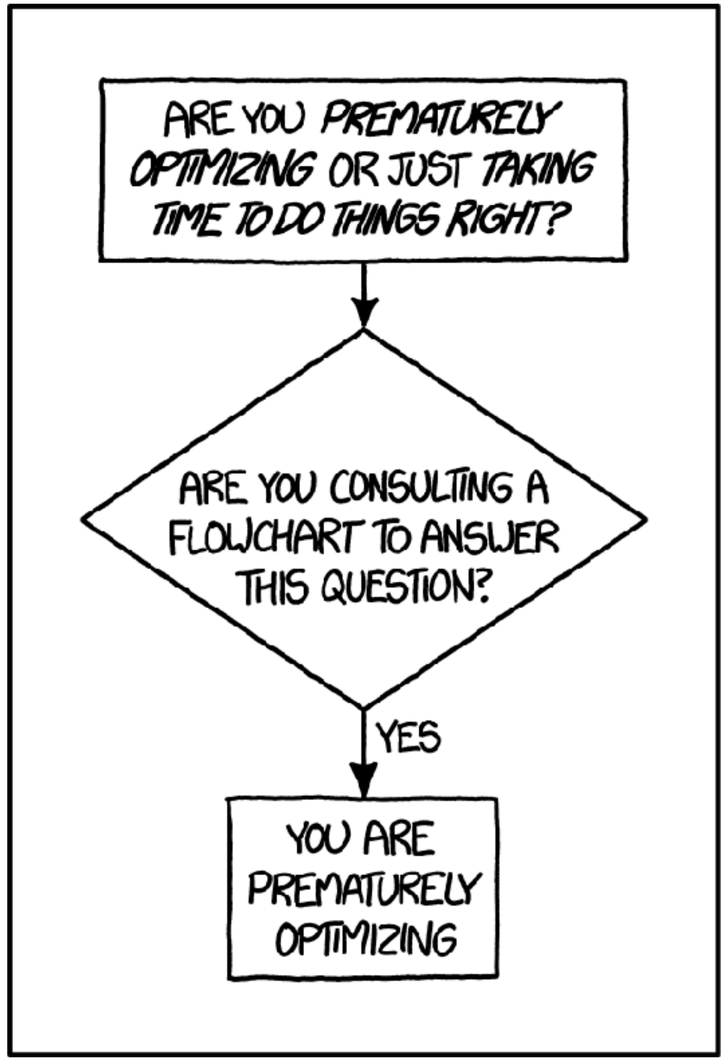
OPTIMIZATION


ARE YOU PREMATURELY OPTIMIZING OR JUST TAKING TIME TO DO THINGS RIGHT?
ARE YOU CONSULTING A FLOWCHART TO ANSWER THIS QUESTION?
YES
YOU ARE PREMATURELY OPTIMIZING

WHAT IS IT?

MAKING THE MOST OF THE RESOURCES YOU HAVE

MAKING THE MOST OF THE RESOURCES YOU HAVE

- Time

MAKING THE MOST OF THE RESOURCES YOU HAVE

- Time
- Space

MAKING THE MOST OF THE RESOURCES YOU HAVE

- Time
- Space
- Money

MAKING THE MOST OF THE RESOURCES YOU HAVE

- Time
- Space
- Money
- Electricity

MAKING THE MOST OF THE RESOURCES YOU HAVE

- Time
- Space
- Money
- Electricity
- Brain power

MAKING THE MOST OF THE RESOURCES YOU HAVE

- Time
- Space
- Money
- Electricity
- Brain power
- etc.

MAKING THE MOST OF THE RESOURCES YOU HAVE

- **Time** ← *We usually talk about this*
- **Space**
- **Money**
- **Electricity**
- **Brain power**
- **etc.**

MAKING THE MOST OF THE RESOURCES YOU HAVE

- **Time** ← *We usually talk about this*
- **Space**
- **Money**
- **Electricity**
- **Brain power**
- **etc.**

but these are important too

CONSIDER THE CONSTRAINTS

CONSIDER THE CONSTRAINTS

Ask your interviewer

FIRST RULE OF OPTIMIZATION:

FIRST RULE OF OPTIMIZATION:

Don’t do it

FIRST RULE OF OPTIMIZATION:

Don’t do it
Seriously

FIRST RULE OF OPTIMIZATION:

Don’t do it
Seriously
Don’t

...UNLESS YOU HAVE TO

...UNLESS YOU HAVE TO

use “benchmarking” to help you find out when it’s necessary

...OR YOU HAVE IMPORTANT INFO AHEAD OF TIME ABOUT HOW YOUR PROGRAM IS GOING TO BE USED

- input size
- rate of requests
- how many other things will rely on it
- etc.

...OR THERE ARE REALLY EASY WINS YOU CAN GET WITHOUT EXPENDING MUCH TIME OR EFFORT

SO...
HOW DO WE GO ABOUT THIS?

DECIDE WHAT YOU'RE OPTIMIZING FOR

YOU CAN'T OPTIMIZE ALL THE THINGS

YOU CAN'T OPTIMIZE ALL THE THINGS

- Time

YOU CAN'T OPTIMIZE ALL THE THINGS

- Time
- Space

YOU CAN'T OPTIMIZE ALL THE THINGS

- Time
- Space
- Money

YOU CAN'T OPTIMIZE ALL THE THINGS

- Time
- Space
- Money
- Electricity

YOU CAN'T OPTIMIZE ALL THE THINGS

- Time
- Space
- Money
- Electricity
- Brain power

YOU CAN'T OPTIMIZE ALL THE THINGS

- Time
- Space
- Money
- Electricity
- Brain power
- etc.

YOU CAN'T OPTIMIZE ALL THE THINGS

- Time
- Space
- Money
- Electricity
- Brain power
- etc.

pick one

YOU CAN'T OPTIMIZE ALL THE THINGS

- Time
- Space
- Money
- Electricity
- Brain power
- etc.

pick one (or two, but don't be greedy)

HOW DO WE DECIDE WHAT TO OPTIMIZE?

IDENTIFY THE BOTTLENECK

IDENTIFY THE BOTTLENECK

Think about the environment you're developing for

IDENTIFY THE BOTTLENECK

Think about the environment you're developing for

Ask your interviewer

bottleneck

bottleneck

making this part wider
won’t help you pour faster

FOCUS ON WIDENING THE BOTTLENECK

FOCUS ON WIDENING THE BOTTLENECK

- apply this recursively

FOCUS ON WIDENING THE BOTTLENECK

- **apply this recursively**
  - “What’s our scarcest resource? Time? Space?”

FOCUS ON WIDENING THE BOTTLENECK

- **apply this recursively**
  - “What’s our scarcest resource? Time? Space?”
  - “Time is our scarcest resource. Which part of our program is taking the most time?” (use benchmarking)

FOCUS ON WIDENING THE BOTTLENECK

- **apply this recursively**
  - “What’s our scarcest resource? Time? Space?”
  - “Time is our scarcest resource. Which part of our program is taking the most time?” (use benchmarking)
  - “This utility function is taking the most time. What’s the Big O? Which part of the function is taking the most time? Can it be improved?”

FOCUS ON WIDENING THE BOTTLENECK

- **apply this recursively**
  - “What’s our scarcest resource? Time? Space?”
  - “Time is our scarcest resource. Which part of our program is taking the most time?” (use benchmarking)
  - “This utility function is taking the most time. What’s the Big O? Which part of the function is taking the most time? Can it be improved?”
- **go around bottlenecks you don’t have much control over**

FOCUS ON WIDENING THE BOTTLENECK

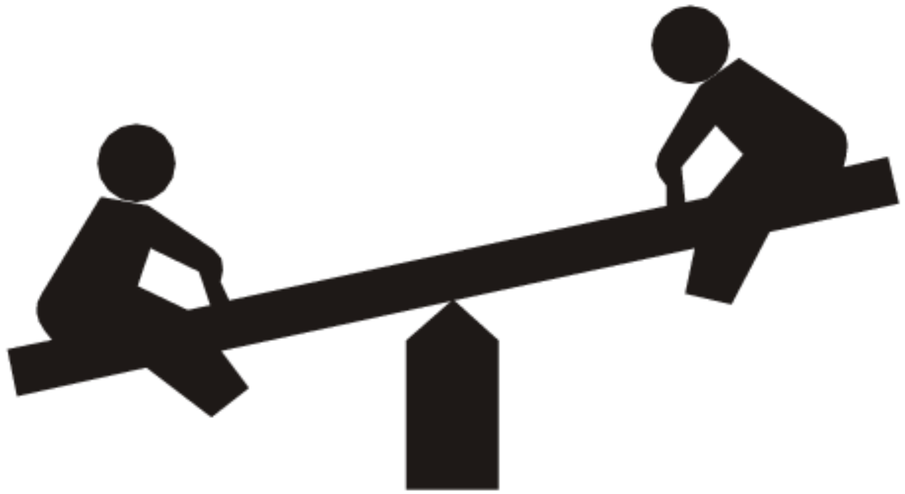
- **apply this recursively**
  - “What’s our scarcest resource? Time? Space?”
  - “Time is our scarcest resource. Which part of our program is taking the most time?” (use benchmarking)
  - “This utility function is taking the most time. What’s the Big O? Which part of the function is taking the most time? Can it be improved?”
- **go around bottlenecks you don’t have much control over**
  - “Network latency is our bottleneck. Let’s try to minimize the size and frequency of our API calls.”

PRO TIP #1:
SEE IF THE PROBLEM CAN BE REDUCED TO A SIMPLER PROBLEM

EXAMPLE PROBLEM #1:

Write a function that returns true if any permutation of a string is a palindrome.

OPTIMIZATION GENERALLY INVOLVES TRADE-OFFS



SPACE FOR TIME IS THE MOST COMMON TRADEOFF

HOW DO WE SAVE TIME AT THE EXPENSE OF SPACE?

DATA STRUCTURES! 🎉

PRO TIP #2:
USE A HASH TABLE

EXAMPLE PROBLEM #2:

Write a function which takes in a number and a sorted array of numbers. Return true if any 2 numbers could add up to the number passed in.

PRO TIP #3:
FOR SORTED DATA, USE: BINARY SEARCH OR "RATCHET"

DYNAMIC PROGRAMMING

DYNAMIC PROGRAMMING

Breaking a big problem down into smaller sub-problems and solving those instead

DYNAMIC PROGRAMMING

Breaking a big problem down into smaller sub-problems and solving those instead

Think recursion!

MEMOIZATION

MEMOIZATION

Storing the results of previous function invocations for easy (fast) future access

EXAMPLE PROBLEM #3: FIBONACCI

ASK QUESTIONS

ASK QUESTIONS

- Should I worry about optimization?

ASK QUESTIONS

- Should I worry about optimization?
- What should I optimize for?

ASK QUESTIONS

- **Should I worry about optimization?**
- **What should I optimize for?**
  - What environment are we in? What are the constraints?

ASK QUESTIONS

- **Should I worry about optimization?**
- **What should I optimize for?**
  - What environment are we in? What are the constraints?
- **Is the input sorted?**

ASK QUESTIONS

- **Should I worry about optimization?**
- **What should I optimize for?**
  - What environment are we in? What are the constraints?
- **Is the input sorted?**
- **Will this only run one time or many times?**

ASK QUESTIONS

- **Should I worry about optimization?**
- **What should I optimize for?**
  - What environment are we in? What are the constraints?
- **Is the input sorted?**
- **Will this only run one time or many times?**
  - Optimizing a one-off solution is different than optimizing the average for repeated executions

OTHER TIPS:

OTHER TIPS:

- Pay very careful attention to the details in the problem description

OTHER TIPS:

- **Pay very careful attention to the details in the problem description**
  - Most problems won’t contain irrelevant info (though it’s not impossible)

OTHER TIPS:

- **Pay very careful attention to the details in the problem description**
  - Most problems won't contain irrelevant info (though it's not impossible)
  - Try to take advantage of EVERY piece of info given to you

OTHER TIPS:

- **Pay very careful attention to the details in the problem description**
  - Most problems won't contain irrelevant info (though it's not impossible)
  - Try to take advantage of EVERY piece of info given to you
- **Consider the best conceivable runtime**

OTHER TIPS:

- **Pay very careful attention to the details in the problem description**
  - Most problems won't contain irrelevant info (though it's not impossible)
  - Try to take advantage of EVERY piece of info given to you
- **Consider the best conceivable runtime**
- **See if you can do some pre-computation up front to save time later**

OTHER TIPS:

- **Pay very careful attention to the details in the problem description**
  - Most problems won't contain irrelevant info (though it's not impossible)
  - Try to take advantage of EVERY piece of info given to you
- **Consider the best conceivable runtime**
- **See if you can do some pre-computation up front to save time later**
  - Boyer-Moore string search algorithm

TAKEAWAYS

- **Understand the input, environment, and use cases**
- **Think about repeated executions**
- **Don’t optimize unless you have to**
- **In which case identify the bottleneck**
  - by measuring, e.g. dev tools performance tab
  - by analysis, e.g. big O
- **Use the right data structure for the job (ehem hash table)**
- **Sorted data: binary search OR ratcheting**
- **Dynamic programming (for overlapping sub-problems)**

TAKEAWAYS

- **Understand the input, environment, and use cases**
- **Think about repeated executions**
- **Don't optimize unless you have to**
- **In which case identify the bottleneck**
  - by measuring, e.g. dev tools performance tab
  - by analysis, e.g. big O
- **Use the right data structure for the job (ehem hash table)**
- **Sorted data: binary search OR ratcheting**
- **Dynamic programming (for overlapping sub-problems)**

ask questions