Computer Science Concepts for Data Scientists - Exercises

Table of contents

Week 1: Case Study: Bits & Bytes

Format: `IIIIIIII.FFFF` (8 bits integer, 4 bits fraction)

Example: `45.375`

1. Integer Part (`45`)

- **Method:** Repeatedly divide by 2 & note the remainder.
- **Calculation:**
  - `45 / 2 = 22` Remainder `1`
  - `22 / 2 = 11` Remainder `0`
  - `11 / 2 = 5` Remainder `1`
  - `5 / 2 = 2` Remainder `1`
  - `2 / 2 = 1` Remainder `0`
  - `1 / 2 = 0` Remainder `1`
- **Result (read remainders from bottom to top):** `101101`
- **Pad to 8 bits (on the left):** `00101101`

2. Fractional Part (`0.375`)

- **Method:** Repeatedly multiply by 2 & note the integer part.
- **Calculation:**
  - `0.375 * 2 = 0.75 -> 0`
  - `0.75 * 2 = 1.5 -> 1`
  - `0.5 * 2 = 1.0 -> 1`
- **Result (read digits from top to bottom):** `011`
- **Pad to 4 bits (on the right):** `0110`

3. Combine

- **Integer:** `00101101`
- **Fraction:** `0110`
- **Final (12-Bit):** `001011010110`

Binary to Decimal Conversion

Example: `11001.1101`

1. Integer Part (`11001`)

- **Calculation:**
  - `1 * 2^4` -> 16
  - `1 * 2^3` -> 8
  - `0 * 2^2` -> 0
  - `0 * 2^1` -> 0 `1 * 2^0` -> 1

Total = `25`

2. Fractional Part (`.1101`)

- **Calculation:**
  - `1 * 2^-1` -> 0.5
  - `1 * 2^-2` -> 0.25
  - `0 * 2^-3` -> 0
  - `1 * 2^-4` -> 0.0625

Total = 0.8125

3. Combine

- **Integer:** `25`
- **Fraction:** `0.8125`
- **Final (12-Bit):** `25.8125`

Special Case: Loss of Precision (Ex: `2.3`)

- Not every fraction can be represented exactly.
- **Integer** `2` -> `00000010`
- **Fraction** `0.3` -> `0.010011...` **(repeating)**
- **Truncated to 4 bits:** `0100` (which represents `0.25`)
- **Final result for 2.3:** `000000100100` (represents `2.25`)

Bits and Bites

Conversion logic (short form)

1. Decimal system (base 1000) – standard among hardware manufacturers:

- B -> KB -> MB -> GB -> TB
- Calculation: Value ÷ 1,000 (upwards) or × 1,000 (downwards).

2. Binary system (base 1024) – standard in operating systems (Windows):

- B -> KiB -> MiB -> GiB -> TiB
- Calculation: Value ÷ 1,024 (upwards) or × 1,024 (downwards).

Week 2: Computer Networks

Exercise 1: Internet Delays and Routes

a

Traceroute is a command line tool for determining the transport route of IP data packets between your own computer and a selected remote station.

```
traceroute gaia.cs.umass.edu
```

b

The number of routers along the path depends on the network route and topology — for example, whether a VPN or proxy is used, but not on the local access medium (WiFi vs. Ethernet).

Exercise 2: Internet Delays

a

$$\text{file} = 4\text{MB} = 4000\text{bytes} = 32000\text{bites}$$

$$R_s = 1\text{Mbit/sec} = \frac{32000}{1000} = 32\text{sec}$$

Exercise 3: Domains, Hosts and URLs

```
flowchart TD
    A(Root) --> B(edu)
    B --> C[umass]
    C --> E(cs)
    F[gaia]
    D[www]

    %% Subgraph trick
    subgraph row1[ ]
        direction LR
        C
        D
    end

    %% Subgraph trick
    subgraph row2[ ]
        direction LR
        E
        F
    end
```

Exercise 4: http Protocol

Whenever a user downloads a new web page a copy is saved on the local web server (cache). If the same page is requested again later by the same or another user AND it is still valid (i.e., the expiration date has not yet been reached), the page does not have to be fetched again from the remote web server.

Week 3: Web Applications & HTML

Exercise 2: Develop a web page - Task 1

```
<!DOCTYPE html>
<html>
  <head>
    Exercise 2: Develop a web page - Task 1
```

```
  </head>
  <body>
    <img src="img/siliconValley.jpg" hight="1000" width="500">
  </body>
</html>
```

Exercise 2: Develop a web page - Task 2

```
<!DOCTYPE html>
<html>
  <head>
    Exercise 2: Develop a web page - Task 2
  </head>

  <body>
    <h2>Links to well-known companies in Silicon Valley</h2>
    <img src="img/tesla.png">
    <h2>Link to overview map Silicon Valley</h2>
    <a href="https://siliconvalleyguide.info/">Map</a>

  </body>
</html>
```

Exercise 3: Develop a CSS file

```
<!DOCTYPE html>
<html>
<head>
    <title>Exercise 3: Develop a CSS file</title>

    <style>
      body {
        background-color: lightgoldenrodyellow;
        color: blue;
      }

      img {
        border: 2px solid blue;
```

```
        padding: 5px;
      }

      a {
        color: red;
      }
    </style>
</head>

<body>
    <h2>Links to well-known companies in Silicon Valley</h2>
    <img src="img/tesla.png" alt="Tesla Logo">
    <h2>Link to overview map Silicon Valley</h2>
    <a href="https://siliconvalleyguide.info/">Map</a>
</body>

</html>
```

Week 4: Algorithms

Exercise 1

Develop a recursive algorithm for the function `power(x, n)`.

```
def power(x, n):
    if n == 0:
        return 1
    else:
        return x * power(x, n - 1)

res = power(2, 3)
print(res)
```

```
8
```

Exercise 2

Develop a recursive algorithm for the product of the natural odd numbers from `m` to `n`: `product(m, n)`.

```python
def product(m: int, n: int) -> int:
  """ Returns the product of the odd numbers from m to n """
  # Check for only odd number inputs
  assert m % 2 == 1 and n % 2 == 1

  # Recursion
  if m == n:
    return m
  else:
    return m * product(m + 2, n)


res = product(3, 11)
print(res)
```

```
10395
```

Exercise 3

Linear Search

Linear search sequentially checks each element of the list until a match is found or the whole list has been searched. For a list of length n results a complexity of $O(n)$.

Binary Search

Binary search compares the target value to the middle element of the list. For a list of length n results a complexity of $O(\log(n))$.