Pre-announcements

Data Science Society @ Berkeley

- Berkeley Data Science Forum 2018
- Pinterest, Spotify, etc. will be there.

CS70 Decal:

- http://imt-decal.org
- CS70 is hard. This decal helps you prepare.
- Everything is online over the summer if you want to prepare to prepare.
- Don't take WITH 70, but before 70.

Announcements

Makeup HW6 and HW7 will be out as soon as possible

- Can replace up to two old HWs.
- HW7 is easier.

HKN Survey on Friday in class:

- 4 points if you fill out the HKN survey.
- You can also do survey at home if you can't make it to the physical lecture.

Post-final exam survey will be worth 8 extra credit points.

Many of today's topics are "extra". We will not assume knowledge of these topics for the final exam, but we might include problems inspired by these topics.

CS61B

Lecture 39: Impossible and Intractable Problems (170 Preview)

- Kolmogorov Complexity (Extra)
- Problem Difficulty: The Independent Set Problem
- Reductions
- Complexity Classes (Extra)
- P=NP (Extra)

HugPlant

Huffman Coding can be used to compress any data, not just text. In bitmap format, the plant below is simply the stream of bits shown on the right.

Original Uncompressed Bits B

```
42 4d 7a 00 10 00 00 00 00
00 7a 00 00 00 6c 00 00 00
00 02 00 00 00 02 00 00 01
00 20 00 03 00 00 00 00 00
10 00 12 0b 00 00 12 0b 00
00 00 00 00 00 00 00 00 00
00 00 ff 00 00 ff 00 00 ff
00 00 00 00 00 00 ff 01 00
00 00 00 00 00 00 00 00 00
00 01 00 00 00 00 00 00 00
00 00 00 00 01 00 ...
```

Total: 8389584 bits

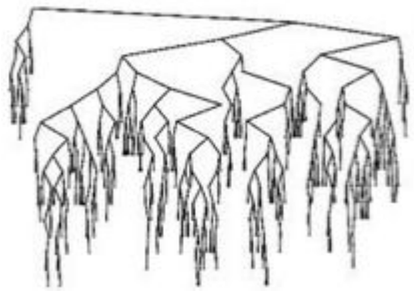
HugPlant Compressed

```
42 4d 7a 00 10 00 00 00 00 00 7a 00 00 00 6c 00 00 00 00
02 00 00 00 02 00 00 01 00 20 00 03 00 00 00 00 00 10 00
12 0b 00 00 12 0b 00 00 00 00 00 00 00 00 00 00 00 00 ff
00 00 ff 00 00 ff 00 00 00 00 00 00 ff 01 00 00 00 00 00
00 00 00 00 00 00 01 00 00 00 00 00 00 00 00 00 00 00 01
00 ...
```

Total: 8389584 bits

Huffman.java
compress()

```
74 68 65 20 70 61 73 73 63 6F 64 65 20 69 73 20
68 75 67 39 31 38 32 37 78 79 7A 2E 65 75 7a c0
09 eb cd d4 2a 55 9f d8 98 d1 4e e7 97 56 58 68
0c 7a 43                                7 bc 26 01
e0 92 28                                f 0e 87 bd
87 9e 03                                1 b5 79 13
9b 95 d5                                9 c0 e6 53
c7 cb dd                                a 74 34 3a
a9 c1 ca                                4 3a 5c 5a
62 e9 2f                                7 4e 50 be
c0 15 1b                                4 e2 23 a2
b6 84 22                                2 0f 2d bd
e5 81 f4 c6 de 15 59 f1 68 a4 a5 88 16 b0 7f bf
8a 1d 98                                   ce 12
57 23 62                                   d6 9b
12 49 62                                   dd e3
dc 1c 7f c4 a4 69 77 6e 5e 60 db 5a 69 01 95 c8
d7 2e 57 62 b7 8e 5c 51 f9 70 55 1b 7c ba 68 bc
```



Coding Trie: 2560 bits

Total: 1994024 bits

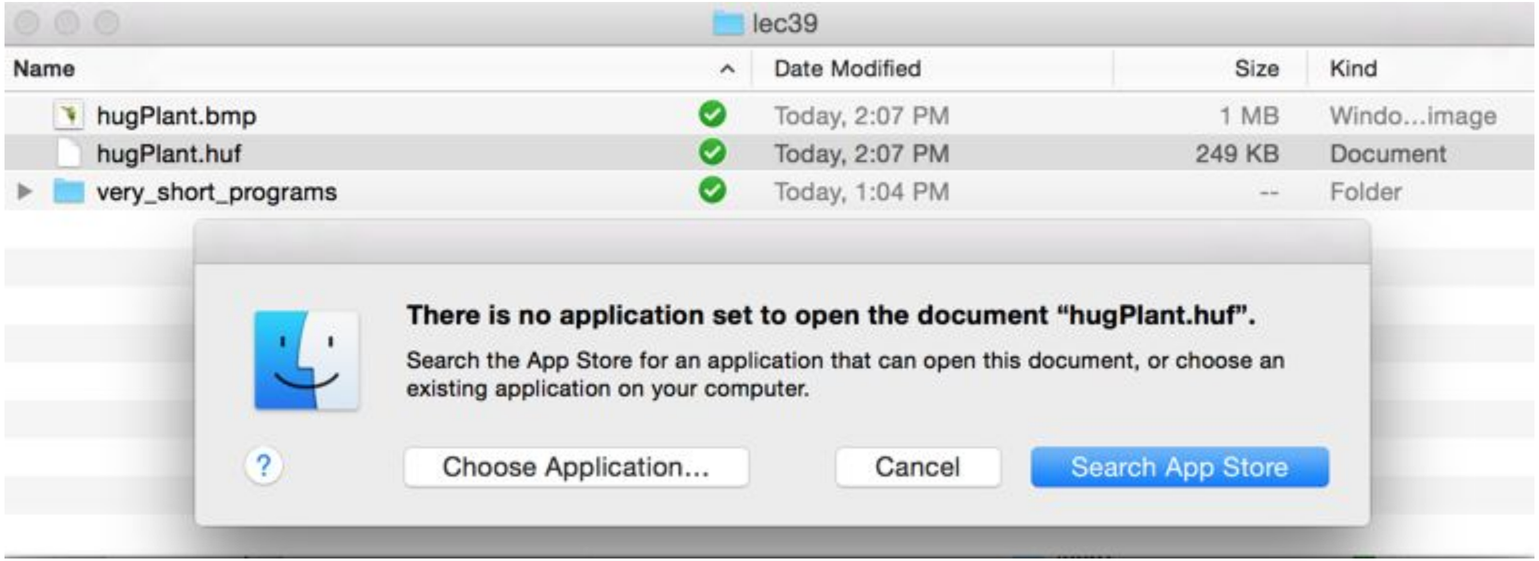
```
00 20 00 f4 c3 b7 6d c2 31 24 92 dc 24 a7 c9 25 ae 24
b5 c4 85 88 40 be c4 92 46 25 79 2f c4 af 25 f8 92 49
24 92 64 c9 92 49 30 b1 24 92 49 24 2c 49 24 92 49 0b
12 49 24 92 42 c4 92 49 24 92 49 ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff                                         f ff
ff ff ff                                         f ff
ff ff ff                                         f ff
ff ff ff                                         f ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ...
```

Image data: 1991464 bits

Opening a .huf File

Of course, major operating systems have no idea what to do with a .huf file.

- Have to send over the 43,400 bits of Huffman.java code as well.
- Total size (including .java file): 2,037,424 bits.



lec39

Name	Date Modified	Size	Kind
hugPlant.bmp	Today, 2:07 PM	1 MB	Windo...image
hugPlant.huf	Today, 2:07 PM	249 KB	Document
very_short_programs	Today, 1:04 PM	--	Folder

There is no application set to open the document “hugPlant.huf”.

Search the App Store for an application that can open this document, or choose an existing application on your computer.

Or an Even Simpler View

To keep things conceptually simpler, let’s package the compressed data plus decoder into a single self-extracting .java file.

- Bitstream on the left generates bitstream on the right.

`SelfExtractingHugPlant.java`

```
70 75 62 6c 69 63 20 63 6c 61 73 73 20 53 65 6c
66 45 78 74 72 61 63 74 69 6e 67 48 75 67 50 6c
61 6e 74 20 7b 0d 0a ...    74 68 65 20 70 61 73
73 63 6F 64 65 20 69 73 20 68 75 67 39 31 38 32
37 78 79 7A 2E 65 75 7a c0 09 eb cd d4 2a 55 9f
d8 98 d1 4e e7 97 56 58 68 0c 7a 43 dd 80 00 7b
11 58 f4 75 73 77 bc 26 01 e0 92 28 ef 47 24 66
9b de 8b 25 04 1f 0e 87 bd 87 9e 03 c9 f1 cf ad
fa 82 dc 9f a1 31 b5 79 13 9b 95 d5 63 26 8b 90
5e d5 b0 17 fb e9 c0 e6 53 c7 cb dd 5f 77 d3 bd
80 f9 b6 5e 94 aa 74 34 3a ...    00 20 00 f4 c3
b7 6d c2 31 24 92 dc 24 a7 c9 25 ae 24 b5 c4 85
88 40 be c4 92 46 25 79 2f c4 af 25 f8 92 49 24
92 64 c9 92                          f ff ff ff
ff ff ff ff                          f ff ff ff
ff ff ff ff                          f ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ...
```

2,037,424 bits

`HugPlant.bmp`

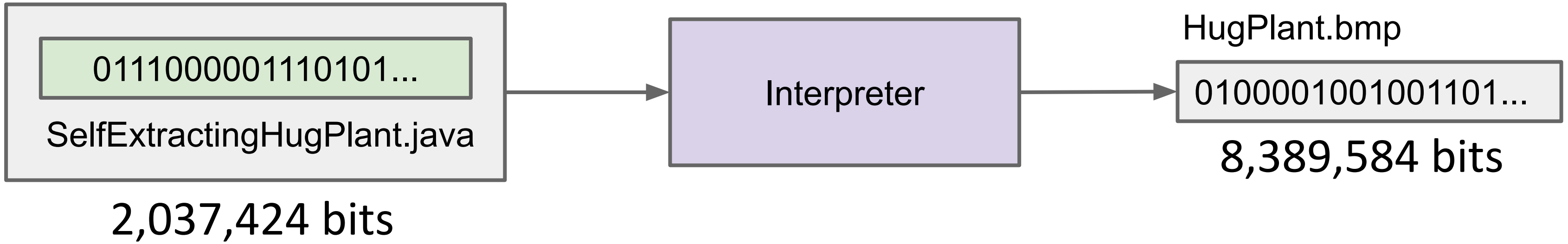
```
42 4d 7a 00 10 00 00 00 00 00 7a 00 00 00 6c 00 00 00
00 02 00 00 00 02 00 00 01 00 20 00 03 00 00 00 00 00
10 00 12 0b 00 00 12 0b 00 00 00 00 00 00 00 00 00 00
00 00 ff 00 00 ff 00 00 ff 00 00 00 00 00 00 ff 01 00
00 00 00 00 00 00 00 00 00 00 01 00 00 00 00 00 00 00
00 00 00 00 01 00 00 00 00 00 00 00 00 00 00 00 01 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff f                      f ff ff ff ff
ff ff ff ff ff f                      f ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff...
```

8,389,584 bits

Compression Model 2: Self-Extracting Bits

As a model for the decompression process, let’s **treat the algorithm and the compressed bitstream as a single sequence of bits.**

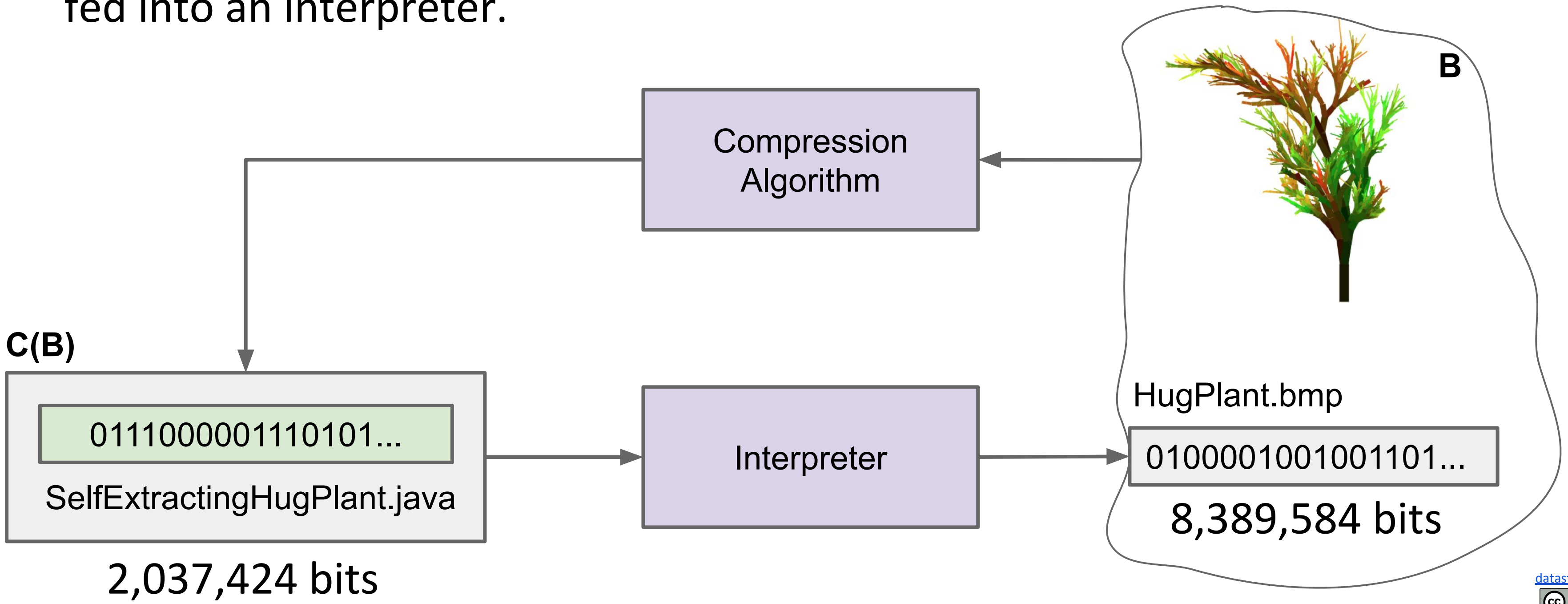
- One example: SelfExtractingHugPlant.java.
    - Java file contained Huffman encoding of HugPlant.bmp, along with code needed to decompress this encoding.



Compression Model 2: Self-Extracting Bits

The goal of a compression algorithm is to find short sequences of bits that generate desired longer sequences of bits.

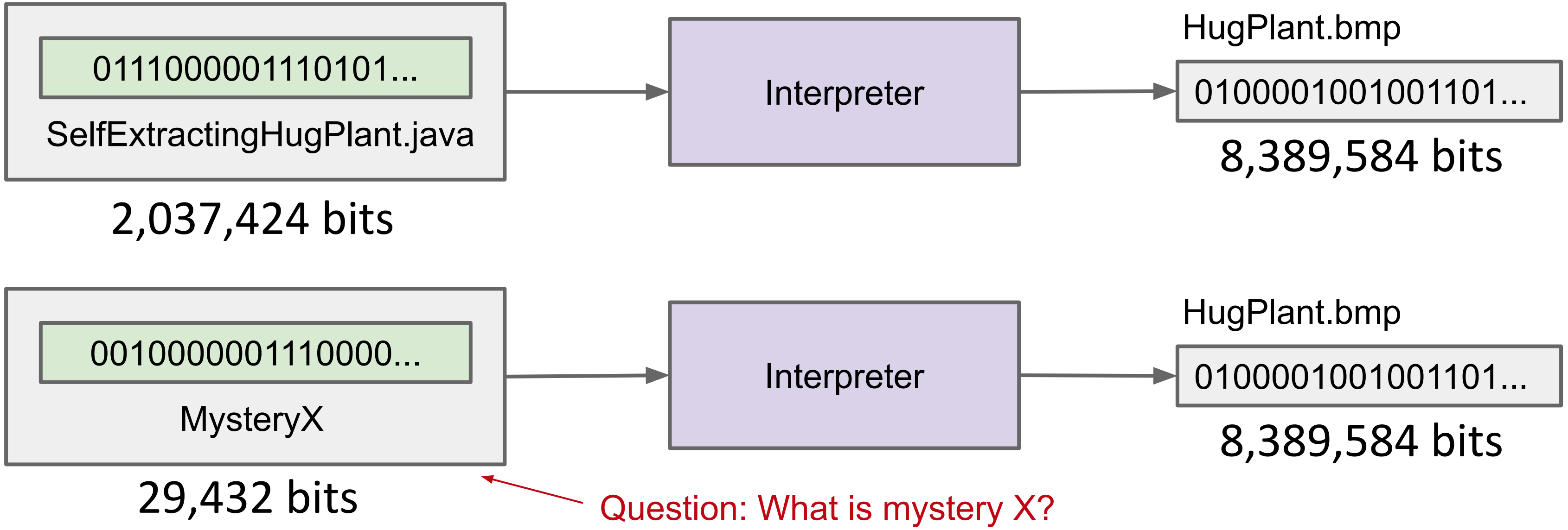
- Given a sequence of bits B, find a shorter sequence C(B) that produces B when fed into an interpreter.



Even Better Compression

Compression ratio of 25% is certainly very impressive, but we can do much better. MysteryX achieves a 0.35% compression ratio!

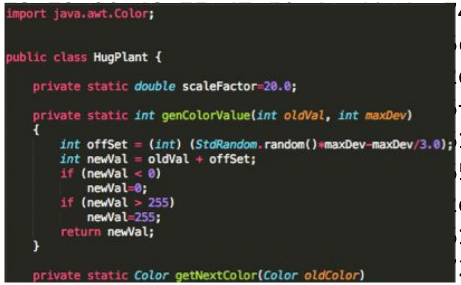
- Of the $2^{8389584}$ possible bit streams of length 8389584, only one in $2^{8360151}$ can be generated by our interpreter using an input of length 29,432 bits.



MysteryX: HugPlant.java

MysteryX is just HugPlant.java, the piece of code that I used to generate the .bmp file originally.

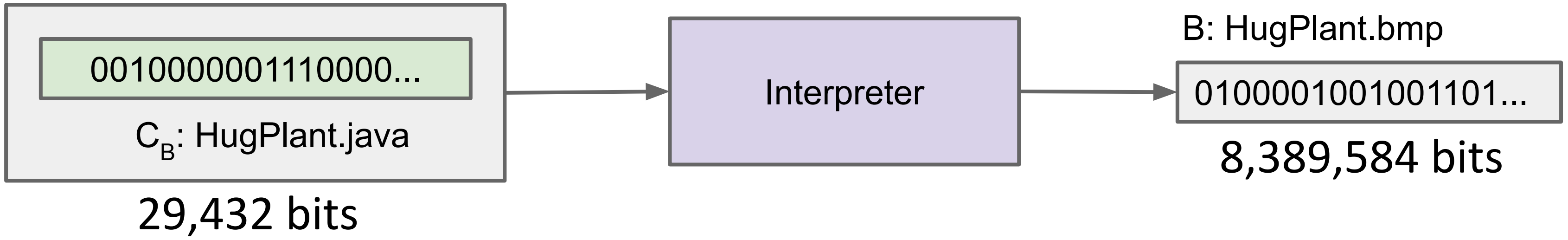
```
69 6d 70 6f 72 74 20 6a 61 76 61 2e 61 77 74 2e
43 6f 6c 6f 72 3b 0a 0a 0a 70 75 62 6c 69 63 20
63 6c 61                                  4 20 7b
0a 09 2f                                  c 20 74
6f 20 70                                  e 75 67
20 74 6f                                  f 75 72
20 70 72                                  1 74 65
20 73 74                                  5 20 73
63 61 6c                                  e 30 3b
0a 0a 09                                  1 74 69
63 20 69                                  2 56 61
6c 75 65 28 69 6e 74 20 6f 6c 64 56 61 6c 2c
```



```java
import java.awt.Color;

public class HugPlant {

    private static double scaleFactor=20.0;

    private static int genColorValue(int oldVal, int maxDev)
    {
        int offSet = (int) (StdRandom.random()*maxDev-maxDev/3.0);
        int newVal = oldVal + offSet;
        if (newVal < 0)
            newVal=0;
        if (newVal > 255)
            newVal=255;
        return newVal;
    }

    private static Color getNextColor(Color oldColor)
```

Interesting question:

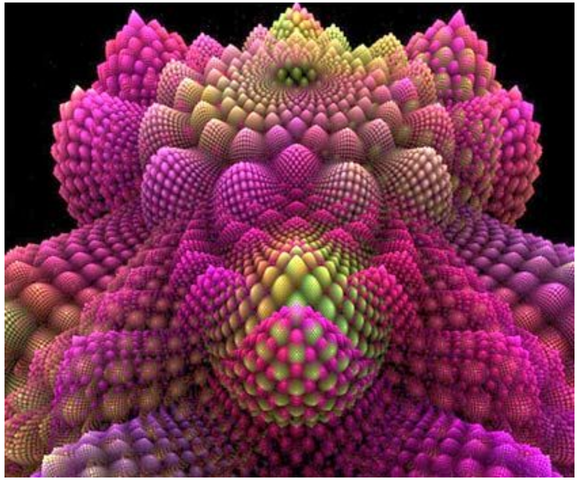
- Given a target bitstream B, what is the **shortest** bitstream C_B that outputs B?



Fractals

In the 1970s, Mandelbrot built demonstrations that very short programs could generate highly complex visual patterns.

- Biological processes exploit these same ideas. Short sequences of DNA give rise to interesting spatial (and other) patterns.



Music

These ideas are arguably more interesting (and alarming) when applied towards sound generation.

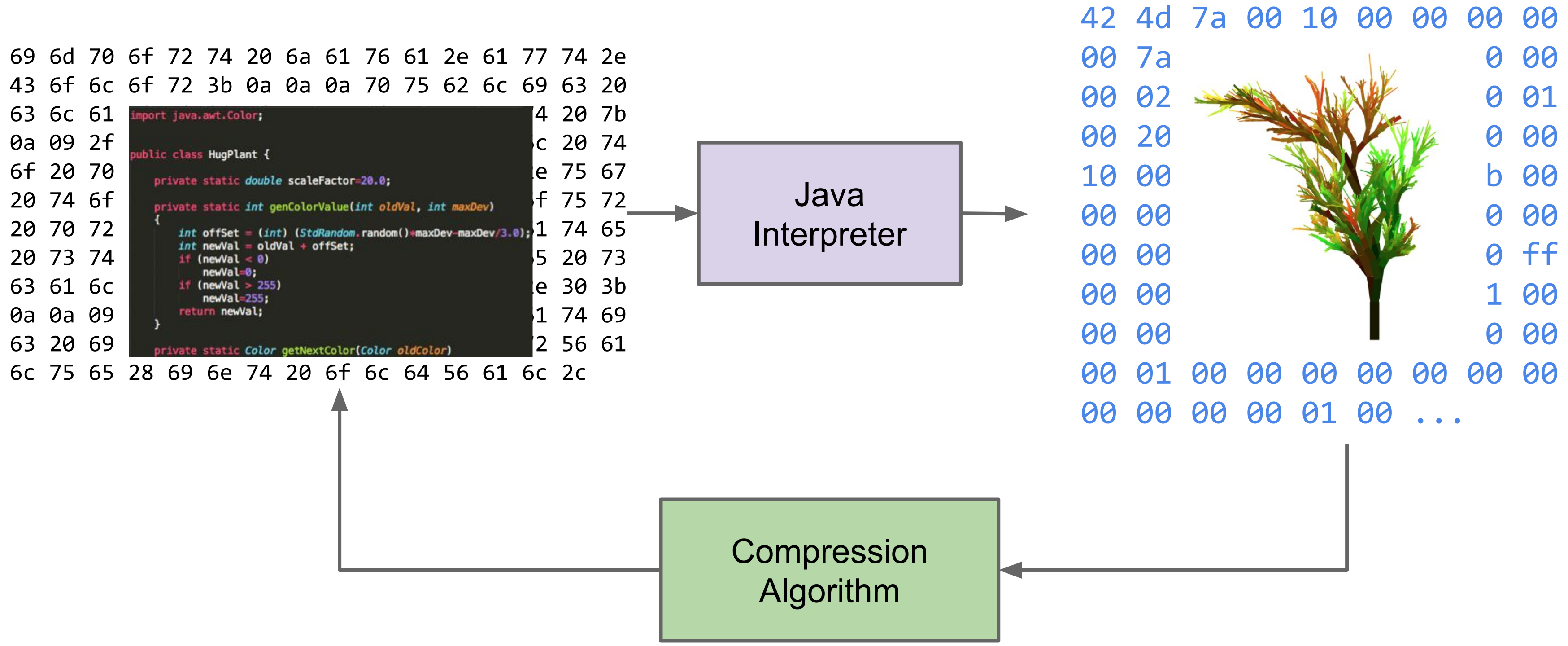
Music from very short programs (3rd iteration): Youtube

- In lab 14, you'll explore this phenomenon.

Kolmogorov Complexity (Extra - CS172 Preview)

Finding the Shortest Bit Sequence

Given a target bitstream B, what is the shortest bitstream C_B that outputs B?



```
69 6d 70 6f 72 74 20 6a 61 76 61 2e 61 77 74 2e
43 6f 6c 6f 72 3b 0a 0a 0a 70 75 62 6c 69 63 20
63 6c 61                                  4 20 7b
0a 09 2f                                  c 20 74
6f 20 70                                  e 75 67
20 74 6f                                  f 75 72
20 70 72                                  1 74 65
20 73 74                                  5 20 73
63 61 6c                                  e 30 3b
0a 0a 09                                  1 74 69
63 20 69                                  2 56 61
6c 75 65 28 69 6e 74 20 6f 6c 64 56 61 6c 2c
```

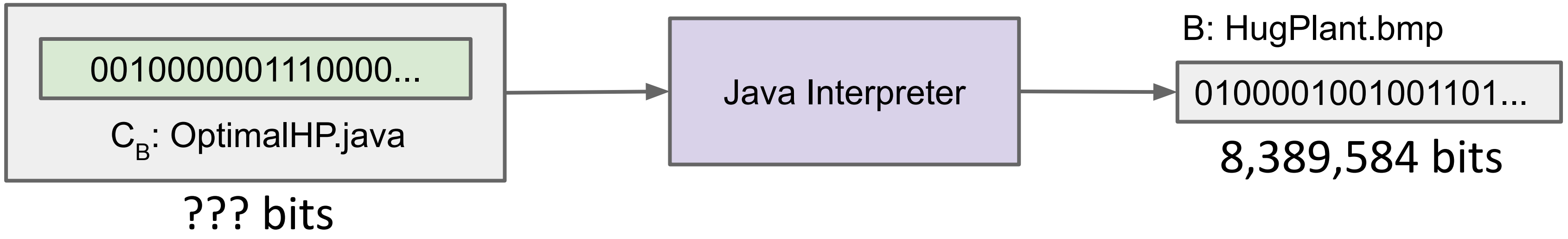
```java
import java.awt.Color;

public class HugPlant {

    private static double scaleFactor=20.0;

    private static int genColorValue(int oldVal, int maxDev)
    {
        int offSet = (int) (StdRandom.random()*maxDev-maxDev/3.0);
        int newVal = oldVal + offSet;
        if (newVal < 0)
            newVal=0;
        if (newVal > 255)
            newVal=255;
        return newVal;
    }

    private static Color getNextColor(Color oldColor)
```

```
42 4d 7a 00 10 00 00 00 00
00 7a                 0 00
00 02                 0 01
00 20                 0 00
10 00                 b 00
00 00                 0 00
00 00                 0 ff
00 00                 1 00
00 00                 0 00
00 01 00 00 00 00 00 00 00
00 00 00 00 01 00 ...
```

Kolmogorov Complexity

Given a target bitstream B, what is the shortest bitstream C_B that outputs B.

- Definition: The Java-Kolmogorov complexity $K_J(B)$ is the length of the shortest Java program (in bytes) that generates B.
  - There IS an answer. It just might be very hard to find.



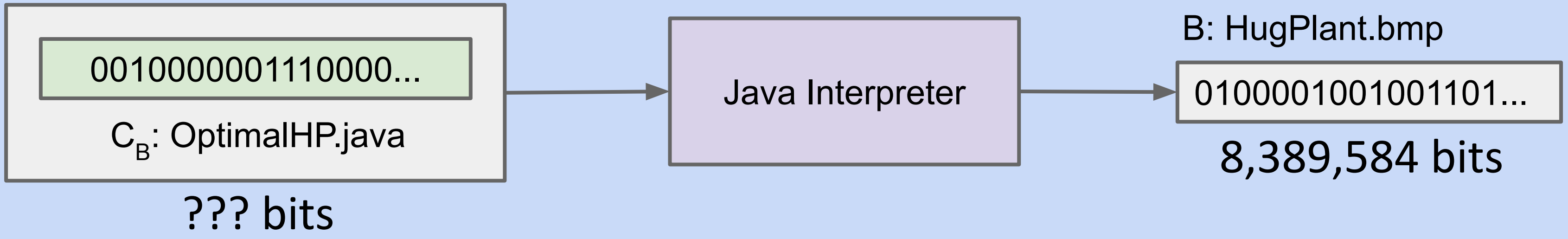
Kolmogorov Complexity

Given a target bitstream B, what is the shortest bitstream C_B that outputs B.

- Definition: The Java-Kolmogorov complexity $K_J(B)$ is the length of the shortest Java program (in bytes) that generates B.
  - There IS an answer. It just might be very hard to find.

Fact #1: Kolmogorov Complexity is effectively independent of language.

- For any bit stream, the Java-Kolmogorov Complexity is no more than a constant factor larger than the Python-Kolmogorov Complexity.
  - Why?



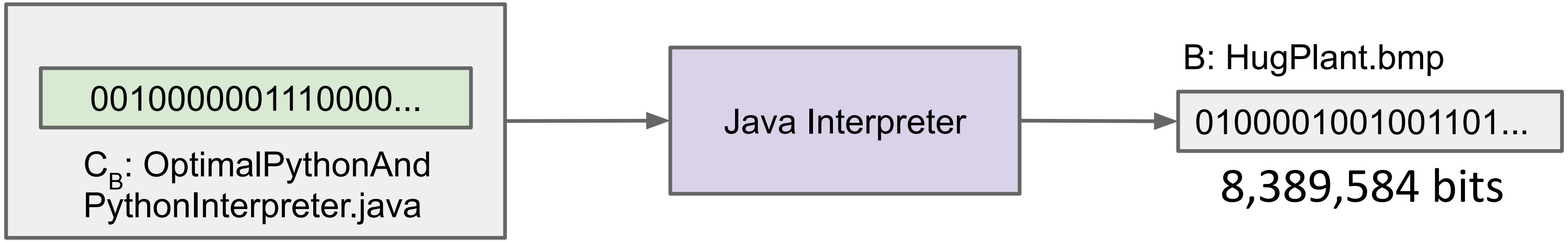
Kolmogorov Complexity (Language Independence)

Fact #1: Kolmogorov Complexity is effectively **independent** of language.

- For any bit stream, the Java-Kolmogorov Complexity is no more than a constant factor larger than the Python-Kolmogorov Complexity.
  - Why?

Kevin Lin generates a Python program that is very short and I am jealous and cannot think of a similar thing in Java.

- I could just write a Python interpreter in Java and then run Kevin's program.
  - $K_J(B) \le K_P(B)$ + size(python interpreter)



Kolmogorov Complexity (Uncomputability)

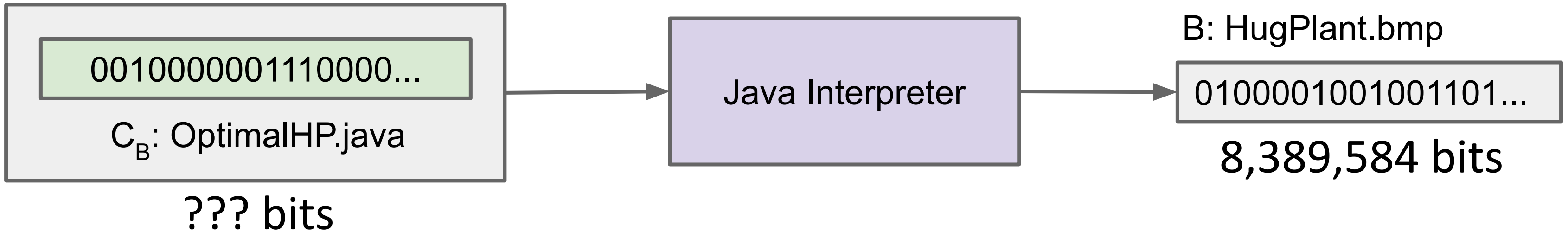
Given a target bitstream B, what is the shortest bitstream C_B that outputs B.

- Definition: The Java-Kolmogorov complexity $K_J(B)$ is the length of the shortest Java program (in bytes) that generates B.

Fact #1: Kolmogorov Complexity is effectively **independent** of language.

Fact #2: It is **impossible** to write a program that calculates the Kolmogorov Complexity of any bitstream. Proof available here.

- Corollary: It is **impossible** to write the "perfect" compression algorithm.



Problem Difficulty: How Hard is Independent Set

Problem Difficulty

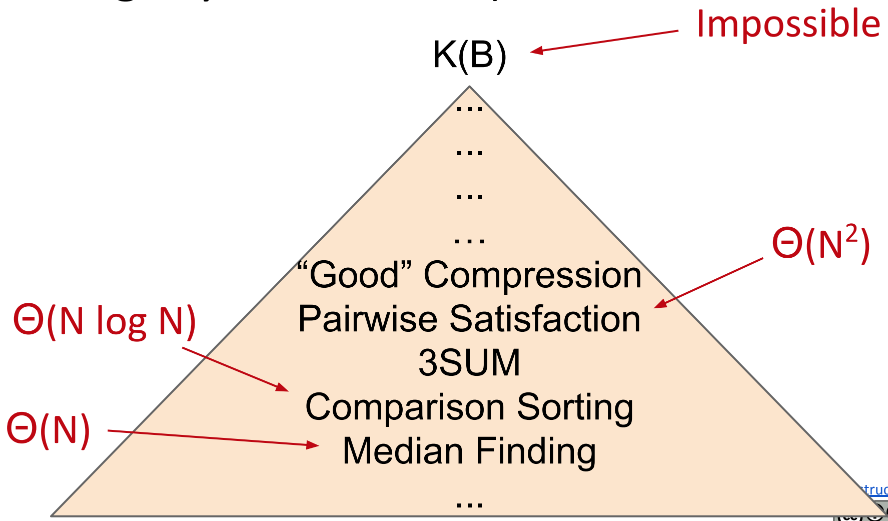
Some problems are fundamentally harder than others. What we know about the runtime of the optimal algorithms for some problems:

(optimal compression)

- Calculating the Kolmogorov complexity K(B) of a bitstream: Impossible
- Good Compression: ??? ← What we're working towards in this lecture.
- Checking if an array contains two items such that f(x1, x2) = 0: $\Theta(N^2)$
- 3SUM: $O(N^2 / (\log N / \log \log N)^{2/3})$ [just slightly less than N^2]
- Comparison based sorting: $\Theta(N \log N)$
- Finding the median of an array: $\Theta(N)$

All results above assume a simple model of computation that uses comparisons for decisions.

Note: Nobody has proven an interesting lower bound for 3SUM's difficulty, but it is widely conjectured to be slightly less than N^2.



Example: The Independent Set Problem

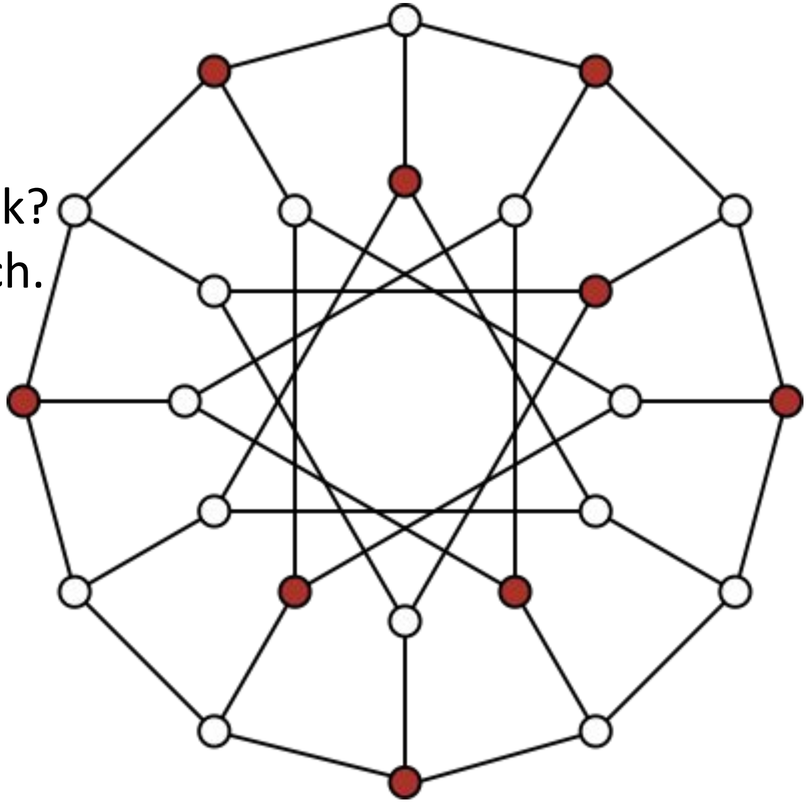
An independent set is a set of vertices in which no two vertices are adjacent.

The Independent-set Problem:

- Does there exist an independent set of size k?
- i.e. color k vertices red, such that none touch.

Example for the graph on the right and k = 9

- For this particular graph, N=24.



The Independent Set Problem

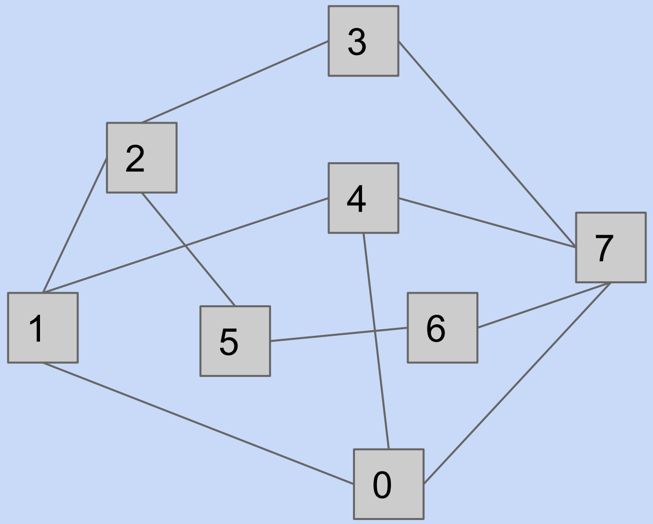
An independent set is a set of vertices in which no two vertices are adjacent.

The Independent-set Problem:

- Does there exist an independent set of size k?
- i.e. color k vertices red, such that none touch.

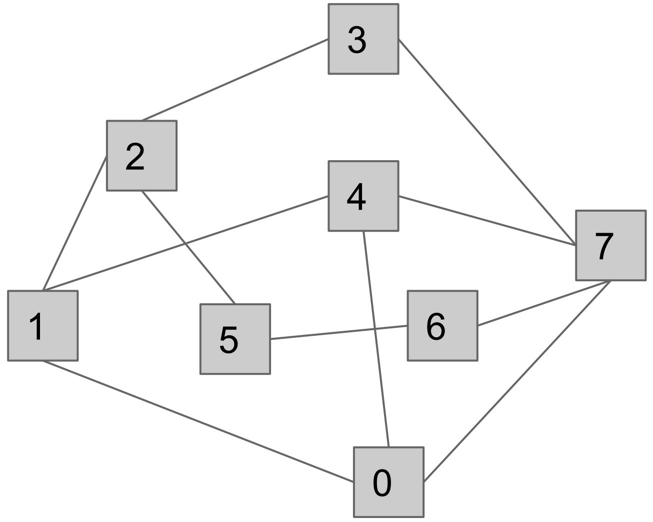
Give an algorithm for solving this problem.

- Must work for any graph and any k.
- Don't worry about runtime.



The Independent Set Problem

An independent set is a set of vertices in which no two vertices are adjacent.



The Independent-set Problem:

- Does there exist an independent set of size k?
- i.e. color k vertices red, such that none touch.

Give an algorithm for solving this problem.

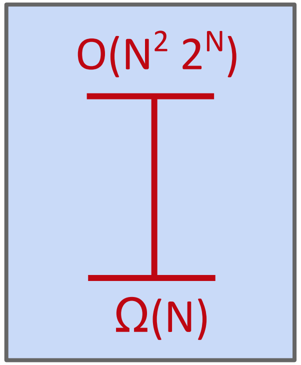
- For each of the possible 2^N colorings:
  - Check if number of colored vertices is equal to k: O(N)
  - For every red vertex, check that neighbors are all white: O(k*N)
  - If both checks succeed, return true.
  - If either check fails, go on to next coloring.
- Runtime: $O(k*N*2^N)$. Since $k \leq N$, **$O(N^2*2^N)$**

Independent Set Difficulty

Let algorithm TUISA be the asymptotically optimal algorithm for the Independent Set problem.

What can we say about the worst case runtime of TUISA?

- At least N (have to do something with at least every vertex), so $\Omega(N)$.
- No greater than our existing algorithm, so $O(N^2 2^N)$.



Can we say more? Kinda. We'll need to take a couple of detours into:

- Reductions
- Complexity Classes

At the end, we'll tie this all back to compression.

Reductions

Reductions And “Cracking”

We informally say that problem X ***reduces to*** problem Y if an instance of X can be transformed into a useful instance of problem Y that solves X.

Example: X = “median finding ignoring all zeros”, Y = “sorting”.

- Suppose we want to find the median ignoring zeros of `[99.5, 87.3, 92.5, 0, 0, 62.5, 89.2]`. We create a new array `[99.5, 87.3, 92.5, 62.5, 92.2]`, sort, and return the middle item.

For clarity in this lecture, I will define my own term: We informally say that problem Y ***cracks*** problem X if X reduces to Y.

- Example: “sorting” cracks “median finding ignoring all zeros”.
- Example 2: “sorting” cracks “puppy-cat-dog”.

The 3SAT Problem

3SAT: Given a boolean formula, does there exist a truth value for boolean variables that obeys a set of 3-variable disjunctive constraints?

Example: $\overbrace{\text{(x1 || x2 || !x3)}}^{\text{3 variable disjunctive constraint}}$ && (x1 || !x1 || x1) && (x2 || x3 || x4)

- Solution: x1 = true, x2 = true, x3 = true, x4 = false

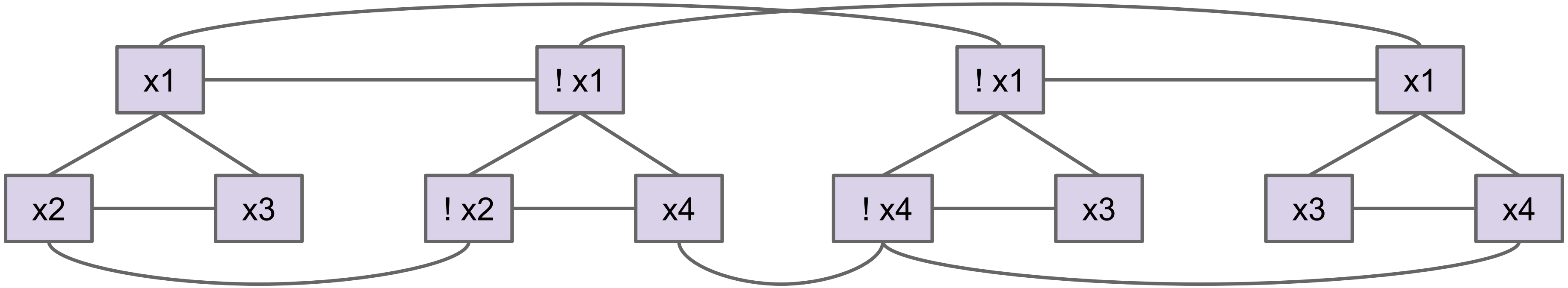
Independent Set Cracks 3SAT

Proposition: Independent-set cracks 3SAT.

Proof. Given an instance ϕ of 3-SAT, create an instance G of Independent-set:

- For each clause in ϕ, create 3 vertices in a triangle.
- Add an edge between each literal and its negation (can't both be true in 3SAT means can't be in same set in Independent-set)

k = number of variables = 4



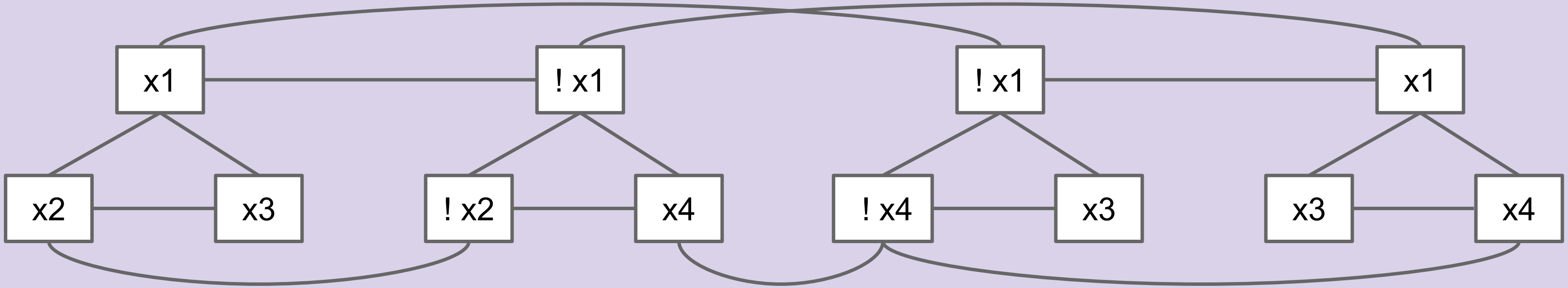
Φ = (x1 or x2 or x3) and (! x1 or ! x2 or x4) and (! x1 or x3 or ! x4) and (x1 or x3 or x4)

Independent Set Cracks 3SAT

Find an independent set of size k = 4. Use this set to generate a solution to the 3SAT problem.

- Reminder: An independent set of size 4 is a set of 4 (red) vertices that do not touch.

k = number of variables = 4



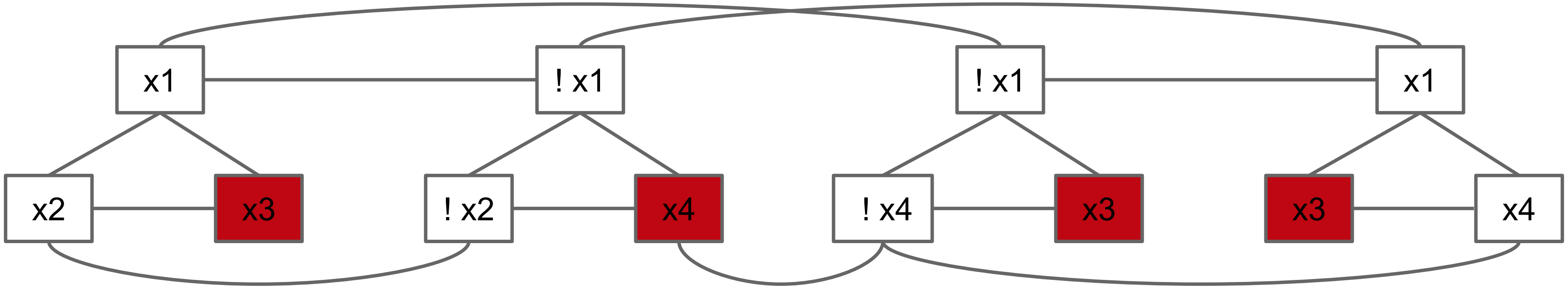
Φ = (x1 or x2 or x3) and (! x1 or ! x2 or x4) and (! x1 or x3 or ! x4) and (x1 or x3 or x4)

Independent Set Cracks 3SAT

Find an independent set of size k = 4. Use this set to generate a solution to the 3SAT problem.

- Reminder: An independent set of size 4 is a set of 4 (red) vertices that do not touch.

k = number of variables = 4



Φ = (x1 or x2 or **x3**) and (! x1 or ! x2 or **x4**) and (! x1 or **x3** or ! x4) and (x1 or x3 or **x4**)

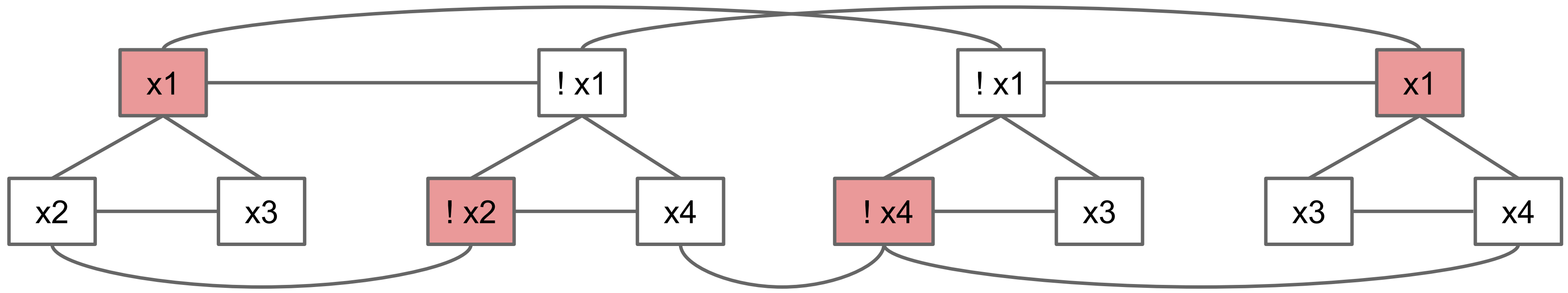
x1: don’t care, x2: don’t care, x3: true, x4: true

Independent Set Cracks 3SAT

Find an independent set of size k = 4. Use this set to generate a solution to the 3SAT problem.

- Reminder: An independent set of size 4 is a set of 4 (red) vertices that do not touch.

k = number of variables = 4



Φ = (x1 or x2 or x3) and (! x1 or ! x2 or x4) and (! x1 or x3 or ! x4) and (x1 or x3 or x4)

x1: TRUE, x2: FALSE, x4: FALSE, x3: Don't care

Independent Set Difficulty

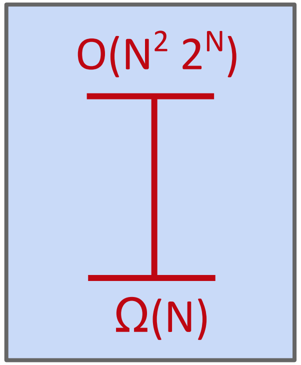
Let algorithm TUISA be the asymptotically optimal algorithm for the Independent Set problem.

What can we say about the worst case runtime of TUISA?

- At least N (have to do something with at least every vertex), so $\Omega(N)$.
- No greater than our algorithm, so $O(N^2 2^N)$.

Can we say more? Kinda. We now know that TUISA cracks 3SAT, so if we had some sort of lower bound on 3SAT, we could apply it to TUISA.

- We used this same idea to put a lower bound on comparison sorting by showing that it cracked puppy-cat-dog.
- Unfortunately, there is no known useful lower bound on 3SAT.
- Let's take yet another detour into complexity classes.



Complexity Classes (Extra)

The Class P

A ***decision problem*** is a problem with a yes or no answer.

- Example: Does there exist a pair of duplicates in an array?
- Not a decision problem: How many duplicates are there in an array?

Minor technical point: N is the number of bits needed to specify the input.

We say that a problem is ***in the complexity class P*** if:

- It is a decision problem.
- An answer can be found in $O(N^k)$ time for some k.

Example: Are there two items in an array whose sum is zero? Can solve using technique from discussion (sort, then use two pointers). Runtime is $O(N^2)$.

The Class NP

A ***decision problem*** is a problem with a yes or no answer.

- Example: Does there exist an independent set of size k for graph G?
- Not a decision problem: How big is the biggest independent set for graph G?

Clyde Kruskal has suggested that "VP" is a better name, for "Verifiable in Polynomial Time"

We say that a problem is ***in the complexity class NP*** if:

- It is a decision problem.
- A "yes" answer can be verified in $O(N^k)$ time for some k. More precisely, we can verify a specific example of a "yes" answer in $O(N^k)$ time.

Example: Is there an independent set of size k? Yes, e.g. some set of red vertices Q. To verify, check that all vertices adjacent to vertices in Q are white. Runtime is $O(QN)$, which is $O(N^2)$.

Why Does The Complexity Class P Matter?

Problems in the complexity class P are generally regarded as "easy". For typical N, k, can complete execution within a human lifetime. Example k values:

- Comparison Sorting: 2
- BreadthFirstPaths: 1

Nice features of P:

- $O(N^k)$ is closed under addition and multiplication.
  - Run two P algorithms, overall still in P.
  - Run a P algorithm in N times, still in P.

Exponents for practical problems are typically small.

Why Does The Complexity Classes NP Matter?

Many (most?) practical problems can be cast as a problem in NP:

- Is there a way to route my airplanes at a total cost of less than $1B/yr?
- Is there a way to route the wires inside this microchip with a total path length of less than 1 micrometer?
- Given Z, are there two primes such that X*Y = Z?
- Is there a protein configuration for amino acid sequence X whose total energy is less than Y?

Aside: can generalize idea to problems for which a "no" answer is verifiable.

Independent Set Difficulty

Let algorithm TUISA be the asymptotically optimal algorithm for the Independent Set problem.

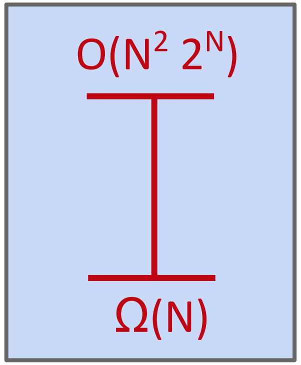
What can we say about the worst case runtime of TUISA?

- At least N (have to do something with at least every vertex), so $\Omega(N)$.
- No greater than our algorithm, so $O(N^2 2^N)$.

Can we say more? Kinda. Here's what we can say so far:

- The Independent Set problem is in NP.
  - Why? It's a yes/no question "Is there an independent set of size k?" for which a yes answer can be quickly verified.

But is Independent Set in P? Great question!



NP Complete Problems

We'll define a problem π as ***NP-Complete*** if:

- π is a member of NP.
- π *cracks* every other problem in NP.

This raises two questions:

- Are there any NP-Complete problems?
- Do we know how to solve any of them efficiently?

The Cook-Levin Theorem

In 1971, Stephen Cook showed that the 3SAT problem is NP Complete.

- 3SAT is a member of NP.
- 3SAT *cracks* every other problem in NP.

Punchline: If we could efficiently solve 3SAT, we could solve ANY yes/no question whose answer we can efficiently verify.

3SAT: Does there exist a truth value for boolean variables that obeys a set of 3-variable disjunctive constraints: (x1 || x2 || !x3) && (x1 || !x1 || x1)

Levin later (1973) showed a similar result. See Cook-Levin Theorem for more.

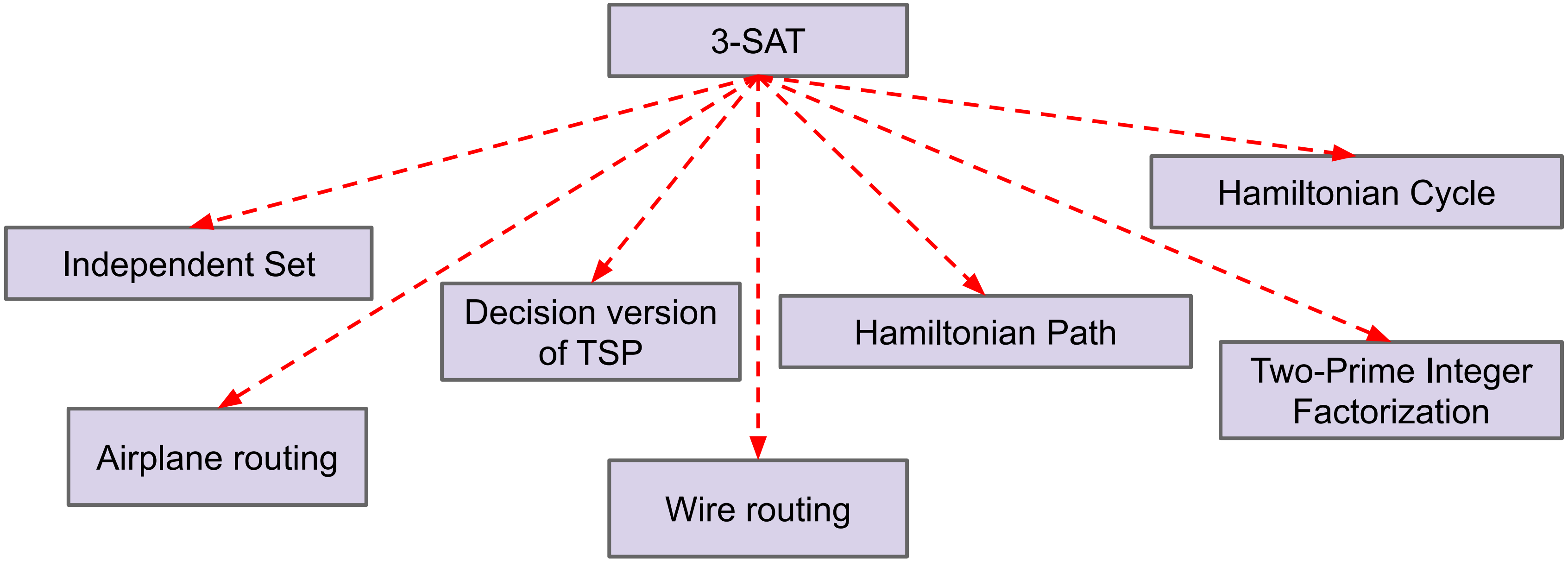
Cook-Levin Theorem

Very rough idea of Cook's proof:

- Create giant (!!) boolean logic expression that represents entire state of your computer at every time step.
- If solution takes polynomial time, boolean logic circuit is polynomial in size.
- Example boolean logic variable: True if 57173rd bit of memory is true and we're on line 38 of code during cycle 7591872 of execution.

Cook-Levin Theorem Visually

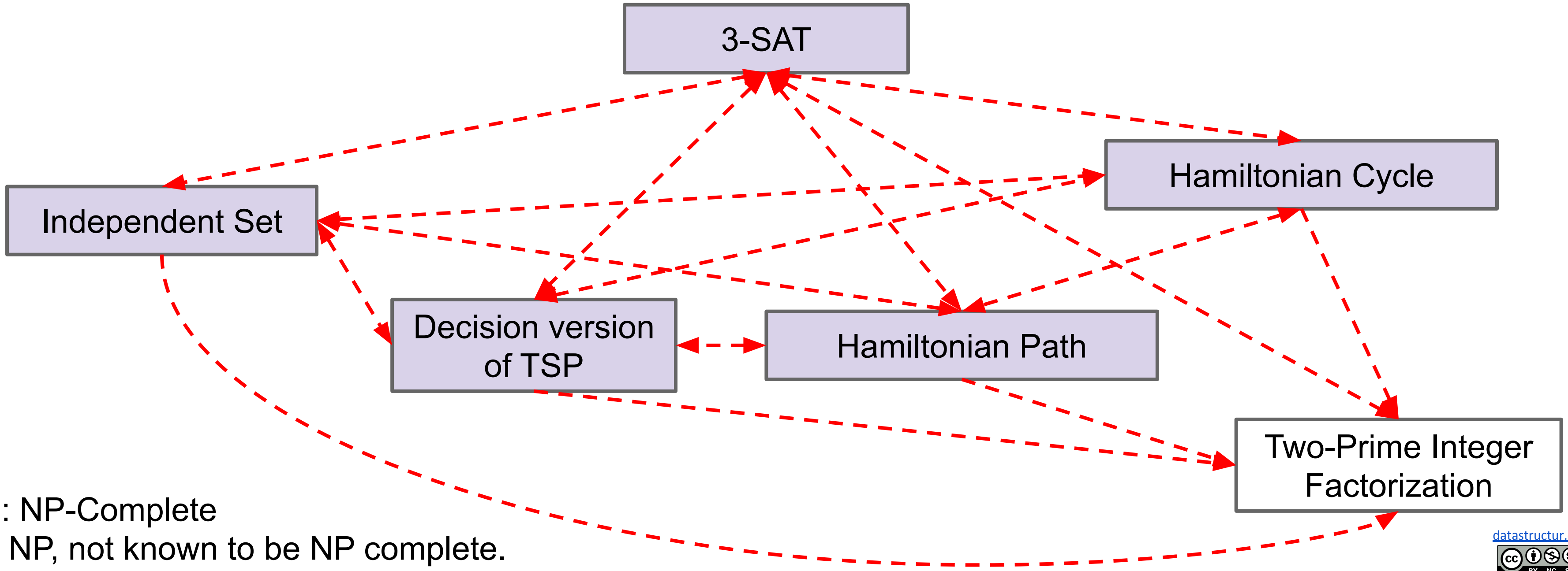
3-SAT *cracks* any problem for which a yes answer can be efficiently verified.



Karp's 21 NP-complete problems

In 1972, Dick Karp showed that 21 well-known NP problems crack 3SAT.

- Implication: These well known NP problems are also NP-Complete.
  - Example: Hamiltonian Cycle cracks 3SAT, therefore it also cracks every other NP problem!



Purple: NP-Complete
White: NP, not known to be NP complete.

Everyone Else’s NP Complete Problems

Since 1972, thousands of additional problems have been proven NP-Complete, including:

- Finding the longest paths in a graph.
- Finding the minimal number of edges that we can add to a graph so that it contains a Hamiltonian cycle.
- Determining whether a Super Mario Bros. level has a solution.

To prove that X is NP-Complete, simply have to crack ANY NP-Complete problem with X.

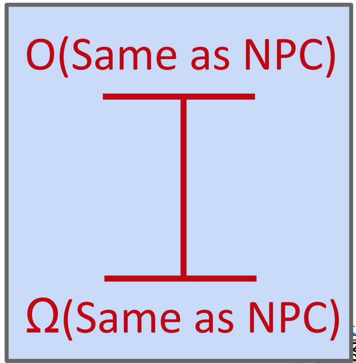
As number of NP-Complete problems grow, it gets easier to find new NP-Complete problems. An efficient solution to ANY of these problems would solve ALL of them.

Independent Set Difficulty

Let algorithm TUISA be the asymptotically optimal algorithm for the Independent Set problem.

What can we say about the worst case runtime of TUISA?

- At least N (have to do something with at least every vertex), so Ω(N).
- No greater than our algorithm, so $O(N^2\, 2^N)$.
- Same runtime as thousands of other interesting NP-Complete problems.
  - If someone could prove that any NP-Complete problem takes at least exponential time, would also apply to X.
  - Likewise:
    - Proof about X would apply to all NPC problems.
    - Solution for X would solve all NPC complete problems.

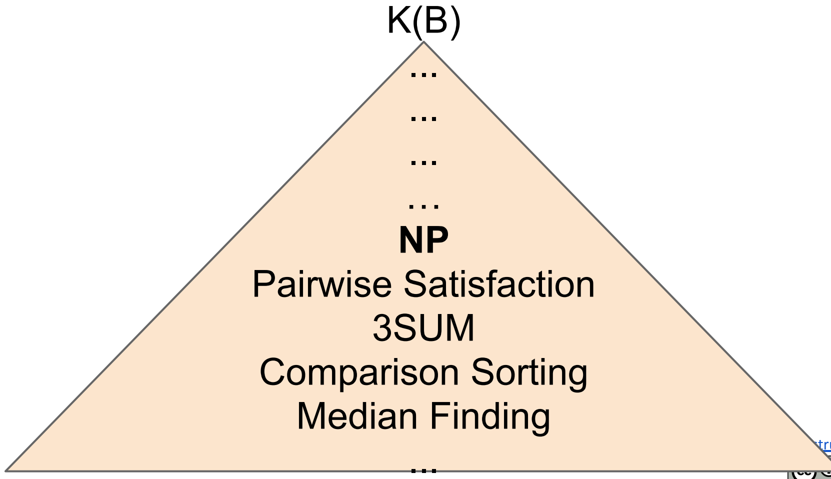


Problem Difficulty

Some problems are fundamentally harder than others. What we know about the runtime of the optimal algorithms for some problems:

(optimal compression)

- Calculating the Kolmogorov complexity K(B) of a bitstream: Impossible
- **NP Problems (including Good Compression): Unknown Difficulty**
- Checking if an array contains two items such that f(x1, x2) = 0: $\Theta(N^2)$
- 3SUM: $O(N^2 / (\log N / \log\log N)^{2/3})$ [just slightly less than N^2)
- Comparison based sorting: $\Theta(N \log N)$
- Finding the median of an array: $\Theta(N)$



Nobody knows how hard NP is, but it is conjectured to be quite hard!

How is compression related to NP? Stay tuned.

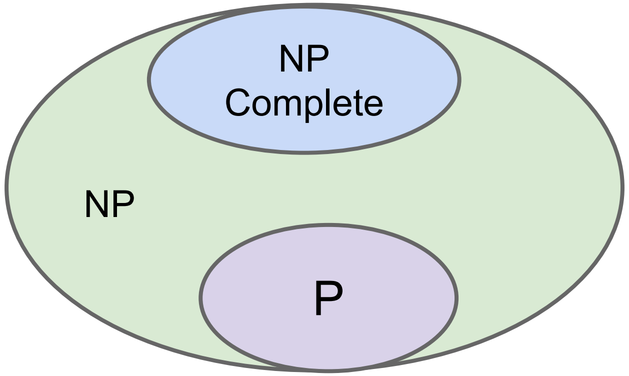
P=NP?

So far:

- P is the set of decision problems that can be <u>solved</u> in $O(N^k)$ time
- NP is the set for which a yes answer can be <u>verified</u> in $O(N^k)$ time.
- NP-complete problems are NP problems that *<u>crack</u>* all other NP problems.

It is obvious that P and NP-complete are both subsets of NP.

- P is a subset of NP because if you can solve something you can trivially verify a solution.



P = NP? (Extra)

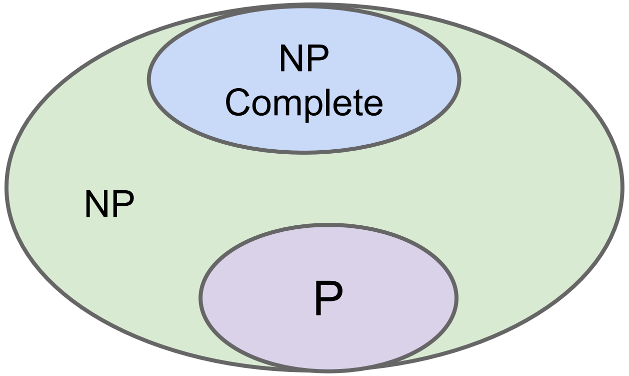
P = NP?

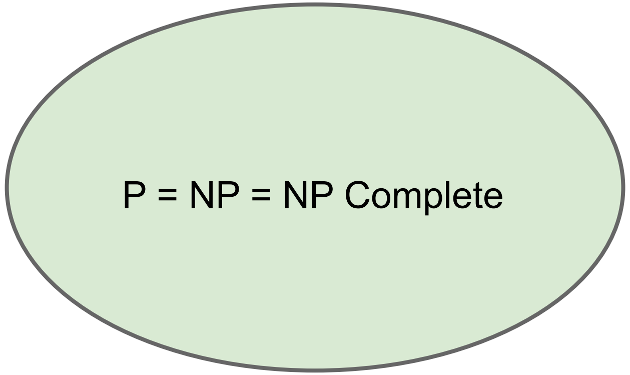
So far:

- P is the set of decision problems that can be <u>solved</u> in $O(N^k)$ time
- NP is the set for which a yes answer can be <u>verified</u> in $O(N^k)$ time.
- NP-complete problems are NP problems that *<u>crack</u>* all other NP problems.

In 1971, Cook posed the question: Are all of the problems in NP also in P?

- Are efficiently verifiable problems also efficiently solvable, i.e. is P = NP?





P = NP?

Consensus Opinion (Bill Gasarch Poll, 2012 poll)

- 83%: P ≠ NP (126 respondents)
- 9%: P = NP (12 respondents)
- 9%: Other (13 respondents)

Why is opinion generally negative?

- Someone would have proved it by now.
  - "The only supporting arguments I can offer are the failure of all efforts to place specific NP-complete problems in P by constructing polynomial-time algorithms." - Dick Karp
- Creation of solutions seems philosophically more difficult than verification.

Mathematical Proofs Are in NP!

Is there a proof that the Riemann Hypothesis is true?

- If P=NP, then we can automate mathematical proof! First observed informally by Kurt Gödel himself.

"[A linear or quadratic-time procedure for what we now call NP complete problems would have] consequences of the greatest magnitude. [For such a procedure] would clearly indicate that, despite the unsolvability of the Entscheidungsproblem, the mental effort of the mathematician in the case of yes-or-no questions could be completely replaced by machines." - *Kurt Gödel*

One of These Things, Is Not Like The Others

In 2000, the Clay Mathematics Institute set up $1,000,000 prizes for the solution of each of seven problems.

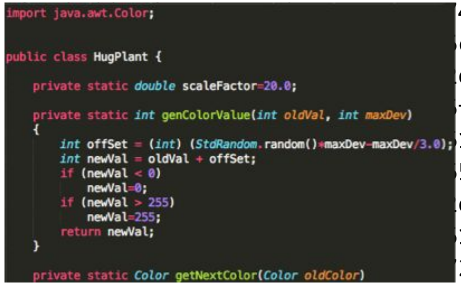
Millenium Prize Problems.

- Hodge conjecture
- Poincare conjecture (solved!)
- Riemann hypothesis
- Yang-Mills existence and mass gap
- Navier-Stokes existence and smoothness
- Birch and Swinnerton-dyer conjecture
- P=NP
    - If true, proof might allow you to trivially solve all of these problems.

Even More Impressive Consequences

Recall the humble HugPlant.

```
69 6d 70 6f 72 74 20 6a 61 76 61 2e 61 77 74 2e
43 6f 6c 6f 72 3b 0a 0a 0a 70 75 62 6c 69 63 20
63 6c 61                               4 20 7b
0a 09 2f                               c 20 74
6f 20 70                               e 75 67
20 74 6f                               f 75 72
20 70 72                               1 74 65
20 73 74                               5 20 73
63 61 6c                               e 30 3b
0a 0a 09                               1 74 69
63 20 69                               2 56 61
6c 75 65 28 69 6e 74 20 6f 6c 64 56 61 6c 2c
```



```
import java.awt.Color;

public class HugPlant {

    private static double scaleFactor=20.0;

    private static int genColorValue(int oldVal, int maxDev)
    {
        int offSet = (int) (StdRandom.random()*maxDev-maxDev/3.0);
        int newVal = oldVal + offSet;
        if (newVal < 0)
            newVal=0;
        if (newVal > 255)
            newVal=255;
        return newVal;
    }

    private static Color getNextColor(Color oldColor)
```

$\rightarrow$ Interpreter $\rightarrow$

```
42 4d 7a 00 10 00 00 00 00
00 7a                 0 00
00 02                 0 01
00 20                 0 00
10 00                 b 00
00 00                 0 00
00 00                 0 ff
00 00                 1 00
00 00                 0 00
00 01 00 00 00 00 00 00 00
00 00 00 00 01 00 ...
```

0010000001110000...

C_B: HugPlant.java

29,432 bits

$\rightarrow$ Interpreter $\rightarrow$

B: HugPlant.bmp

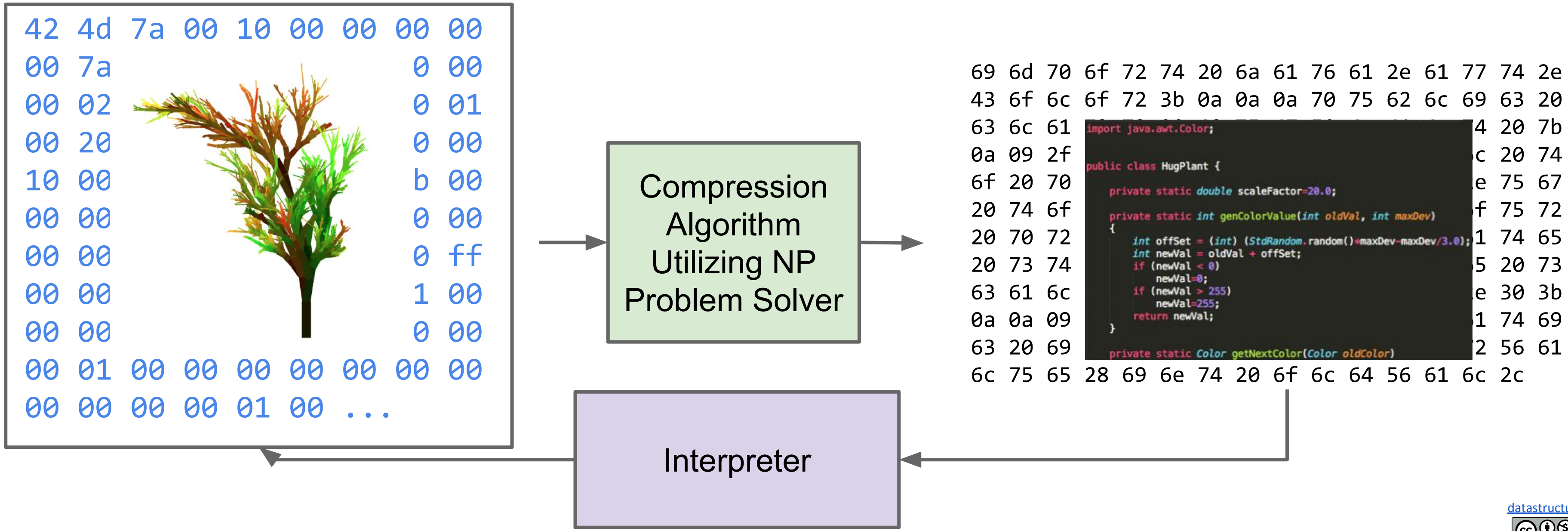
0100001001001101...

8,389,584 bits

Even More Impressive Consequences

Recall the humble HugPlant.

- Could we run this process in reverse?
- Example NP Problem: “Is there a program of less than 50,000 bits that generates HugPlant.bmp in less than 100,000,000 cycles?”



```
42 4d 7a 00 10 00 00 00 00
00 7a                 0 00
00 02                 0 01
00 20                 0 00
10 00                 b 00
00 00                 0 00
00 00                 0 ff
00 00                 1 00
00 00                 0 00
00 01 00 00 00 00 00 00 00
00 00 00 00 01 00 ...
```

```
69 6d 70 6f 72 74 20 6a 61 76 61 2e 61 77 74 2e
43 6f 6c 6f 72 3b 0a 0a 0a 70 75 62 6c 69 63 20
63 6c 61                                  4 20 7b
0a 09 2f                                  c 20 74
6f 20 70                                  e 75 67
20 74 6f                                  f 75 72
20 70 72                                  1 74 65
20 73 74                                  5 20 73
63 61 6c                                  e 30 3b
0a 0a 09                                  1 74 69
63 20 69                                  2 56 61
6c 75 65 28 69 6e 74 20 6f 6c 64 56 61 6c 2c
```

```java
import java.awt.Color;

public class HugPlant {

    private static double scaleFactor=20.0;

    private static int genColorValue(int oldVal, int maxDev)
    {
        int offSet = (int) (StdRandom.random()*maxDev-maxDev/3.0);
        int newVal = oldVal + offSet;
        if (newVal < 0)
            newVal=0;
        if (newVal > 255)
            newVal=255;
        return newVal;
    }

    private static Color getNextColor(Color oldColor)
```

Even More Impressive Consequences

"I have heard it said, with a straight face, that a proof of P = NP would be important because it would let airlines schedule their flights better, or shipping companies pack more boxes in their trucks!

If [P = NP], then we could quickly find the smallest Boolean circuits that output (say) a table of historical stock market data, or the human genome, or the complete works of Shakespeare. It seems entirely conceivable that, by analyzing these circuits, we could make an easy fortune on Wall Street, or retrace evolution, or even generate Shakespeare's 38th play. For broadly speaking, that which we can compress we can understand, and that which we can understand we can predict.

So if we could solve the general case—if knowing something was tantamount to knowing the shortest efficient description of it—then we would be almost like gods. [Assuming P ≠ NP] is the belief that such power will be forever beyond our reach."

- Scott Aaronson *http://www.scottaaronson.com/papers/npcomplete.pdf*

More NP Complete Problems (Extra)

3-Colorability

Goal: Determine whether a graph is bipartite (every vertex may be be colored black or white, with no vertices touching any vertex of same color).

- Solution: See discussion 13.

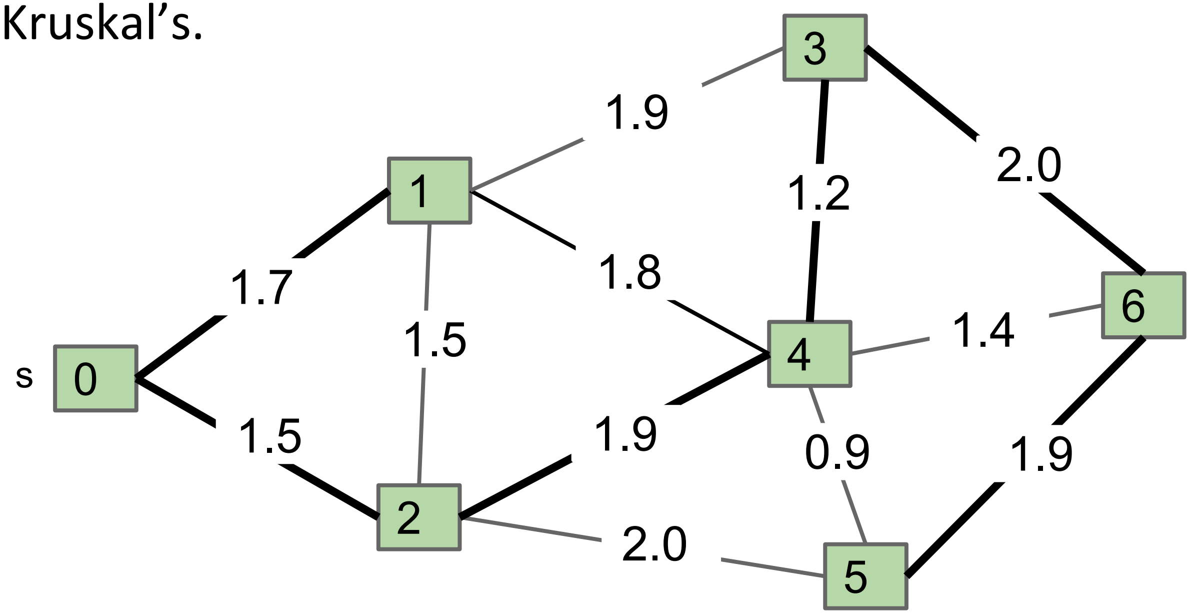
Goal: Determine whether a graph is tripartite (every vertex may be colored red, green, or blue, with no vertices touching any vertex of same color).

- Much more difficult problem that nobody knows how to efficiently solve.

Euclidean Minimum Spanning Trees

Goal: Find a connected acyclic subgraph of minimum weight.

- The answer is a tree (which we call the minimum spanning tree).
- Algorithms: Prim's and Kruskal's.



In the Euclidean Minimum Spanning Trees problem, edge weights are based on position of vertices in space.

Euclidean Steiner Trees

Goal: Given vertices in space, connect them with a graph of minimum total length.

- If connections are direct, the problem is trivial (just the MST problem).
- If we allow substations, problem is much more difficult.