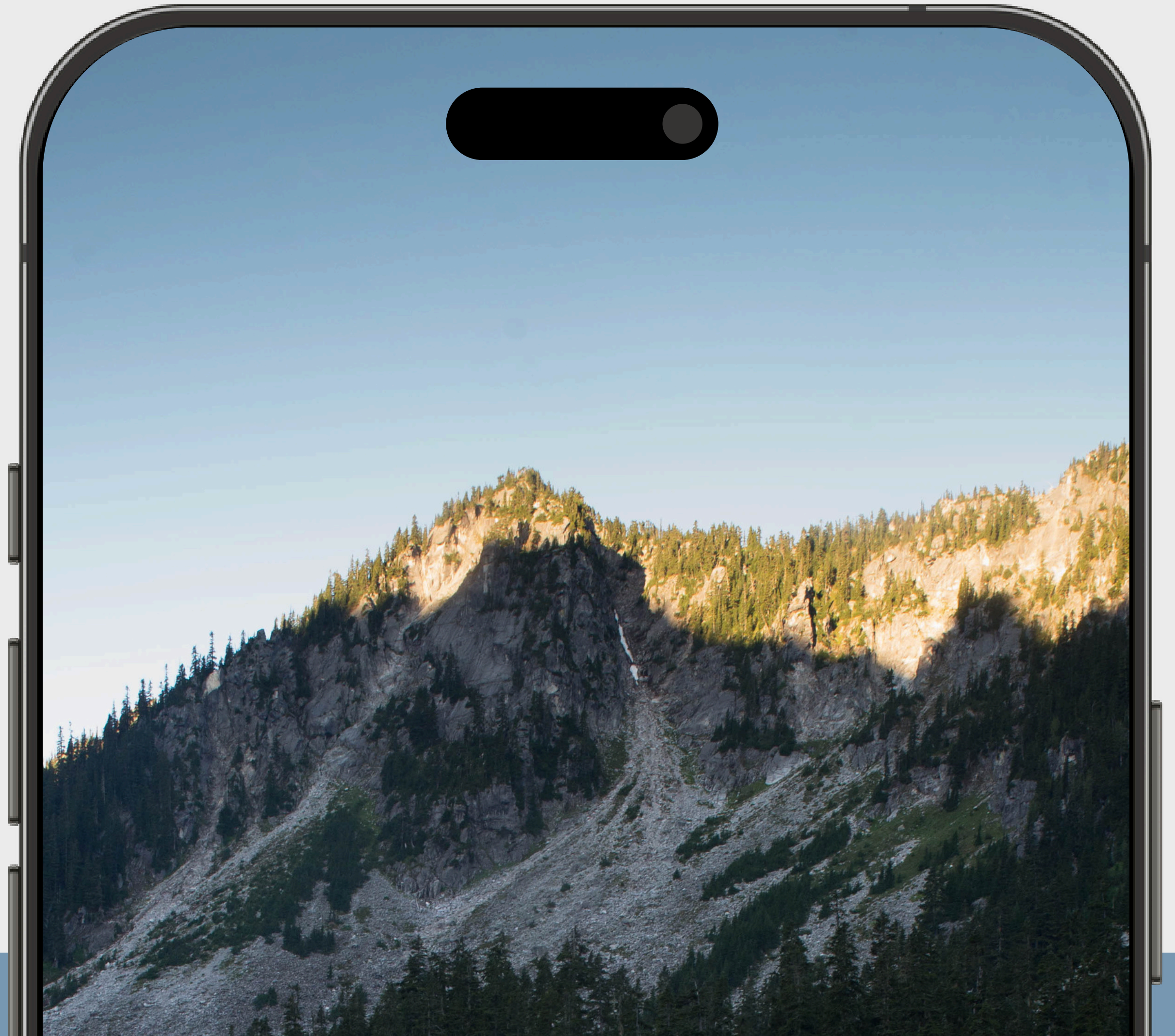
Dare2Pair

Case Study

Solving Real-world Problems using
Dynamic Progrgramming



Naira Carigo
Team Member

Micki Corpuz
Team Member

Dare2Pair

Welcome

Executive Summary

PROBLEM

Efficiently managing mobile data to maximize screen time while staying within a data limit.

OBJECTIVE

Select apps that provide the longest total usage without exceeding the data cap.

ALGORITHM

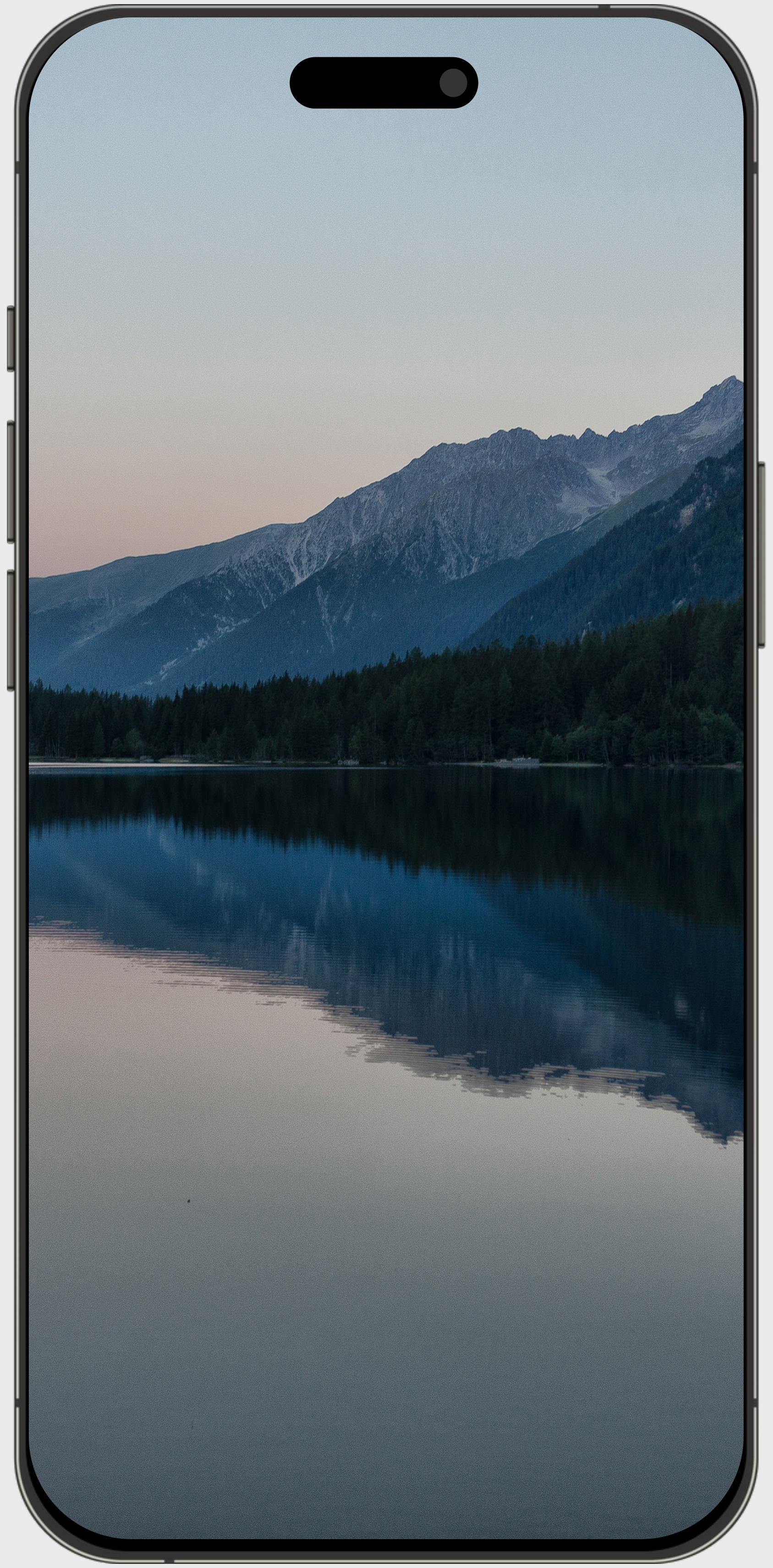
A memoized recursive approach based on the 0/1 Knapsack Problem, optimizing app selection for maximum screen time while staying within the data limit.

Dare2Pair

Iteration 1

Problem Identification

How can we maximize screen time while ensuring total app data usage stays within a given data limit?



Decomposition

- Break down apps by data usage, screen time, and selection constraints.

Pattern Recognition

- Choosing apps efficiently leads to better total screen time.
- Recursion and memoization help optimize repeated calculations.

Abstraction

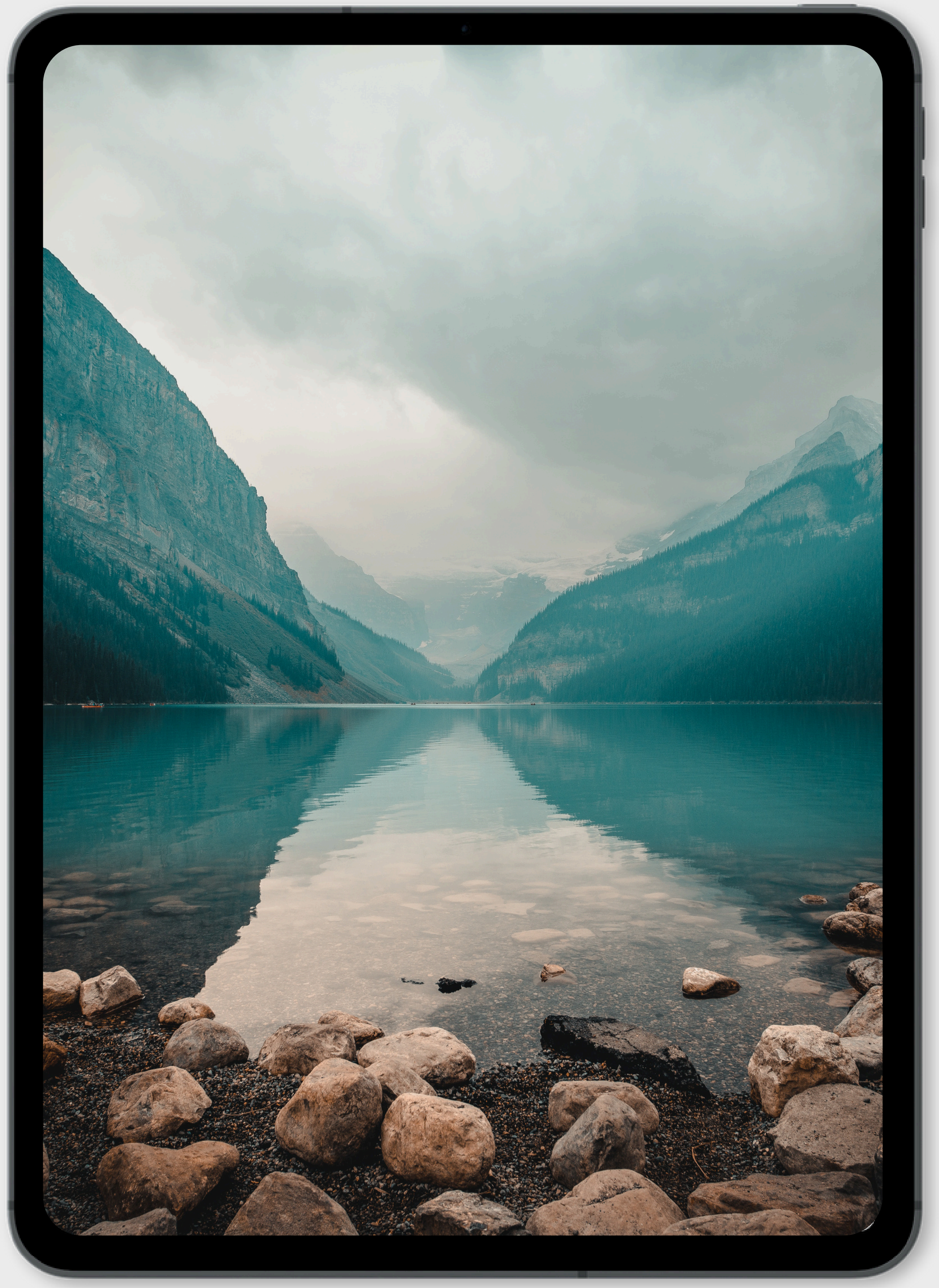
- Focus on data usage and screen time, ignoring app-specific details.

Dare2Pair

Iteration 2

Problem Identification

How can we maximize screen time using boost factors?



Decomposition

- Define boos factors (frequency of use)
- Optimized app selection with boosted screen time

Pattern Recognition

- Apps with high boost factors yield greater screen time without consuming much data
- Combining high-usage and low-usage apps help stay within data limit

Abstraction

- Focus on maximizing the screen times based on boost factor

Dare2Pair

Code Implementation

```python
class App:
    def __init__(self, name, data_usage, screen_time, boost_factor=1.0):
        self.name = name
        self.data_usage = data_usage
        self.screen_time = screen_time
        self.boost_factor = boost_factor

    def __str__(self):
        boosted_time = self.getBoostedScreenTime()
        return f"{self.name} <{self.data_usage}MB/hr, {self.screen_time} hrs, Boosted: {boosted_time:.2f} hrs>"

    def getDataUsage(self):
        return self.data_usage

    def getScreenTime(self):
        return self.screen_time

    def getBoostedScreenTime(self):
        return self.screen_time * self.boost_factor

def buildAppList(name, data_usage, screen_time, boost_factors):
    return [App(name[i], data_usage[i], screen_time[i], boost_factors[i]) for i in range(len(name))]
```

Dare2Pair

Code Implementation

```python
def maxUsage(toConsider, available, cache=None):
    if cache is None:
        cache = {}

    if not toConsider or available == 0:
        return (0, ())

    key = (len(toConsider), available)
    if key in cache:
        return cache[key]

    first = toConsider[0]

    if first.getDataUsage() > available:
        result = maxUsage(toConsider[1:], available, cache)
    else:
        withTime, withApps = maxUsage(toConsider[1:], available - first.getDataUsage(), cache)
        withTime += first.getBoostedScreenTime()

        withoutTime, withoutApps = maxUsage(toConsider[1:], available, cache)

        if withTime > withoutTime:
            result = (withTime, withApps + (first,))
        else:
            result = (withoutTime, withoutApps)

    cache[key] = result
    return result
```

Dare2Pair

Code Implementation

```
def testMaxUsage(apps, max_data):
    bestTime, bestSelection = maxUsage(apps, max_data)
    print("=" * 50)
    print(f" Best total screen time: {bestTime:.2f} hours")
    print(" >>>>", " " * 8, "Selected Apps", " " * 8, "<<<<")
    for app in bestSelection:
        print(f" • {app}")
    print("=" * 50)

app_names = ['YouTube 360p', 'YouTube 1080p', 'Facebook Scrolling',
             'Facebook Videos', 'Instagram', 'TikTok', 'Twitter',
             'Netflix 1080', 'Netflix 4K', 'Canvas LMS']

data_usage = [300, 2000, 80, 160, 500, 900, 360, 1000, 7000, 5000]
screen_time = [1, 0.5, 1, 0.5, 1, 1, 1, 1, 1, 1]

boost_factors = []
print("Enter boost factor for each app:")
for name in app_names:
    boost = float(input(f"{name}: "))
    boost_factors.append(boost)

apps = buildAppList(app_names, data_usage, screen_time, boost_factors)

testMaxUsage(apps, 2000)
```

Dare2Pair

Optimal Solution

```
Enter boost factor for each app:
YouTube 360p: 1
YouTube 1080p: 1
Facebook Scrolling: 1
Facebook Videos: 1
Instagram: 1
TikTok: 1
Twitter: 1
Netflix 1080: 1
Netflix 4K: 1
Canvas LMS: 1
==================================================
 Best total screen time: 4.50 hours
 >>>>          Selected Apps          <<<<
 • Twitter <360MB/hr, 1 hrs, Boosted: 1.00 hrs>
 • TikTok <900MB/hr, 1 hrs, Boosted: 1.00 hrs>
 • Instagram <500MB/hr, 1 hrs, Boosted: 1.00 hrs>
 • Facebook Videos <160MB/hr, 0.5 hrs, Boosted: 0.50 hrs>
 • Facebook Scrolling <80MB/hr, 1 hrs, Boosted: 1.00 hrs>
==================================================
```

Dare2Pair

Thank You

Chiaki Sato
Founder & CEO

www.reallygreatsite.com
hello@reallygreatsite.com