Cracking the Coding Interview

the abridged version

Gayle L.McDowell | Founder / CEO

CareerCup

gayle gayle in/gaylemcd

careercup.com

Hi! I'm Gayle Laakmann McDowell

(CS) Penn University of Pennsylvania   Wharton University of Pennsylvania (MBA)

<dev> Microsoft Google </dev>

careercup.com

Author   Interview Coach   Interview Consulting

CRACKING the CODING INTERVIEW
189 Programming Questions & Solutions
Gayle Laakmann McDowell 6th Edition
CRACKING the PM INTERVIEW
How to Land a Product Manager Job in Technology
Gayle Laakmann McDowell | Jackie Bavaro
CRACKING the TECH CAREER
Insider Advice on Landing a Job at Google, Microsoft, Apple, or Any Top Tech Company
Gayle Laakmann McDowell
Wiley

Email: **g@gayle.com**
Subject: **mockathon2016**

01

Evaluation

What it’s all about

Why?

- Analytical skills
- How you think
- Make tradeoffs
- Push through hard problems
- Communication
- Strong CS fundamentals

What is NOT expected

- To know the answers
- To solve immediately
- To code perfectly

(It's nice. It just doesn't happen*.)

** Okay fine. It happened once, in 2000+ hiring packets.*

What IS expected

- Be excited about hard problems
- Drive!
  - Keep trying when stuck
  - More than just "correct"
- Pay attention to me!
- Write real code

Show me how you think!

Email: **g@gayle.com**
Subject: **mockathon2016**

02

Preparation

Getting ready

Essential Knowledge

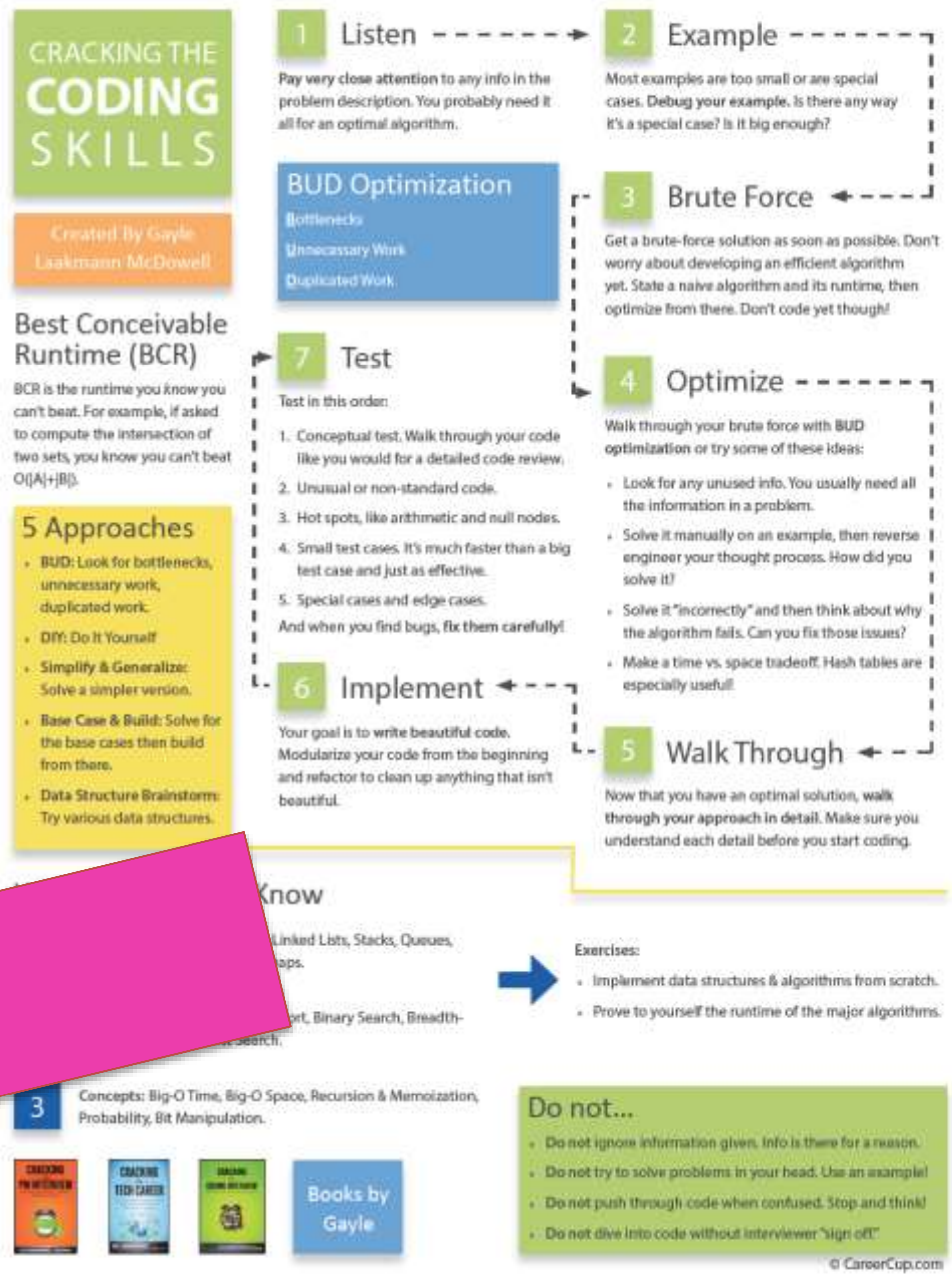
Data Structures	Algorithms	Concepts
ArrayLists	Merge Sort	Big O Time
Hash Tables	Quick Sort	Big O Space
Trees (+ Tries) & Graphs	Breadth-First Search	Recursion
Linked Lists	Depth-First Search	Memoization / Dynamic Programming
Stacks / Queues	Binary Search	
Heaps		

Preparation

- MASTER Big O
- Implement DS/Algorithms
- Practice with interview questions
- Code on paper/whiteboard
- Mock interviews

PUSH YOURSELF!

How To Approach



CRACKING THE CODING SKILLS

Created By Gayle Laakmann McDowell

1 Listen

Pay very close attention to any info in the problem description. You probably need it all for an optimal algorithm.

2 Example

Most examples are too small or are special cases. **Debug your example.** Is there any way it's a special case? Is it big enough?

BUD Optimization

Bottlenecks
Unnecessary Work
Duplicated Work

3 Brute Force

Get a brute-force solution as soon as possible. Don't worry about developing an efficient algorithm yet. State a naive algorithm and its runtime, then optimize from there. Don't code yet though!

Best Conceivable Runtime (BCR)

BCR is the runtime you know you can't beat. For example, if asked to compute the intersection of two sets, you know you can't beat O(|A|+|B|).

7 Test

Test in this order:

1. Conceptual test. Walk through your code like you would for a detailed code review.
2. Unusual or non-standard code.
3. Hot spots, like arithmetic and null nodes.
4. Small test cases. It's much faster than a big test case and just as effective.
5. Special cases and edge cases.

And when you find bugs, **fix them carefully!**

4 Optimize

Walk through your brute force with **BUD optimization** or try some of these ideas:

- Look for any unused info. You usually need all the information in a problem.
- Solve it manually on an example, then reverse engineer your thought process. How did you solve it?
- Solve it "incorrectly" and then think about why the algorithm fails. Can you fix those issues?
- Make a time vs. space tradeoff. Hash tables are especially useful!

5 Approaches

- **BUD:** Look for bottlenecks, unnecessary work, duplicated work.
- **DIY:** Do It Yourself
- **Simplify & Generalize:** Solve a simpler version.
- **Base Case & Build:** Solve for the base cases then build from there.
- **Data Structure Brainstorm:** Try various data structures.

6 Implement

Your goal is to **write beautiful code.** Modularize your code from the beginning and refactor to clean up anything that isn't beautiful.

5 Walk Through

Now that you have an optimal solution, **walk through your approach in detail.** Make sure you understand each detail before you start coding.

Know

Linked Lists, Stacks, Queues, aps.

ort, Binary Search, Breadth- Search.

3 Concepts: Big-O Time, Big-O Space, Recursion & Memoization, Probability, Bit Manipulation.

Exercises:

- Implement data structures & algorithms from scratch.
- Prove to yourself the runtime of the major algorithms.

Books by Gayle

Do not...

- Do not ignore information given. Info is there for a reason.
- Do not try to solve problems in your head. Use an example!
- Do not push through code when confused. Stop and think!
- Do not dive into code without interviewer "sign off."

© CareerCup.com

Email: **g@gayle.com**

Subject: **mockathon2016**

CrackingTheCodingInterview.com → "Resources"

Email: g@gayle.com
Subject: mockathon2016

03

Doing It

7 Steps to Solve

careercup.com

step

1

Listen (for clues & details)

step

2

Draw an Example

Big Enough

+

General Purpose

step

3

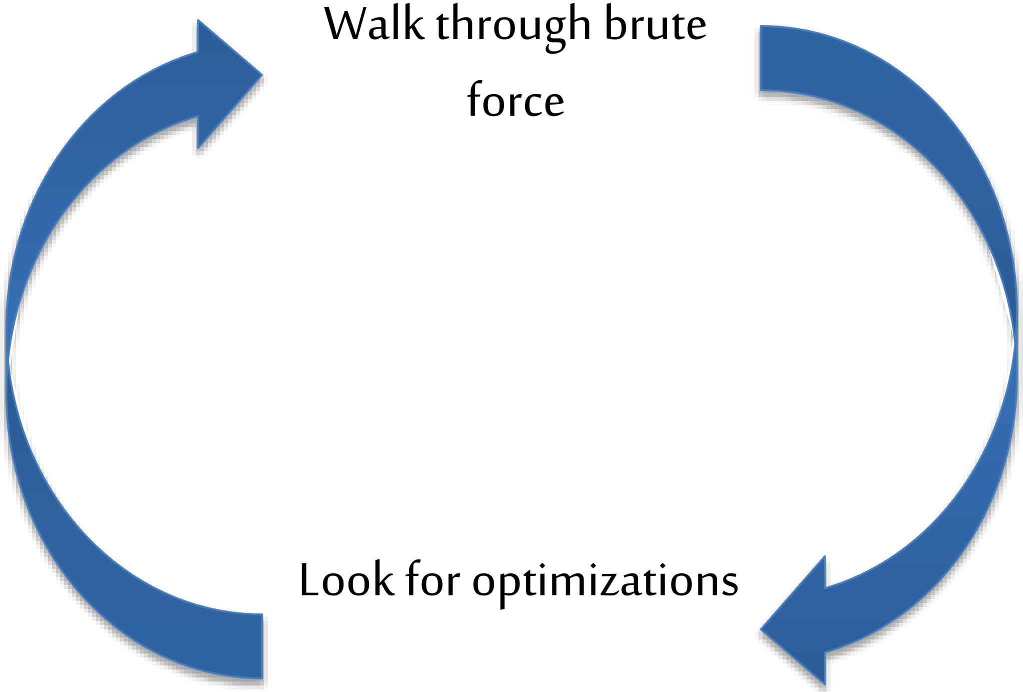
Brute Force / Naive

Stupid & terrible is okay!

step

4

Optimize



Walk through brute force

Look for optimizations

step

5

Walk Through

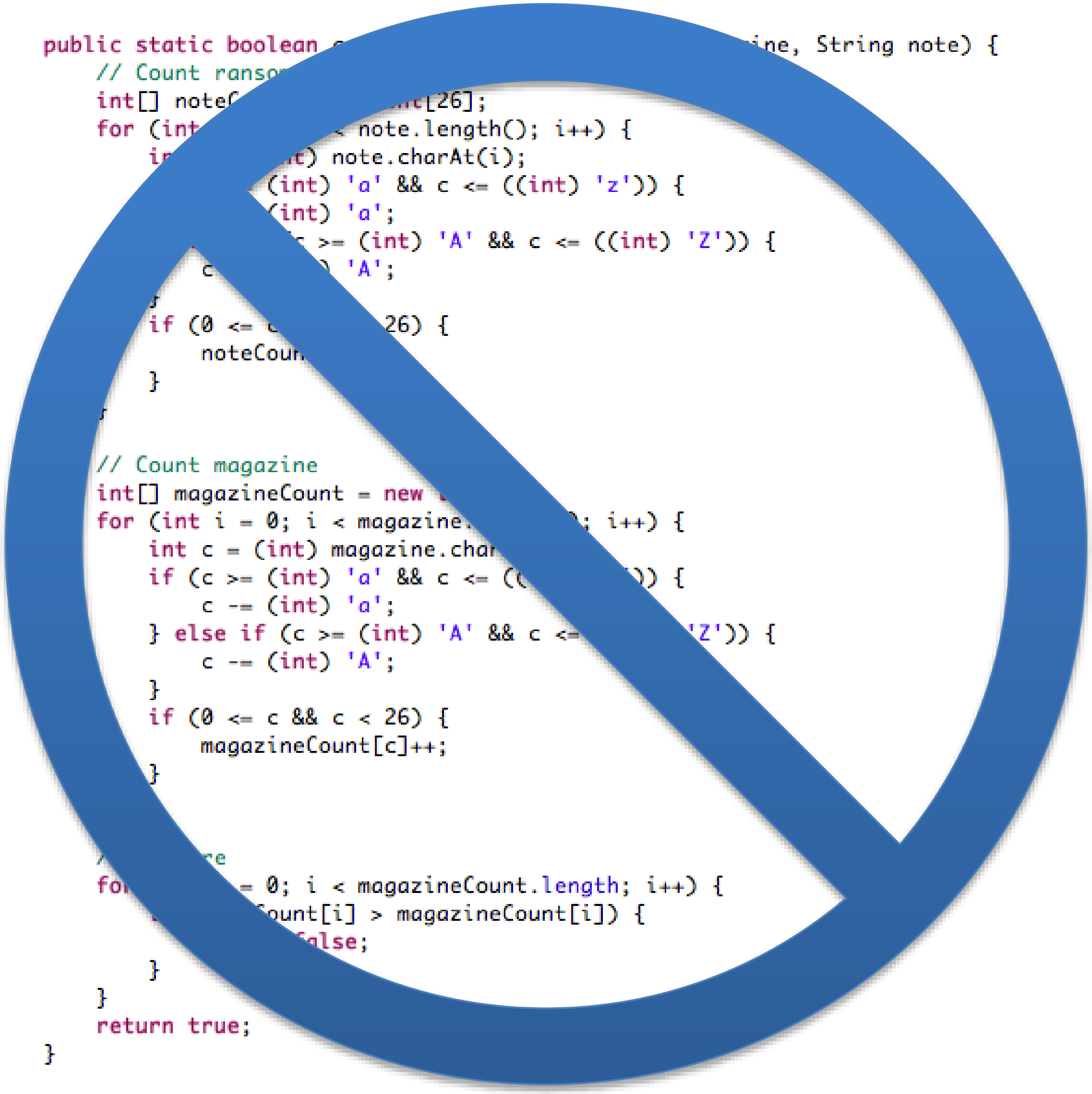
Know the variables

and when they change

step

6

Write Beautiful Code



```
public static boolean                          ine, String note) {
    // Count ranso
    int[] note                 t[26];
    for (int            note.length(); i++) {
        i            t) note.charAt(i);
                (int) 'a' && c <= ((int) 'z')) {
                (int) 'a';
                    c >= (int) 'A' && c <= ((int) 'Z')) {
            c            ) 'A';
        }
        if (0 <=             26) {
            noteCoun
        }
    }

    // Count magazine
    int[] magazineCount = new
    for (int i = 0; i < magazine.           ; i++) {
        int c = (int) magazine.char
        if (c >= (int) 'a' && c <= ((          )) {
            c -= (int) 'a';
        } else if (c >= (int) 'A' && c <=           'Z')) {
            c -= (int) 'A';
        }
        if (0 <= c && c < 26) {
            magazineCount[c]++;
        }


    /          re
    fo          = 0; i < magazineCount.length; i++) {
                  ount[i] > magazineCount[i]) {
                  alse;
        }
    }
    return true;
}
```

step

6

Write Beautiful Code

Real Code

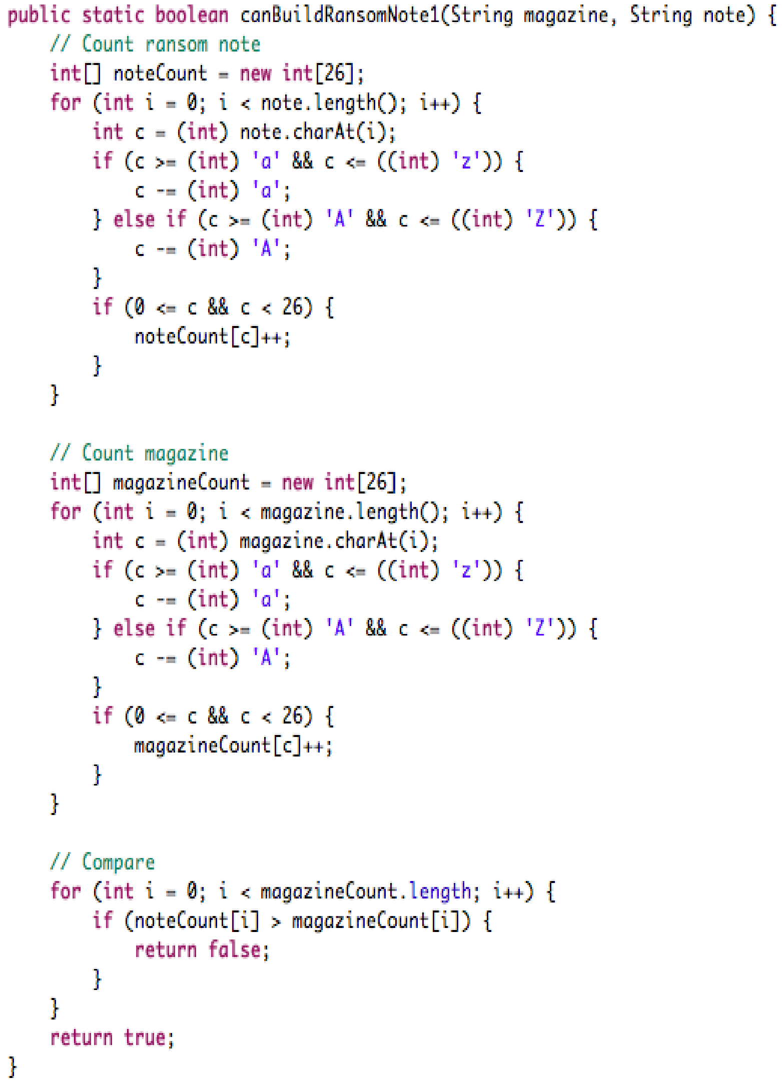
with

Good Style

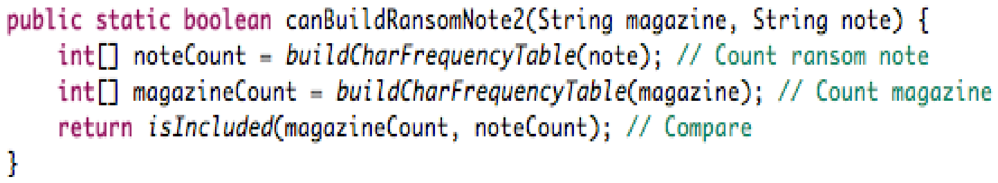
and

Upfront Modularization

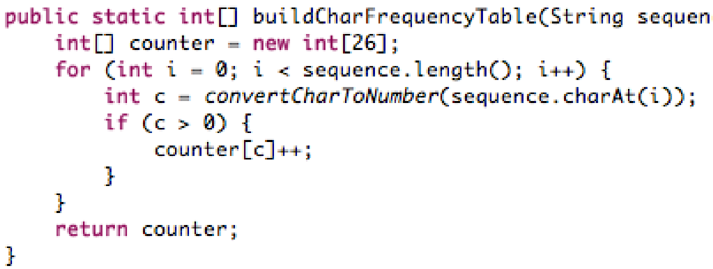
Modularization



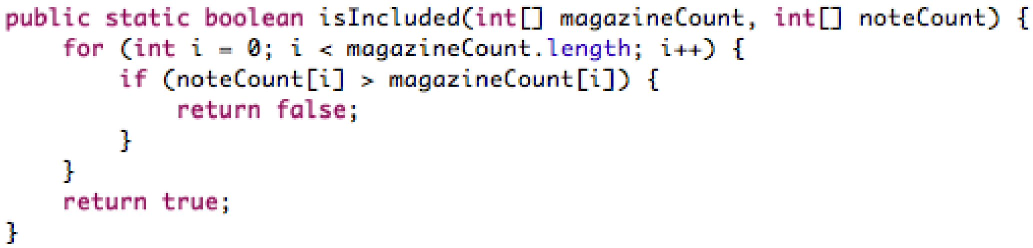
```java
public static boolean canBuildRansomNote1(String magazine, String note) {
    // Count ransom note
    int[] noteCount = new int[26];
    for (int i = 0; i < note.length(); i++) {
        int c = (int) note.charAt(i);
        if (c >= (int) 'a' && c <= ((int) 'z')) {
            c -= (int) 'a';
        } else if (c >= (int) 'A' && c <= ((int) 'Z')) {
            c -= (int) 'A';
        }
        if (0 <= c && c < 26) {
            noteCount[c]++;
        }
    }

    // Count magazine
    int[] magazineCount = new int[26];
    for (int i = 0; i < magazine.length(); i++) {
        int c = (int) magazine.charAt(i);
        if (c >= (int) 'a' && c <= ((int) 'z')) {
            c -= (int) 'a';
        } else if (c >= (int) 'A' && c <= ((int) 'Z')) {
            c -= (int) 'A';
        }
        if (0 <= c && c < 26) {
            magazineCount[c]++;
        }
    }

    // Compare
    for (int i = 0; i < magazineCount.length; i++) {
        if (noteCount[i] > magazineCount[i]) {
            return false;
        }
    }
    return true;
}
```



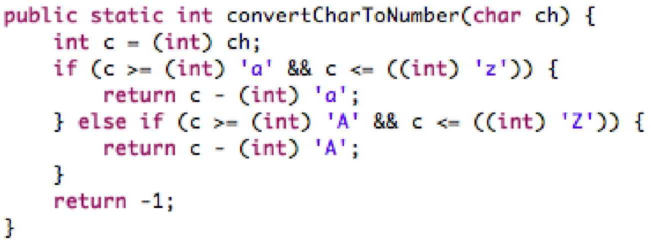
```java
public static boolean canBuildRansomNote2(String magazine, String note) {
    int[] noteCount = buildCharFrequencyTable(note); // Count ransom note
    int[] magazineCount = buildCharFrequencyTable(magazine); // Count magazine
    return isIncluded(magazineCount, noteCount); // Compare
}
```



```java
public static int[] buildCharFrequencyTable(String sequence) {
    int[] counter = new int[26];
    for (int i = 0; i < sequence.length(); i++) {
        int c = convertCharToNumber(sequence.charAt(i));
        if (c > 0) {
            counter[c]++;
        }
    }
    return counter;
}
```

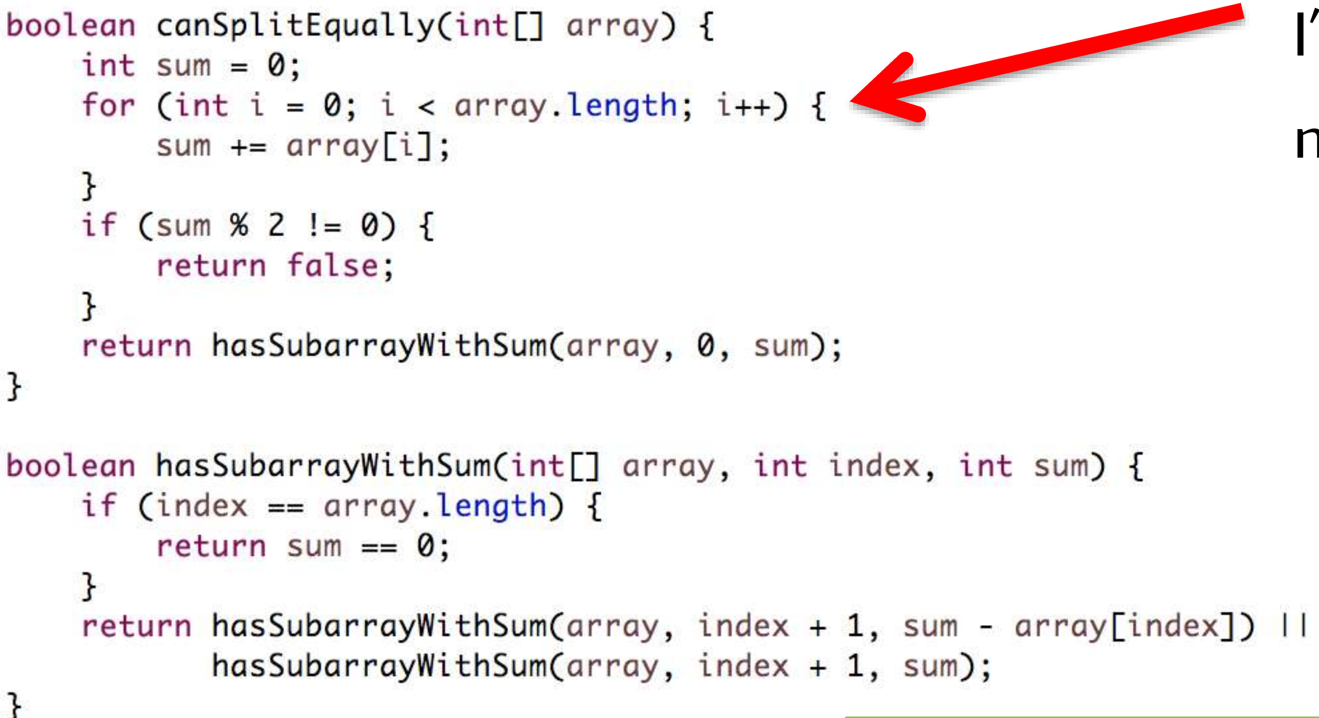


```java
public static boolean isIncluded(int[] magazineCount, int[] noteCount) {
    for (int i = 0; i < magazineCount.length; i++) {
        if (noteCount[i] > magazineCount[i]) {
            return false;
        }
    }
    return true;
}
```



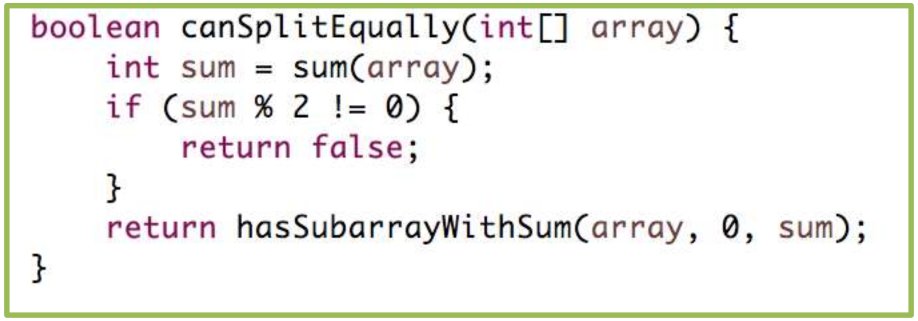
```java
public static int convertCharToNumber(char ch) {
    int c = (int) ch;
    if (c >= (int) 'a' && c <= ((int) 'z')) {
        return c - (int) 'a';
    } else if (c >= (int) 'A' && c <= ((int) 'Z')) {
        return c - (int) 'A';
    }
    return -1;
}
```

Modularize (Upfront!)



```
boolean canSplitEqually(int[] array) {
    int sum = 0;
    for (int i = 0; i < array.length; i++) {
        sum += array[i];
    }
    if (sum % 2 != 0) {
        return false;
    }
    return hasSubarrayWithSum(array, 0, sum);
}

boolean hasSubarrayWithSum(int[] array, int index, int sum) {
    if (index == array.length) {
        return sum == 0;
    }
    return hasSubarrayWithSum(array, index + 1, sum - array[index]) ||
           hasSubarrayWithSum(array, index + 1, sum);
}
```

I’ve learned nothing.



```
boolean canSplitEqually(int[] array) {
    int sum = sum(array);
    if (sum % 2 != 0) {
        return false;
    }
    return hasSubarrayWithSum(array, 0, sum);
}
```

step

7

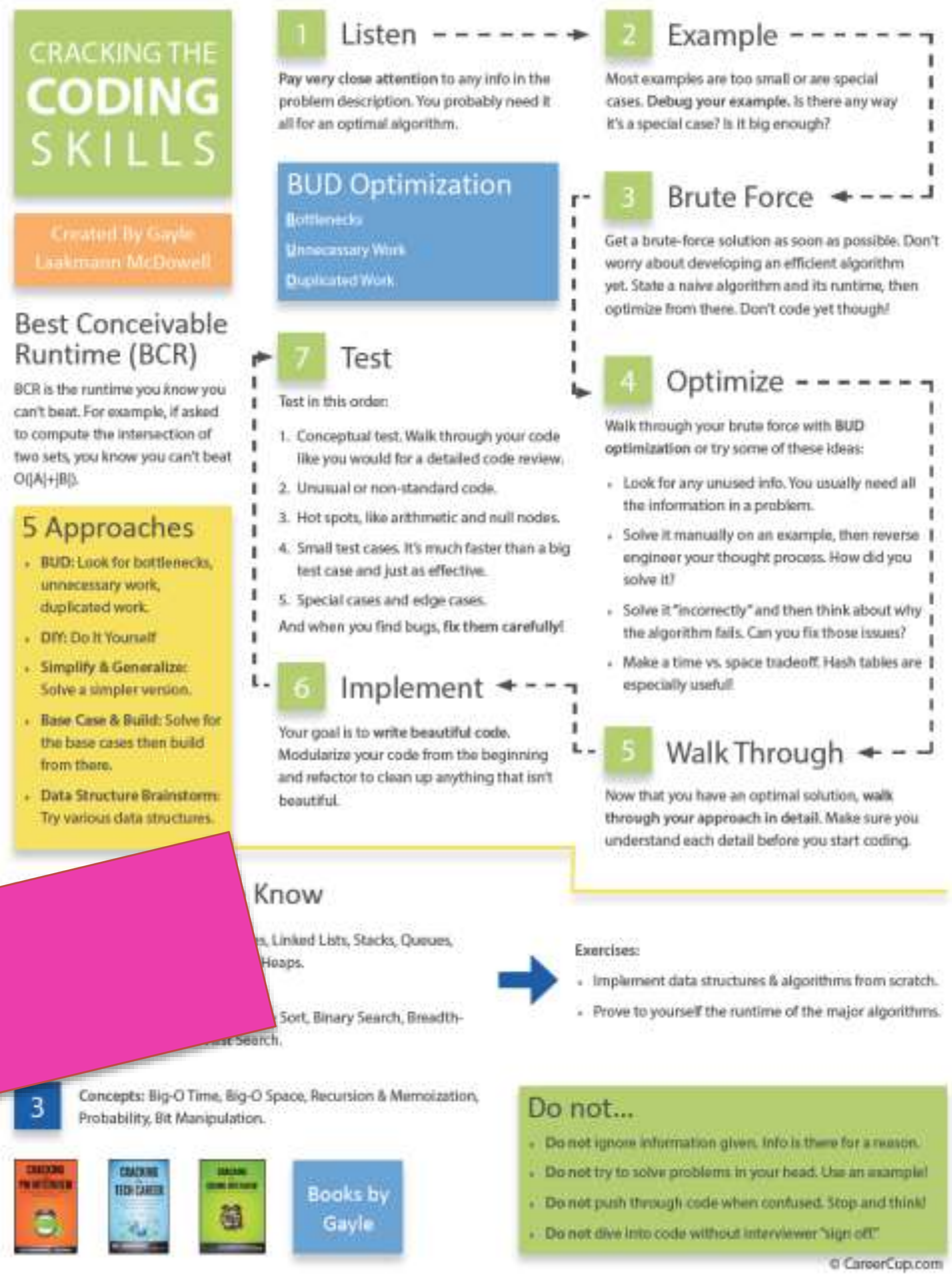
Testing

- FIRST Analyze
  - What's it doing? Why?
  - Anything that looks weird?
  - Error hot spots
- THEN use test cases
  - Small test cases
  - Edge cases
  - Bigger test cases
- BUT…
  - Test code, not algorithm
  - Think as you test
  - Think before you fix

How

To

Approach



CRACKING THE CODING SKILLS

Created By Gayle Laakmann McDowell

1 Listen

Pay very close attention to any info in the problem description. You probably need it all for an optimal algorithm.

2 Example

Most examples are too small or are special cases. **Debug your example.** Is there any way it's a special case? Is it big enough?

3 Brute Force

Get a brute-force solution as soon as possible. Don't worry about developing an efficient algorithm yet. State a naive algorithm and its runtime, then optimize from there. Don't code yet though!

4 Optimize

Walk through your brute force with **BUD optimization** or try some of these ideas:

- Look for any unused info. You usually need all the information in a problem.
- Solve it manually on an example, then reverse engineer your thought process. How did you solve it?
- Solve it "incorrectly" and then think about why the algorithm fails. Can you fix those issues?
- Make a time vs. space tradeoff. Hash tables are especially useful!

5 Walk Through

Now that you have an optimal solution, **walk through your approach in detail.** Make sure you understand each detail before you start coding.

6 Implement

Your goal is to **write beautiful code.** Modularize your code from the beginning and refactor to clean up anything that isn't beautiful.

7 Test

Test in this order:

1. Conceptual test. Walk through your code like you would for a detailed code review.
2. Unusual or non-standard code.
3. Hot spots, like arithmetic and null nodes.
4. Small test cases. It's much faster than a big test case and just as effective.
5. Special cases and edge cases.

And when you find bugs, **fix them carefully!**

BUD Optimization

Bottlenecks
Unnecessary Work
Duplicated Work

Best Conceivable Runtime (BCR)

BCR is the runtime you know you can't beat. For example, if asked to compute the intersection of two sets, you know you can't beat O(|A|+|B|).

5 Approaches

- **BUD:** Look for bottlenecks, unnecessary work, duplicated work.
- **DIY:** Do It Yourself
- **Simplify & Generalize:** Solve a simpler version.
- **Base Case & Build:** Solve for the base cases then build from there.
- **Data Structure Brainstorm:** Try various data structures.

Know

s, Linked Lists, Stacks, Queues, Heaps.

Sort, Binary Search, Breadth- Search.

3 Concepts: Big-O Time, Big-O Space, Recursion & Memoization, Probability, Bit Manipulation.

Exercises:

- Implement data structures & algorithms from scratch.
- Prove to yourself the runtime of the major algorithms.

Do not...

- Do not ignore information given. Info is there for a reason.
- Do not try to solve problems in your head. Use an example!
- Do not push through code when confused. Stop and think!
- Do not dive into code without interviewer "sign off."

Books by Gayle

© CareerCup.com

Email: g@gayle.com
Subject: mockathon2016

CrackingTheCodingInterview.com → "Resources"

Email: g@gayle.com
Subject: mockathon2016

04

Solving & Optimizing

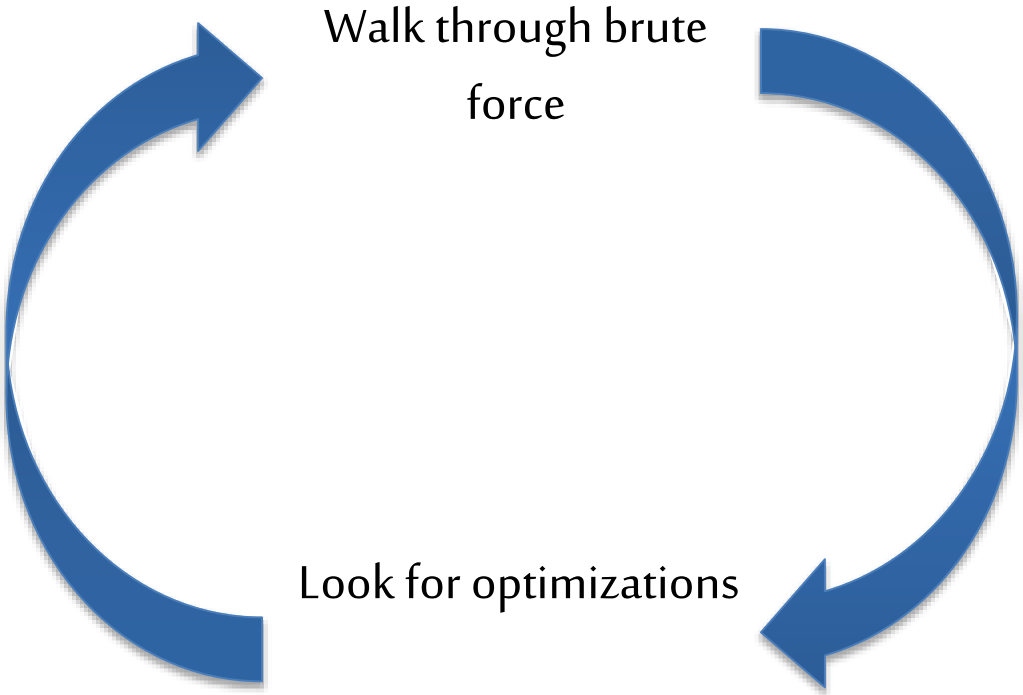
4 Optimization/Solving Techniques

careercup.com

step

4

Optimize



Walk through brute force

Look for optimizations

Techniques to Develop Algorithms

- BUD
- Space and Time
- Do It Yourself
- Recursion

Push yourself!

(A) Look for BUD

- Bottlenecks
- Unnecessary work
- Duplicated work

What’s the bottleneck?

- Ex: counting the intersection

  [1, 12, 15, 19, 20, 21]

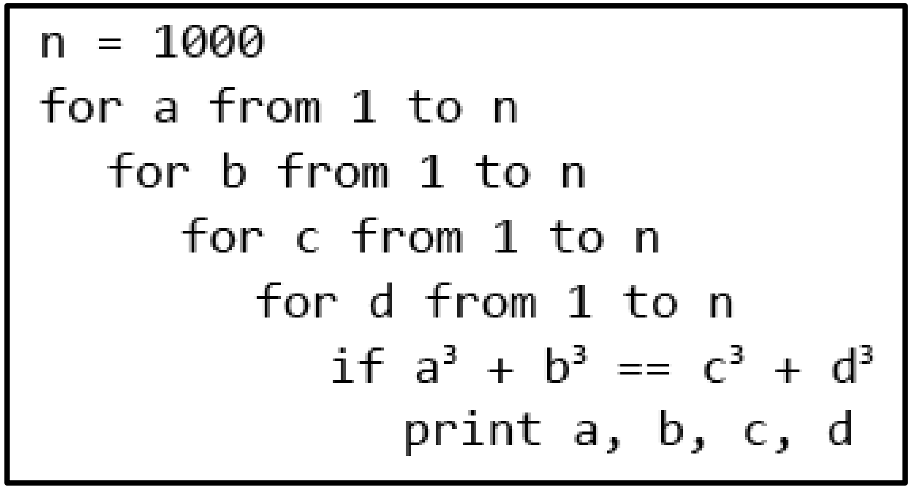
  [2, 15, 17, 19, 21, 25, 27]

- Bottleneck: searching

B

What’s unnecessary?

- Ex: $a^3 + b^3 = c^3 + d^3$ (1 <= a, b, c, d <= 1000

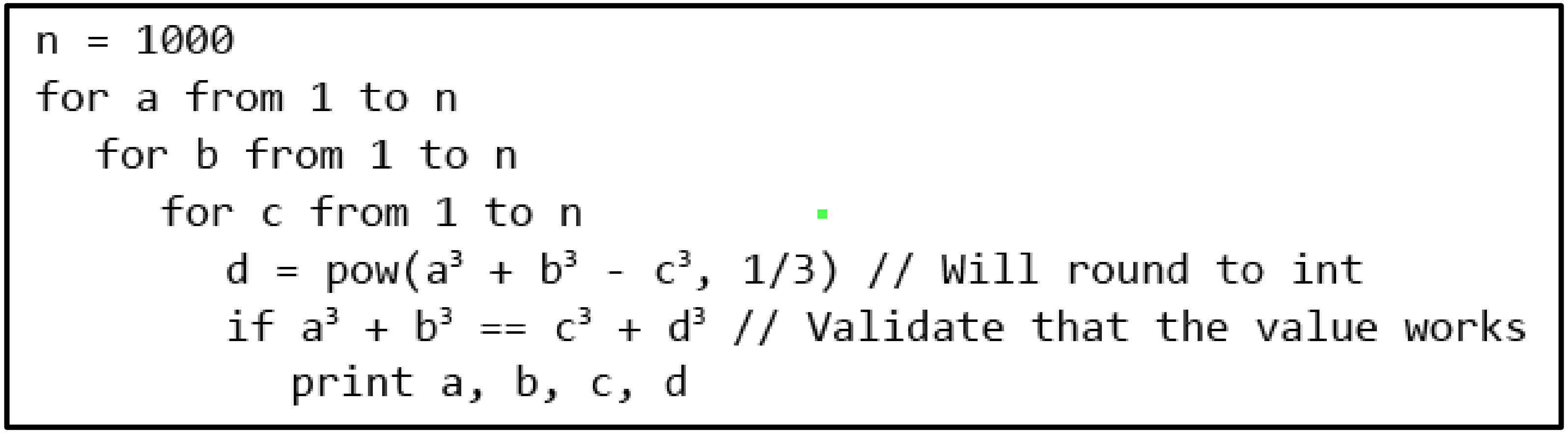


```
n = 1000
for a from 1 to n
   for b from 1 to n
      for c from 1 to n
         for d from 1 to n
            if $a^3 + b^3 == c^3 + d^3$
               print a, b, c, d
```

- Unnecessary: looking for d

What’s unnecessary?

- Ex: $a^3 + b^3 = c^3 + d^3$ (1 <= a, b, c, d <= 1000

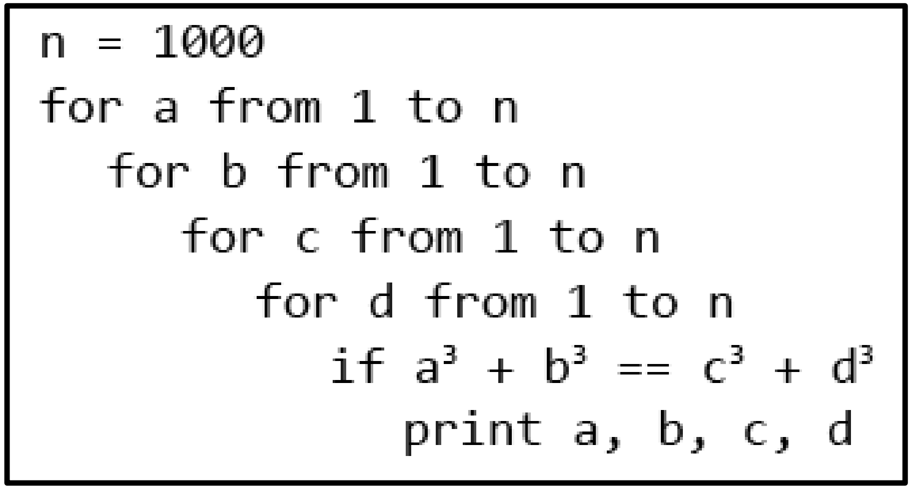


```
n = 1000
for a from 1 to n
   for b from 1 to n
      for c from 1 to n
         d = pow($a^3 + b^3 - c^3$, 1/3) // Will round to int
         if $a^3 + b^3 == c^3 + d^3$ // Validate that the value works
            print a, b, c, d
```

- Unnecessary: looking for d

What’s duplicated?

- Ex: $a^3 + b^3 = c^3 + d^3$ (1 <= a, b, c, d <= 1000



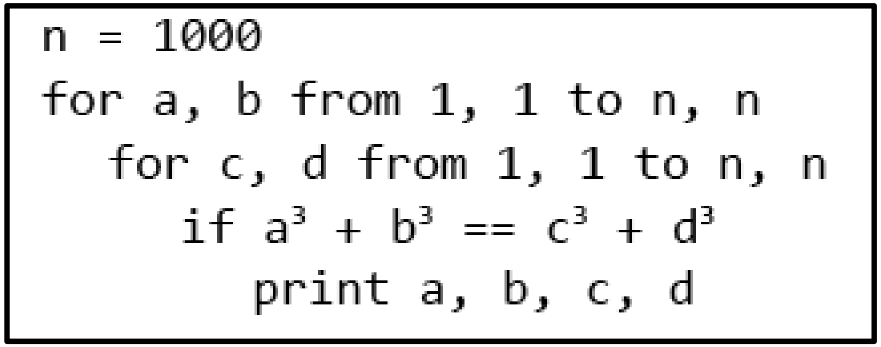
```
n = 1000
for a from 1 to n
    for b from 1 to n
        for c from 1 to n
            for d from 1 to n
                if $a^3 + b^3 == c^3 + d^3$
                    print a, b, c, d
```

- Duplicated: c, d pairs

D

What's duplicated?

- Ex: $a^3 + b^3 = c^3 + d^3$ (1 <= a, b, c, d <= 1000



```
n = 1000
for a, b from 1, 1 to n, n
    for c, d from 1, 1 to n, n
        if $a^3 + b^3 == c^3 + d^3$
            print a, b, c, d
```

c	d	$c^3 + d^3$
…	…	…
4	31	29855
4	32	32832
4	33	36001
…	…	…
5	59	205504
5	60	216125
5	61	227106
…	…	…

- Duplicated: c, d pairs

D

What’s duplicated?

- Ex: $a^3 + b^3 = c^3 + d^3$ (1 <= a, b, c, d <= 1000

```
n = 1000
for a, b from 1, 1 to n, n
  for c, d from 1, 1 to n, n
    if $a^3 + b^3 == c^3 + d^3$
      print a, b, c, d
```

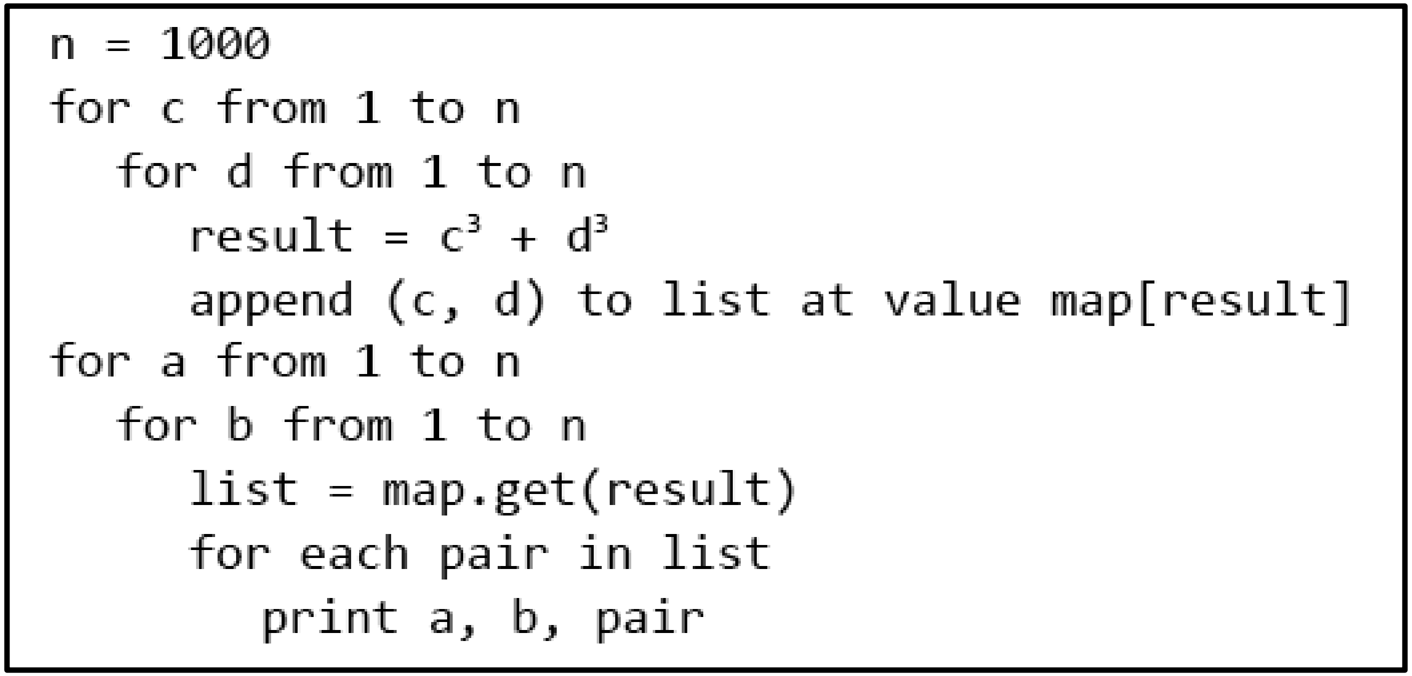
$c^3 + d^3$	(c, d)
…	…
29855	(4, 31)
32832	(4, 32), (18, 30)
36001	(4, 33)
…	…
205504	(5, 59)
216125	(5, 60), (45, 50)
227106	(5, 61)
…	…

- Duplicated: c, d pairs

D

What’s duplicated?

- Ex: $a^3 + b^3 = c^3 + d^3$ (1 <= a, b, c, d <= 1000

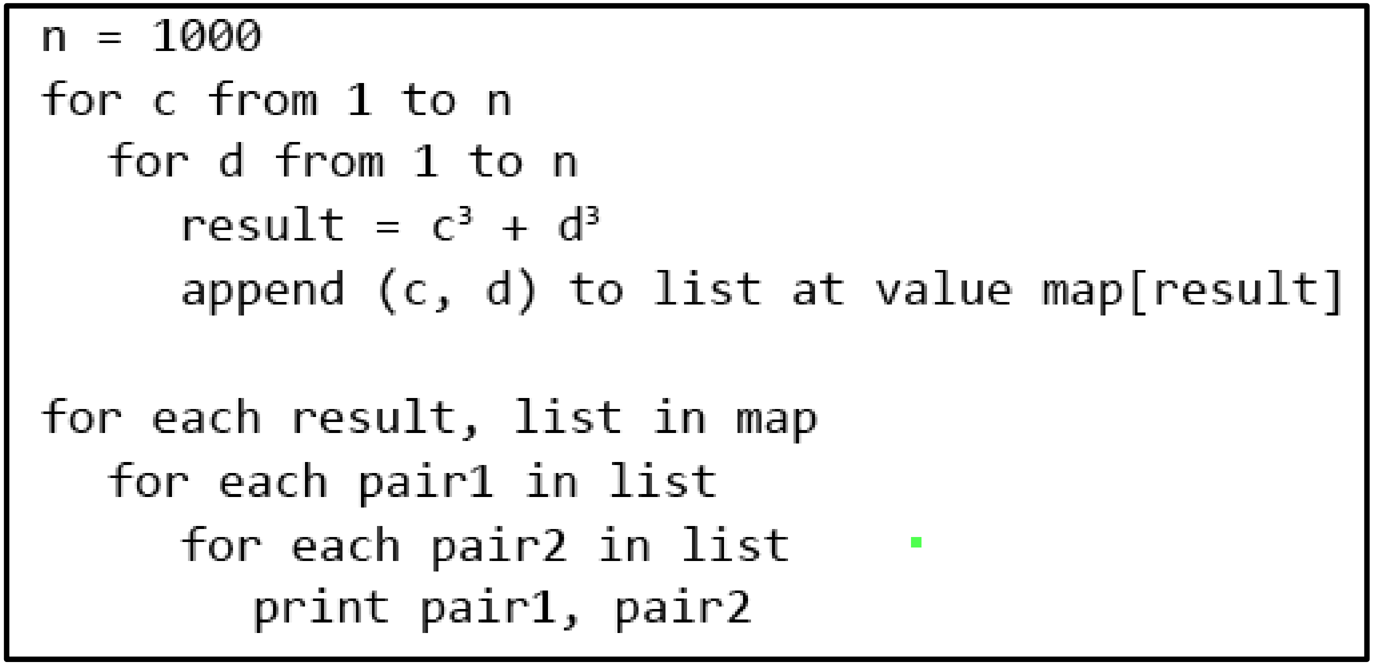


```
n = 1000
for c from 1 to n
   for d from 1 to n
      result = $c^3 + d^3$
      append (c, d) to list at value map[result]
for a from 1 to n
   for b from 1 to n
      list = map.get(result)
      for each pair in list
         print a, b, pair
```

D

What’s duplicated?

- Ex: $a^3 + b^3 = c^3 + d^3$ $(1 <= a, b, c, d <= 1000$



```
n = 1000
for c from 1 to n
   for d from 1 to n
      result = $c^3 + d^3$
      append (c, d) to list at value map[result]

for each result, list in map
   for each pair1 in list
      for each pair2 in list
         print pair1, pair2
```

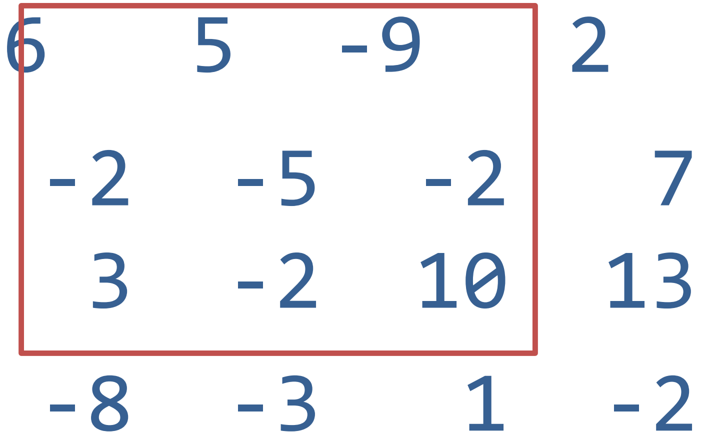
D

(B) Space/Time Tradeoffs

- Hash tables & other data structures
- Precomputing

Space/Time Tradeoffs → Precomputing

- Find rectangle at origin w biggest sum

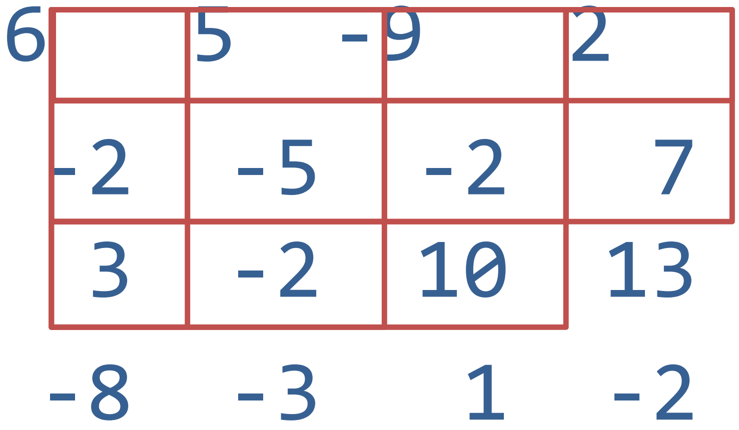


6	5	-9	2
-2	-5	-2	7
3	-2	10	13
-8	-3	1	-2

- Brute force: compute all rectangles and sums

Space/Time Tradeoffs → Precomputing

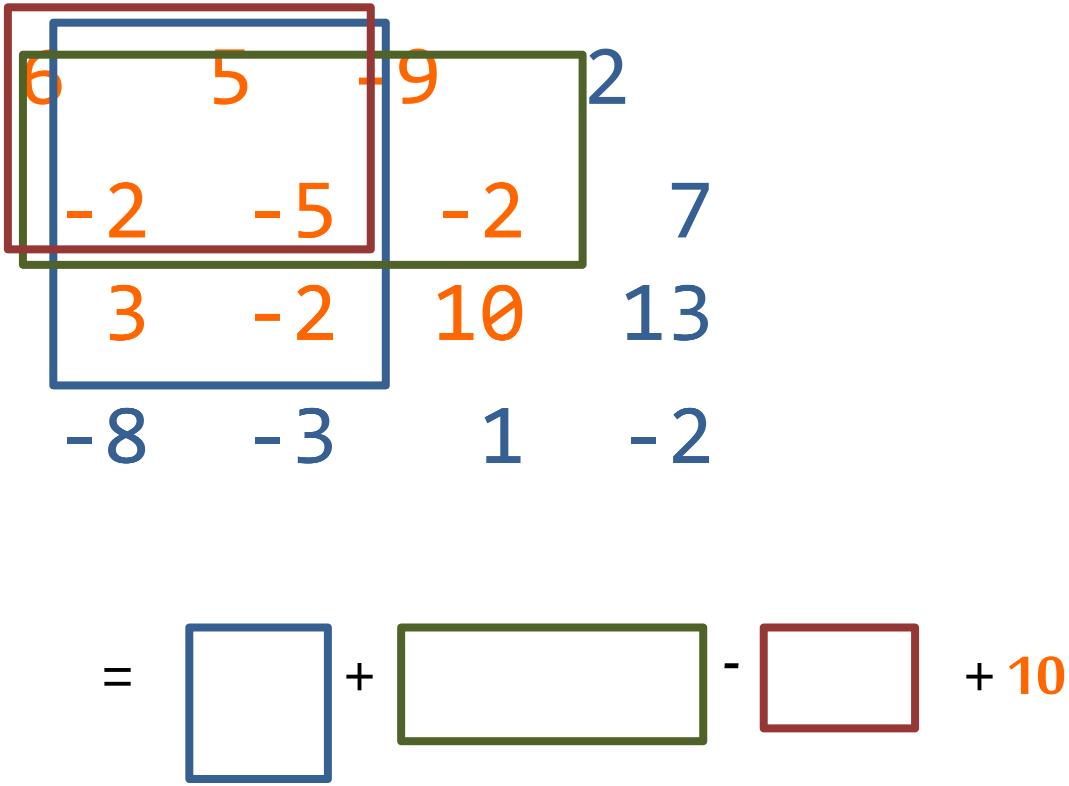
- Find rectangle at origin w biggest sum



6	5	-9	2
-2	-5	-2	7
3	-2	10	13
-8	-3	1	-2

Space/Time Tradeoffs → Precomputing

- Find rectangle with biggest sum

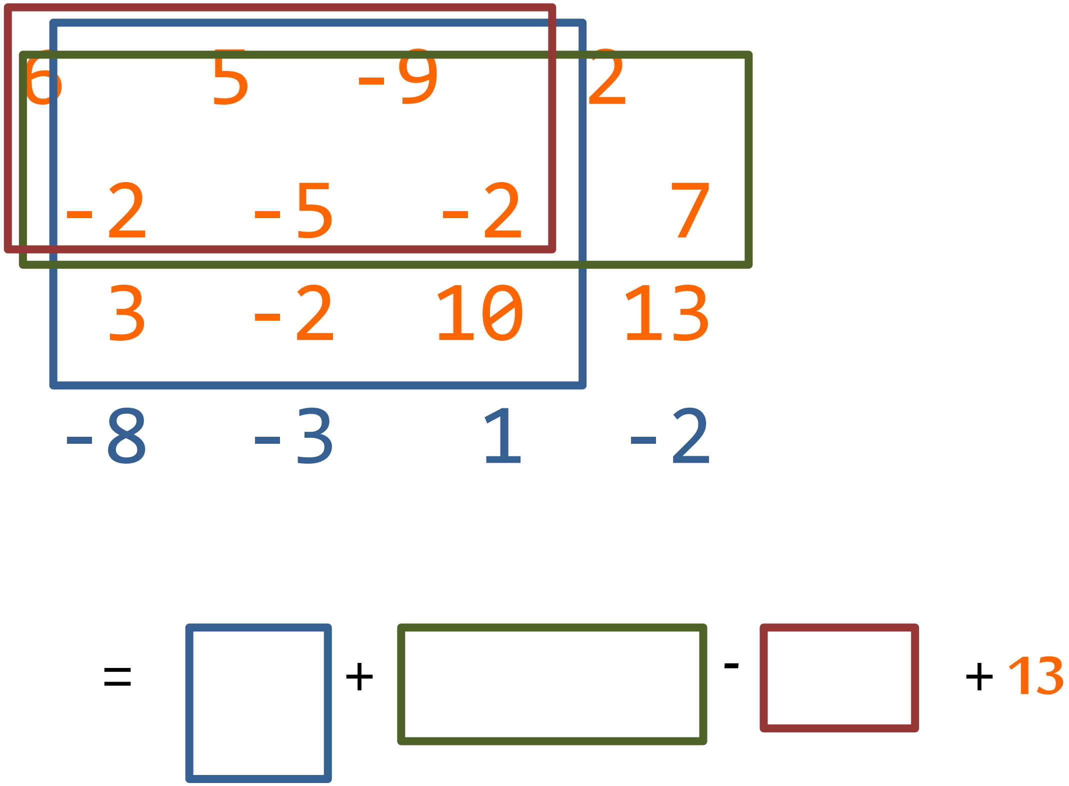


6	5	-9	2
-2	-5	-2	7
3	-2	10	13
-8	-3	1	-2

= □ + □ - □ + 10

Space/Time Tradeoffs → Precomputing

- Find rectangle with biggest sum



6	5	-9	2
-2	-5	-2	7
3	-2	10	13
-8	-3	1	-2

= □ + □ - □ + 13

(C) Do it yourself

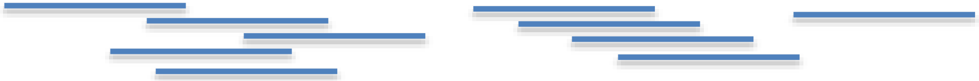
- Find permutations of *s* within *b*

find abbc in

babcabbacaabcbabcacbb

(C) Do it yourself

- Find permutations of *s* within *b*
  - s = abbc
  - b = babcabbacaabcbabcacbb



- Find them!
  - … now how did you *actually* do it?

(D) Recursion

- Use, but don’t cling to, recursion “instinct”
- Try bottom-up
- “Backtracking”
- Draw call-tree
  - Derive runtime
  - Find repeated subproblems

- Subsets of a set
  - {} → {}
  - {a} → {}, {a}
  - {a, b} → {}, {a}, {b}, {a, b}
  - {a, b, c} → …
- Subsets of $\{S_1 \ldots S_{n-1}\} + S_n$ to each

Techniques to Develop Algorithms

- BUD
- Space and Time
- Do It Yourself
- Recursion

Push yourself!

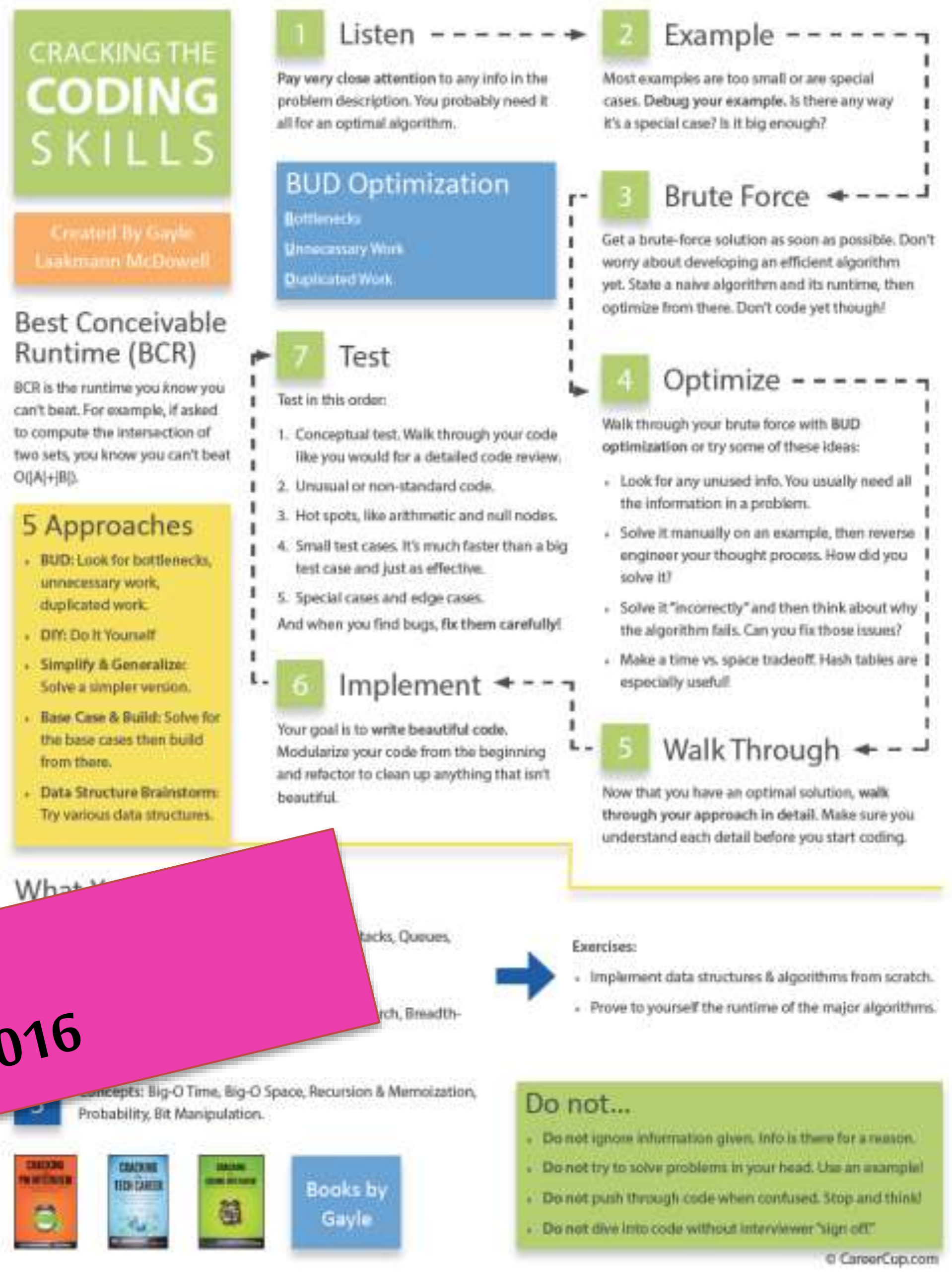
How

To

Approach

Email: g@gayle.com

Subject: mockathon2016



CRACKING THE CODING SKILLS

Created By Gayle Laakmann McDowell

Best Conceivable Runtime (BCR)

BCR is the runtime you know you can't beat. For example, if asked to compute the intersection of two sets, you know you can't beat O(|A|+|B|).

5 Approaches

- **BUD:** Look for bottlenecks, unnecessary work, duplicated work.
- **DIY:** Do It Yourself
- **Simplify & Generalize:** Solve a simpler version.
- **Base Case & Build:** Solve for the base cases then build from there.
- **Data Structure Brainstorm:** Try various data structures.

1 Listen

Pay very close attention to any info in the problem description. You probably need it all for an optimal algorithm.

BUD Optimization

Bottlenecks

Unnecassary Work

Duplicated Work

2 Example

Most examples are too small or are special cases. **Debug your example.** Is there any way it's a special case? Is it big enough?

3 Brute Force

Get a brute-force solution as soon as possible. Don't worry about developing an efficient algorithm yet. State a naive algorithm and its runtime, then optimize from there. Don't code yet though!

4 Optimize

Walk through your brute force with **BUD optimization** or try some of these ideas:

- Look for any unused info. You usually need all the information in a problem.
- Solve it manually on an example, then reverse engineer your thought process. How did you solve it?
- Solve it "incorrectly" and then think about why the algorithm fails. Can you fix those issues?
- Make a time vs. space tradeoff. Hash tables are especially useful!

5 Walk Through

Now that you have an optimal solution, **walk through your approach in detail.** Make sure you understand each detail before you start coding.

6 Implement

Your goal is to **write beautiful code.** Modularize your code from the beginning and refactor to clean up anything that isn't beautiful.

7 Test

Test in this order:

1. Conceptual test. Walk through your code like you would for a detailed code review.
2. Unusual or non-standard code.
3. Hot spots, like arithmetic and null nodes.
4. Small test cases. It's much faster than a big test case and just as effective.
5. Special cases and edge cases.

And when you find bugs, **fix them carefully!**

What Y

tacks, Queues,

rch, Breadth-

oncepts: Big-O Time, Big-O Space, Recursion & Memoization, Probability, Bit Manipulation.

Exercises:

- Implement data structures & algorithms from scratch.
- Prove to yourself the runtime of the major algorithms.

Books by Gayle

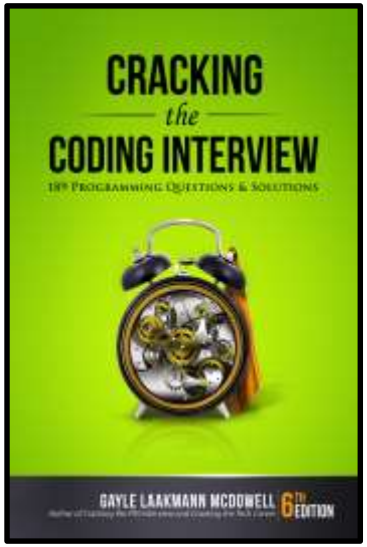
Do not...

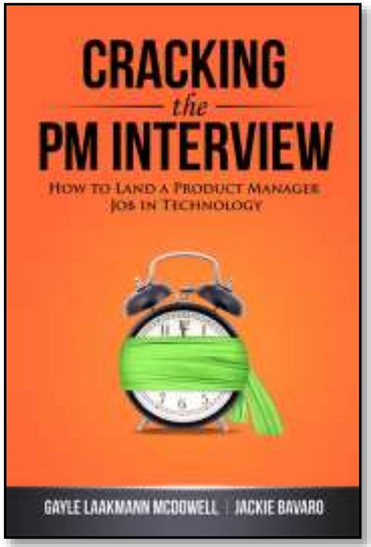
- Do not ignore information given. Info is there for a reason.
- Do not try to solve problems in your head. Use an example!
- Do not push through code when confused. Stop and think!
- Do not dive into code without interviewer "sign off."

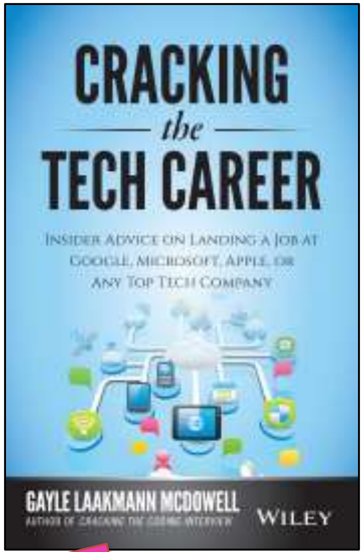
© CareerCup.com

CrackingTheCodingInterview.com → "Resources"

Other Resources


CRACKING the CODING INTERVIEW
189 PROGRAMMING QUESTIONS & SOLUTIONS
GAYLE LAAKMANN MCDOWELL 6TH EDITION


CRACKING the PM INTERVIEW
HOW TO LAND A PRODUCT MANAGER JOB IN TECHNOLOGY
GAYLE LAAKMANN MCDOWELL | JACKIE BAVARO


CRACKING the TECH CAREER
INSIDER ADVICE ON LANDING A JOB AT GOOGLE, MICROSOFT, APPLE, OR ANY TOP TECH COMPANY
GAYLE LAAKMANN MCDOWELL
AUTHOR OF CRACKING THE CODING INTERVIEW
WILEY

CRACKING THE CODING SKILLS
CRACKING THE PRODUCT MANAGER SKILLS

Email: g@gayle.com

Subject: mockathon2016

Gayle.com

CareerCup.com

CrackingTheCodingInterview.com

Or, follow me online

- facebook.com/**gayle**
- twitter.com/**gayle**
- **gayle**.com
- **gayle**@gayle.com
- quora.com

What Now?

- Book signing, photos, etc. [with me!]
- Mock interviews [with AWS!]
- Code challenge [online!]

  - **hr.gs/mockathon**

Email: **g@gayle.com**
Subject: **mockathon2016**