Java programming for fun

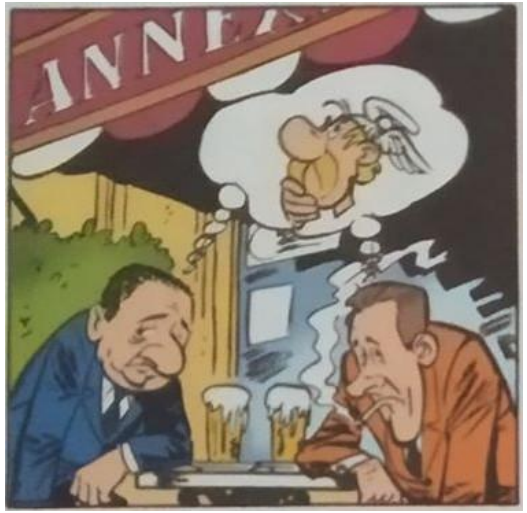
vJHUG June 2020

Anastasopoulos Spyros
@anastasop

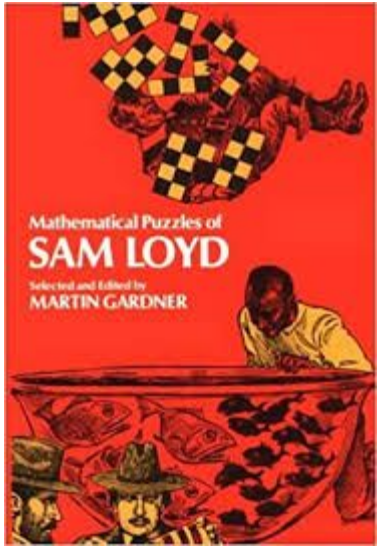
Build something for **fun** ~~and profit~~

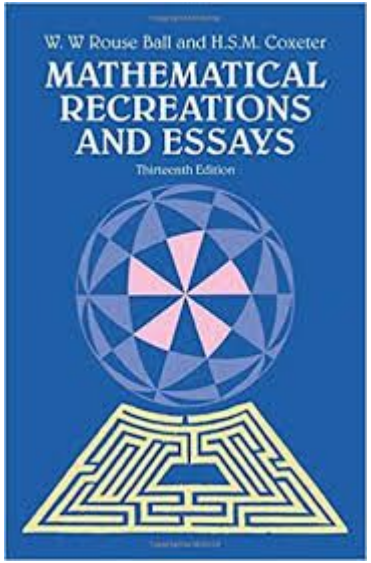
- Learn the new java features
- Have fun
- Explore

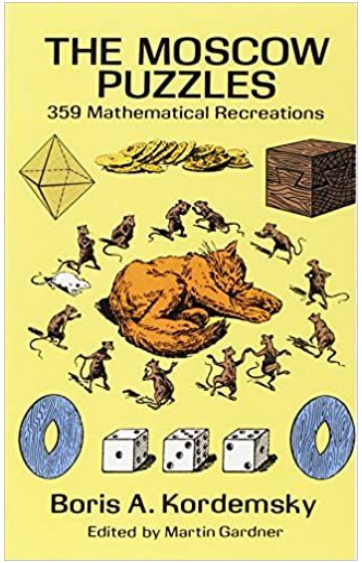
All i need is an idea

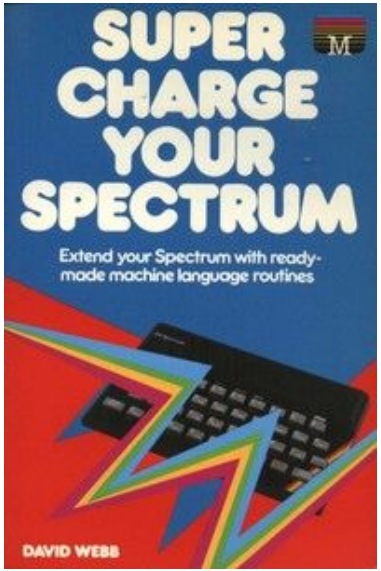


Riddles, puzzles, quizzes, games









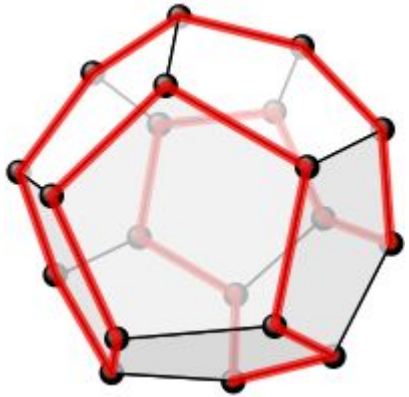
Sam Loyd - The canals on Mars

Here is a map of the newly discovered cities and waterways on our nearest neighbor planet, Mars. Start at the city market **T** at the south pole, and see if you can spell a complete English sentence by making a tour of all the cities, visiting each city only once, and returning to the starting point.

When this puzzle originally appeared in a magazine, more than fifty thousand readers reported, “There is no possible way”. Yet it is a very simple puzzle.

Sam Loyd - The canals on Mars

- Graph theory
  - Graph algorithms
  - Eulerian trail - Seven Bridges of Königsberg
  - Hamiltonian circle - Icosian game
    - NP-complete
    - https://stackoverflow.com/questions/13107545/how-to-find-hamiltonian-cycle-in-a-graph



Sam Loyd - The canals on Mars - Brute force

```
var mars = """
  00 - 00 00
  01 T 02 11
  02 H 04 03 13 17
  03 N 04 05 06 07 14
  04 O 02 03 05
  05 P 03 04 06
  06 O 05 03 07
  07 S 06 03 08 09
  08 B 07 14 09 10 11 16
  09 S 08 07 10
  10 I 11 08 09
  11 Y 08 10 19 12 01
  12 A 11 19
  13 I 02 14 20 15 17 18
  14 S 03 13 08
  15 E 13 18 19
  16 L 08 20 19
  17 E 13 02 18
  18 R 17 15 13
  19 W 12 15 20 16 11
  20 E 19 16 13
""".split("\\n");
```

```
station.setLetter(tokens.get(1));
station.setNextStations(tokens.subList(2, tokens.size()).stream()
    .map((num) -> stations.get(Integer.valueOf(num, 10)))
    .collect(Collectors.toList()));
```

```
public static void generate(Station station, List<Station> trail) {
    trail.add(station);
    station.visited = true;

    if (trail.size() == 20 && station.letter.equals("Y")) {
        var phrase = trail.stream().map((s) -> s.letter).collect(Collectors.joining());
        System.out.println(phrase);
    }

    for (var st : station.nextStations) {
        if (!st.visited) {
            generate(st, trail);
        }
    }

    station.visited = false;
    trail.remove(trail.size() - 1);
}
```

Sam Loyd - The canals on Mars

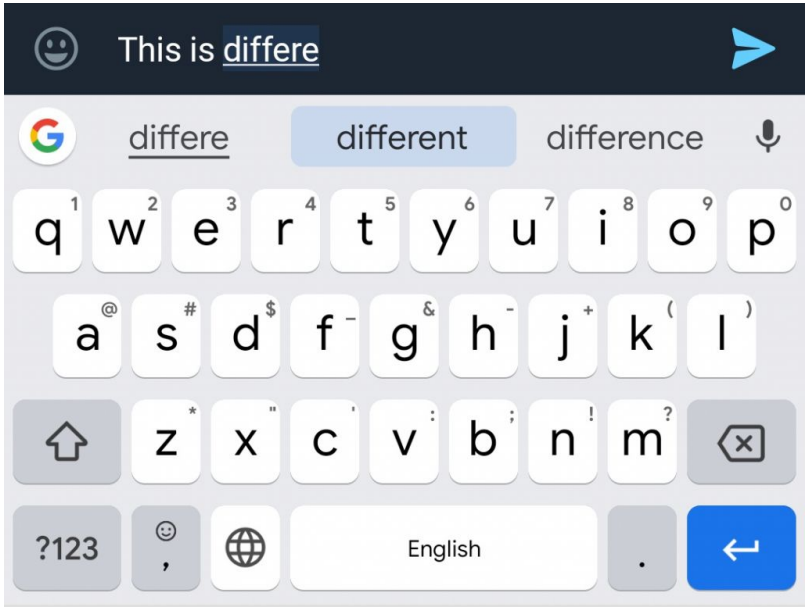
Here is a map of the newly discovered cities and waterways on our nearest neighbor planet, Mars. Start at the city market **T** at the south pole, and see if you can spell a complete English sentence by making a tour of all the cities, visiting each city only once, and returning to the starting point.

When this puzzle originally appeared in a magazine, more than fifty thousand readers reported, **“There is no possible way”**. Yet it is a very simple puzzle.

THEREISNOPOSSIBLEWAY

How to split a string of letters into words?

- Predictive text
- Gboard



Use a dictionary

- Collect letters, keep prefixes and if you find a word in the dictionary, emit it.
- Very easy to break
- Can’t tell when should go for longest word and when for shortest

there is no possible way ⇒ there isn o possible way (break because of word **isn’t**)

`hit space to start` ⇒ `HITS PACE TO START`

`everything takes longer than you think` ⇒
`EVERYTHING TAKES LONGER THAN YOUTH INK`

Use a model for spoken text

- Markov Chains
  - View text as overlapped tuples: (view text) (text as) (as overlapped) (overlapped tuples)
  - Create the mapping (word1, word2) ⇒ [next1, next2, …, nextN]
- Used a lot to generate text based on provided text
  - Choose a word and repeat (word1, word2) ⇒ (word2, nextR)
  - https://towardsdatascience.com/simulating-text-with-markov-chains-in-python-1a27e6d13fc6
  - http://ironprison.blogspot.com/2010/05/kke-generator-v10.html (not exact markov)
- Special mention: The Practice of Programming book

Project Gutenberg provides free books in plain text format. I used the Sherlock Holmes stories.

Interlude: parse a large text into words

```
cat in.txt | tr -cs [A-Za-z] $'\n' | sed '/^$/d' | tr A-Z a-z > out.txt
```

```java
public Stream<String> stream() {
  // ...
  private String readWord() throws IOException {
    var sb = new StringBuilder();

    for (var c = reader.read(); c != -1; c = reader.read()) {
      if (Character.isLetter(c)) {
        sb.append((char)Character.toLowerCase(c));
      } else if (sb.length() > 0) {
        break;
      }
    }

    return sb.length() > 0 ? sb.toString() : null;
  }
}
```

- Java is slower that the pipeline: 0.073s vs 1.858s
- Java handles unicode & accents: fiancé vs fianc e
- What to do with punctuation?

The data structures

```
public class Vocabulary {
    private final Map<String, Integer> wordCounts;
    private final Map<String, Integer> prefixCounts;
}

public class RephraserMarkovChain implements Rephraser {
    record WordTuple(String first, String second) {}

    private final Map<WordTuple, Vocabulary> chain;
}
```

The algorithm

```
var first = "";
var second = "";
for (var i = 0; i < strippedPhrase.length(); i++) {
    var c = strippedPhrase.charAt(i);
    var tryPrefix = builder.toString() + new String(new char[] { c });

    if (!hasPrefixSequence(first, second, tryPrefix)) {
        var word = extractWord(first, second, builder);
        if (word.isPresent()) {
            guess.add(word.get());
            first = second;
            second = word.get();
        }
    }

    builder.append(c);
}
```

The algorithm

```java
private Optional<String> extractWord(String first, String second, StringBuilder builder) {
    var splitPosition = 0;
    var maxFollowerCount = 0;

    for (var n = builder.length(); n > 0; n--) {
        var candidate = builder.substring(0, n);
        if (hasWordSequence(first, second, candidate)) {
            var follower = builder.substring(n, builder.length());
            var followerCount = prefixSequenceCount(second, candidate, follower);
            if (followerCount > maxFollowerCount) {
                maxFollowerCount = followerCount;
                splitPosition = n;
            }
        }
    }

    // if couldn't find a place to split, return whole buffer
    if (splitPosition == 0) {
        splitPosition = builder.length();
    }

    var word = builder.substring(0, splitPosition);
    builder.delete(0, splitPosition);

    return word.isEmpty() ? Optional.empty() : Optional.of(word);
}
```

Testing and debugging

- A lot of test data. I used the unix fortune database, a collection of aphorisms
  - everything takes longer than you think
  - running is not a plan running is what you do when the plan fails
- Problem statement has well defined mental models that map directly to code
- A change log with notes, bugs, fixes, changes and failed tests helps a lot

Does it work? More or less

- It solves the puzzle
  - there is no possible way => there is no possible way
- Problems with unknown words
  - <u>hello</u> world ⇒ **he llo w o r l d**
- Problems with zero occurrences of phrase in training text
  - The answer <u>you seek</u> is in an envelope ⇒ **the answer y o u s e e k i s i n a n e n v e l o p e**
- Improvements
  - Refine training data
  - Try both dictionary and markov and define a metric for readability

Was it fun?

- Explore new java features
  - Streams, lambdas, records, switch expressions, var, instanceof are a charm.
  - Java **the language**, is OK for small projects. Things have improved.
  - Java **the environment**, is not OK from small projects: IDEs, maven, CLI are not as simple as they could be. Disproportionate energy for small things.
    - Started with emacs and *java Mars.java*, ended with a full maven project on vscode.
- Revisit graph algorithms
  - Computer science is usually not a frequent visitor in our 9-5 jobs
- Have something relaxing to do
  - Creativity needs space, useless things and toys.
  - Very important to know when to stop, and do stop otherwise the fun is gone.

question st i m e