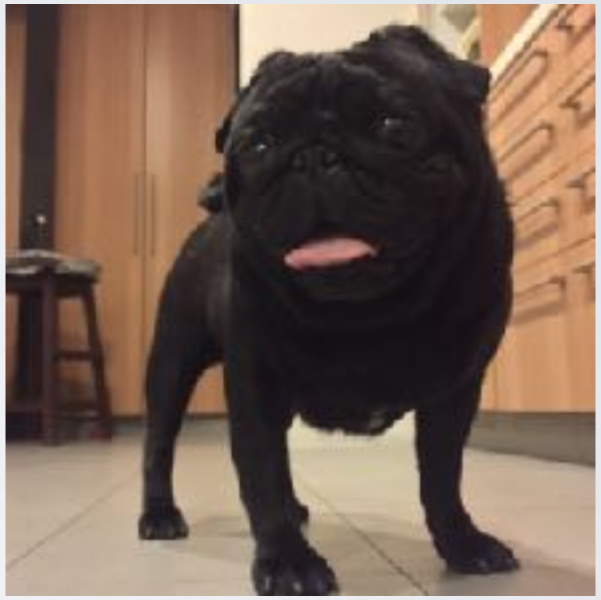
Online Algorithms and Competitive Analysis

CS16: Introduction to Data Structures & Algorithms

Spring 2019

Outline

1. Motivation
2. The Ski-Rental Problem
3. Experts Problem
4. Dating Problem



Motivation

- We don’t always start off with all the data we want at once
- We want the best algorithms to answer questions about such data

Offline Algorithms

- An ***offline algorithm*** has access to all of its data at the start – it "knows" all of its data in advance
- Most of what you have done in this class has been offline (or at least given offline)

Online Algorithms

- An ***online algorithm*** does not have access to all of the data at the start
- Data is received **serially**, with no knowledge of what comes next
- How do you make a good algorithm when you don't know the future?

Ski-Rental Problem

- You like skiing
- You're going to go skiing for **n** days
- You need to decide: Do you rent skis or buy skis?
- Renting:
  - $50 per day
- Buy:
  - $500 once
- Goal: Minimize cost

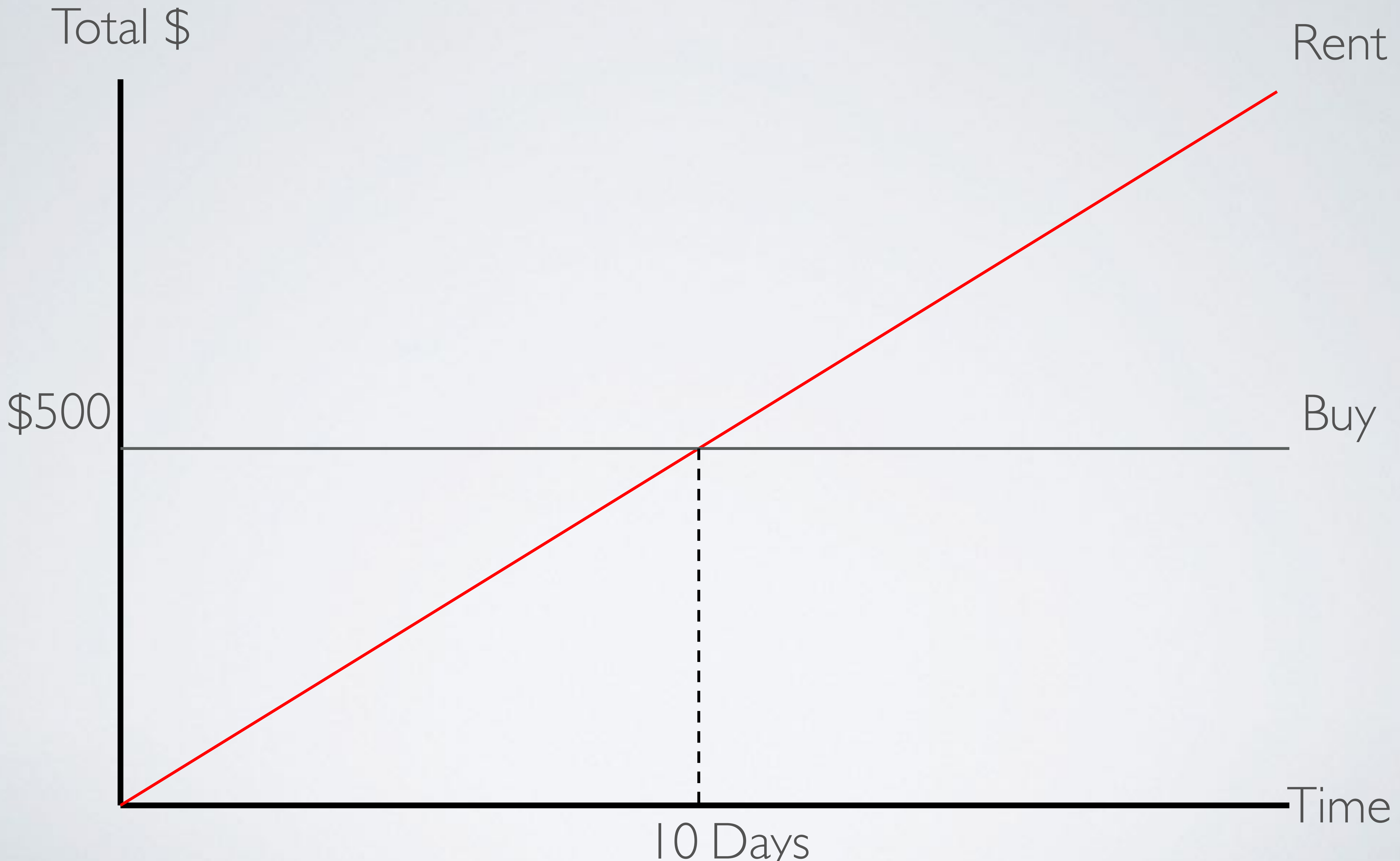
Maggie hitting the slopes

Ski-Rental Problem (Offline)

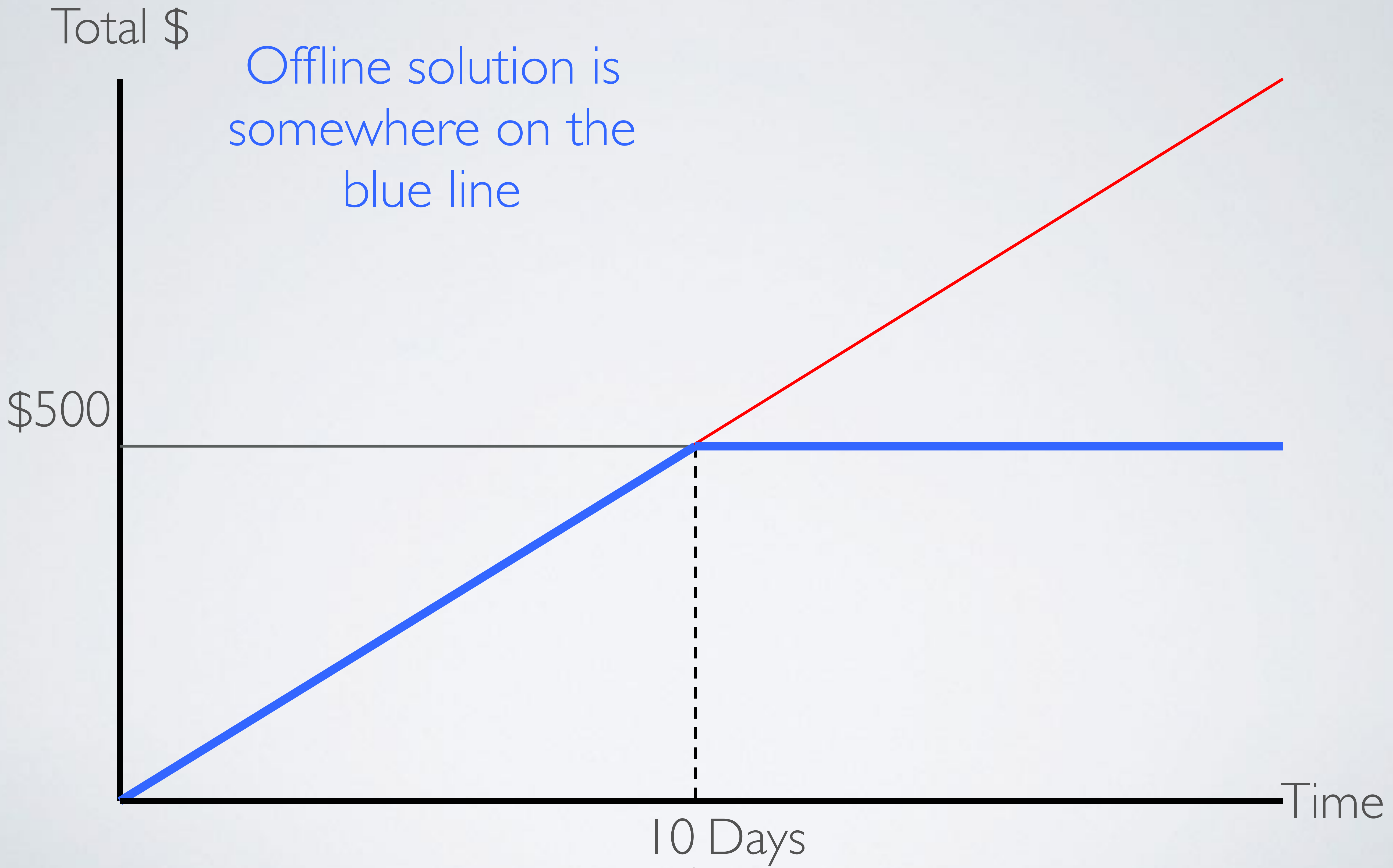
- Offline solution:
  - If $n < 10$, rent!
  - Else, buy!
- Tough luck. You don't know what n is
  - You love skiing so much, you'll ski as long as you don't get injured

Alina Shredding Gnar

Ski-Rental Problem – Rent vs. Buy



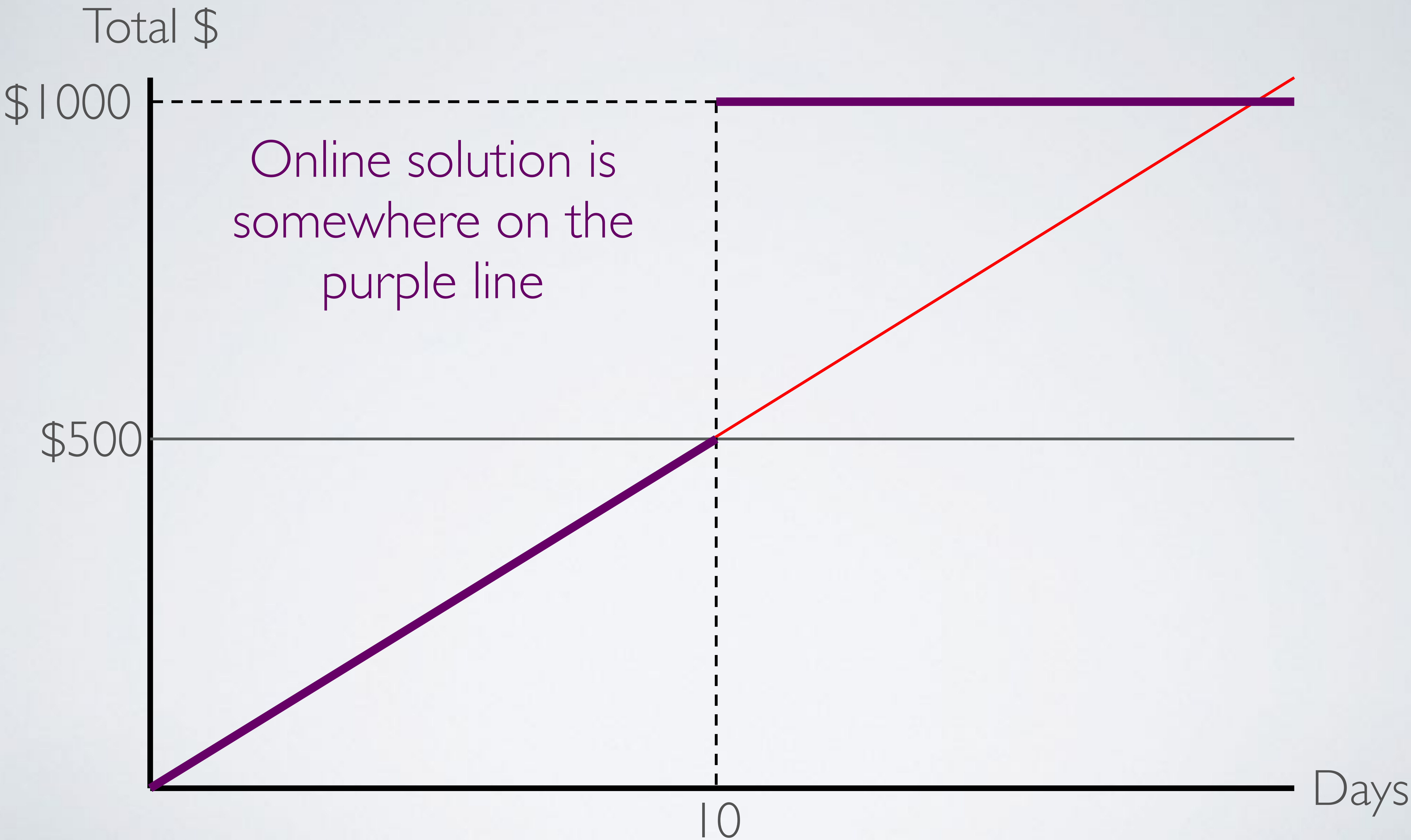
Ski-Rental Problem (Offline)



Ski-Rental Problem (Online)

- We don't know the future, so what can we do?
- Try to get within some constant multiplicative factor of the offline solution!
  - "I want to spend at most **x** times the amount the offline solution would spend"
- Strategy:
  - Rent until total spending equals the cost of buying
  - Then buy if we want to ski some more

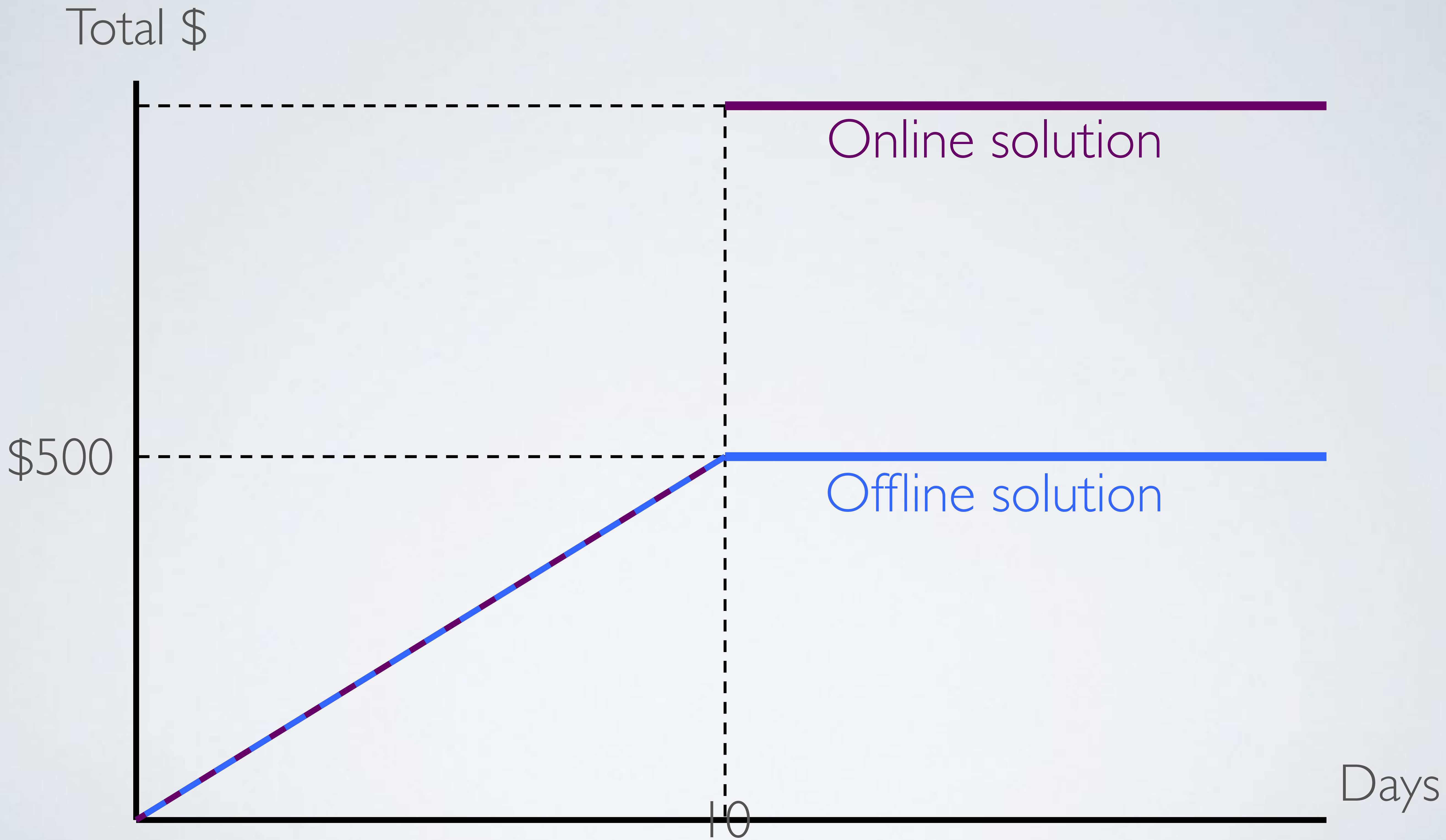
Ski-Rental Problem (Online)



Ski-Rental Problem – Analysis

- How good is this?
  - If we ski 10 days or less, we match the optimal solution!
  - If we ski more than 10 days, we never spend more than twice the offline solution
- This is not the only online solution!

Comparing Solutions



How good is this?

- How do we know that our algorithms are "good", i.e. close to optimal?
- What can we do if we don't even know what the most optimal algorithm is?

Competitive Analysis

- Analyzing an online algorithm by comparing it to an offline counterpart
- *Competitive ratio*: Ratio of performance of an online algorithm to performance of an optimal offline algorithm

$$\text{perf online } \leq c \cdot \text{ perf offline } + \alpha$$

- Our ski-rental solution has a competitive ratio of 2, since we are never more than 2 times as bad as the offline solution
- Our online algorithm is "2-competitive" with the offline solution

More than just skis…

- Refactoring versus working with a poor design
- Unrequited love?

The Experts Problem

- Dating is hard
- You know nothing about dating (oops)
- Dating can be reformulated as a series of binary decisions
  - Not "What should I wear?", but
    - Do I wear these shoes? (yes)
    - Should we go at 7? Should we go at 8? (7)
    - Do I wait 15 minutes to text them back? Or 3 hours? (3)
    - Should I buy them flowers? (no)

The Experts Problem: The Scenario

- You know nothing, so you should ask for help
- You know **n** experts who can give you advice before you make each decision (but you don't know if it's good)

The Experts Problem

- Rules
  - If you make the right decision, you gain nothing
  - If you make the wrong decision, you get 1 unit of embarrassment
  - Total embarrassment = number of mistakes

- Goal: Minimize total embarrassment (relative to what the best expert would've gotten)

The Experts Problem (Offline)

- Offline:
  - We know the best expert
  - We only listen to them
  - Whatever successes and mistakes they have, we have
- Online:
  - We don't know the best expert

The Experts Problem (Online)

- Assign every expert a weight of 1, for total weight of **W = n** across all experts
- Repeat for every decision:
  - Ask every expert for their advice
  - *Weight* **their advice and decide by majority vote**
  - After the outcome is known, take every expert who gave bad advice and cut their weight in half, regardless of whether your bet was good or bad

Lecture Activity 1

Fill in the blank weights on your sheet!

Round 1: Should I buy them flowers?
What do the experts say?

- Expert 1, 2, 3 say no
- Expert 4, 5 say yes

Correct Answer? **Yes!**

Round 2: Should I show up fashionably late?
What do the experts say?

- Expert 3, 5 say no
- Expert 1, 2, 4 say yes

Correct Answer? **No!**

2 Mins….

Lecture Activity 1

Fill in the blank weights on your sheet!

Round 1: Should I buy them flowers?
What do the experts say?

- Expert 1, 2, 3 say no
- Expert 4, 5 say yes

Correct Answer? **Yes!**

Round 2: Should I show up fashionably late?
What do the experts say?

- Expert 3, 5 say no
- Expert 1, 2, 4 say yes

Correct Answer? **No!**

1 Min….

Lecture Activity 1

Fill in the blank weights on your sheet!

Round 1: Should I buy them flowers?
What do the experts say?

- Expert 1, 2, 3 say no
- Expert 4, 5 say yes

Correct Answer? **Yes!**

Round 2: Should I show up fashionably late?
What do the experts say?

- Expert 3, 5 say no
- Expert 1, 2, 4 say yes

Correct Answer? **No!**

0 Mins….

Lecture Activity 1 Answers

Updating the weights

Weights	Expert 1	Expert 2	Expert 3	Expert 4	Expert 5
Initial	1	1	1	1	1
Round 1	0.5	0.5	0.5	1	1
Round 2	0.25	0.25	0.5	0.5	1

- Let's see how we can make a decision in the third round!
- Remember, sum the weights of the experts of both options and pick the majority value!

Lecture Activity 2

Which decision should we make?

Round 3: Should I order the clams and garlic?
What do the experts say?

- Expert 1, 2, 3 say yes
- Expert 4, 5 say no

1 Min….

Lecture Activity 2

Which decision should we make?

Round 3: Should I order the clams and garlic?
What do the experts say?

- Expert 1, 2, 3 say yes
- Expert 4, 5 say no

0 Mins….

Lecture Activity 2 Answer

Which decision should we make?

Round 3: Should I order the clams and garlic?
What do the experts say?

- Expert 1, 2, 3 say yes
- Expert 4, 5 say no

Majority Decision
Yes sum: 0.25 + 0.25 + 0.5 = 1
No sum: 0.5 + 1 = 1.5
Majority answer is **No**, so we don't eat clams and garlic! Good choice…

- How good is our algorithm?

Multiplicative Weights Algorithm - Analysis

- To analyze how good this is, we need to relate the number of mistakes we make (**m**) to the number of mistakes the best expert makes (**b**)
- How can we do that? Use the weights!
- Let **W** represent the sum of the weights across the **n** experts at an arbitrary point in the algorithm

Experts Algorithm - Analysis

- Look at the total weight assigned to the experts
- When the best expert makes the wrong decision…
  - We cut their weight in half
  - They started out with a weight of 1

$$\left(\frac{1}{2}\right)^b \leq W$$

Experts Algorithm - Analysis

- Look at the total weight assigned to the experts
- When we made the wrong decision…
  - At least ½ weight was placed on the wrong decision
  - We will cut at least ¼ of **W**, so we will reduce the total weight to at most ¾ of W
  - Since we gave the experts n total weight at the start:

$$W \leq n \left(\frac{3}{4}\right)^m$$

Experts Algorithm - Analysis

$$W \leq n\left(\frac{3}{4}\right)^m$$

$$\left(\frac{1}{2}\right)^b \leq W$$

$$\left(\frac{1}{2}\right)^b \leq W \leq n\left(\frac{3}{4}\right)^m$$

Experts Algorithm - Analysis

$$\left(\frac{1}{2}\right)^b \le W \le n\left(\frac{3}{4}\right)^m$$

$$\left(\frac{1}{2^b}\right) \le W \le n\left(\frac{3}{4}\right)^m$$

$$\left(\frac{1}{2^b}\right) \le n\left(\frac{3}{4}\right)^m$$

$$-b \le \log_2 n + m\log_2\left(\frac{3}{4}\right)$$

$$-b - \log_2 n \le m\log_2\left(\frac{3}{4}\right)$$

$$b + \log_2 n \ge m\log_2\left(\frac{4}{3}\right)$$

$$\frac{(b+\log_2 n)}{\log_2\left(\frac{4}{3}\right)} \ge m$$

So the number of mistakes we make, ***m***, is at most 2.41 times the number of mistakes the best expert makes, ***b***, plus some change

We want THE BEST

- How to find the best…
  - apartment?
  - deal for a ticket?
  - class to take?
  - partner?
- How can we know that they’re the best?
- How much effort are we willing to spend to find the best one?

The {Best Choice, Dating} Problem

- Also known as the secretary problem
- There are **n** people we are interested in, and we want to end up dating the best one
- Assumptions:
  - People are consistently comparable, and *score*(***a***) != *score*(***b***) for arbitrary people ***a*** and ***b***
  - You don't know anyone's score until you've gone on at least one date with them
  - Can only date one person at a time (serial monogamy)
  - Anyone you ask to stay with you will agree to do so

Dating Problem

- What's the offline solution?
  - If you already know everyone's score, just pick the best person
- A naïve online solution?
  - Try going out with everyone to assign them scores, and ask the best person to take you back
- Problems:
  - Takes a lot of time / money, depending on **n**
  - Assumes that they will take you back

Dating Problem

- Two main constraints:
  - You can’t look ahead into the future
  - There’s no “undo” – if you let someone go, chances are they’ll be taken by the time you ask for them back
- In other words, the problem is: Do I reject the current possibility in hopes of landing something better if I keep looking, or do I stick with what I have?

Dating Problem

- Solution:
  - Pick a random ordering of the **n** people
  - Go out with the first **k** people.
  - No matter how the dates go, reject them (calibration of expectations)
  - After these **k** dates, pick the first person that's ***better than everyone we've seen so far,*** and stick with them – they're probably the best candidate

Dating Problem - Analysis

- What value of **k** maximises our chances of ending up with the best person?
- 3 Cases to consider:
  - What if the best person is in the first **k**?
    - We end up alone. Oops.
  - What if the person that we pick isn't actually the best?
    - Oh well, we live in blissful ignorance
  - Otherwise, we successfully pick the best person!

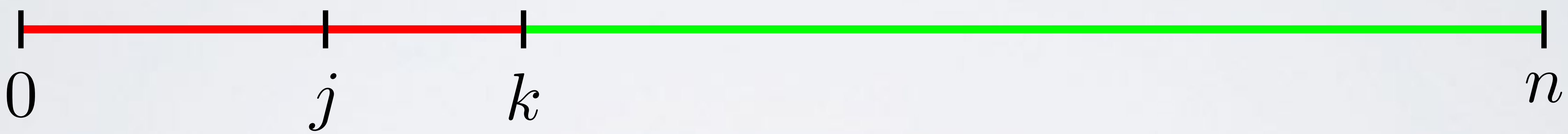
Dating Problem - Analysis

- Consider the candidate at position **j**
- Let's first consider the probability that the algorithm pairs us with this candidate, given a value of **k**

$$P_{choose}(k,j) = \begin{cases} 0 & \text{if } j \leq k, \\ \dfrac{k}{j-1} & \text{otherwise} \end{cases}$$

Dating Problem - Analysis

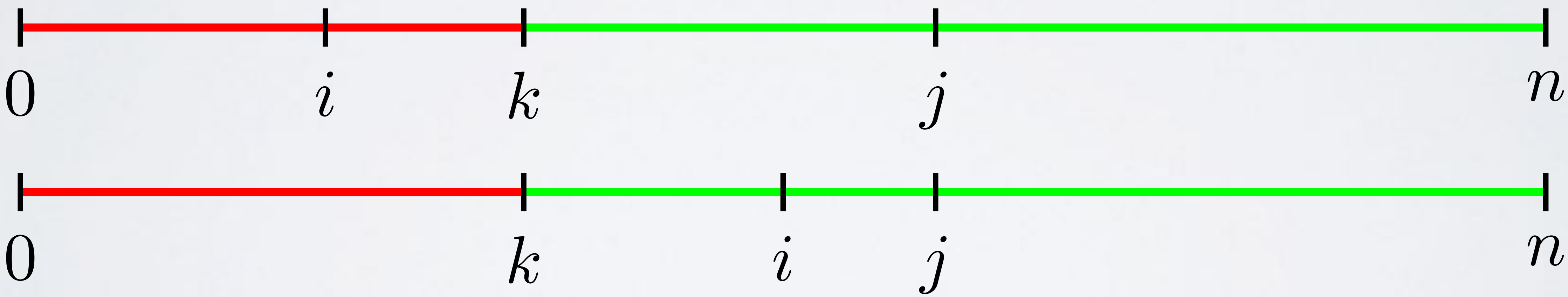
- Consider the candidate at position **j**
- Case 1



$$P_{choose}(k,j) = \begin{cases} 0 & \text{if } j \leq k, \\ \dfrac{k}{j-1} & \text{otherwise} \end{cases}$$

Dating Problem - Analysis

- Case 2:
  - There exists some person at position *i* who has the highest score we've seen so far by the time we're considering the jth person



$$P_{choose}(k, j) = \begin{cases} 0 & \text{if } j \leq k, \\ \dfrac{k}{j-1} & \text{otherwise} \end{cases}$$

Dating Problem - Analysis

- The probability that the **j**th person actually ***is*** the best is 1 / n)
- For a given **k**, the probability that we end up with the best person, P_{best}, is the sum of the conditional probabilities for each valid value of **j**

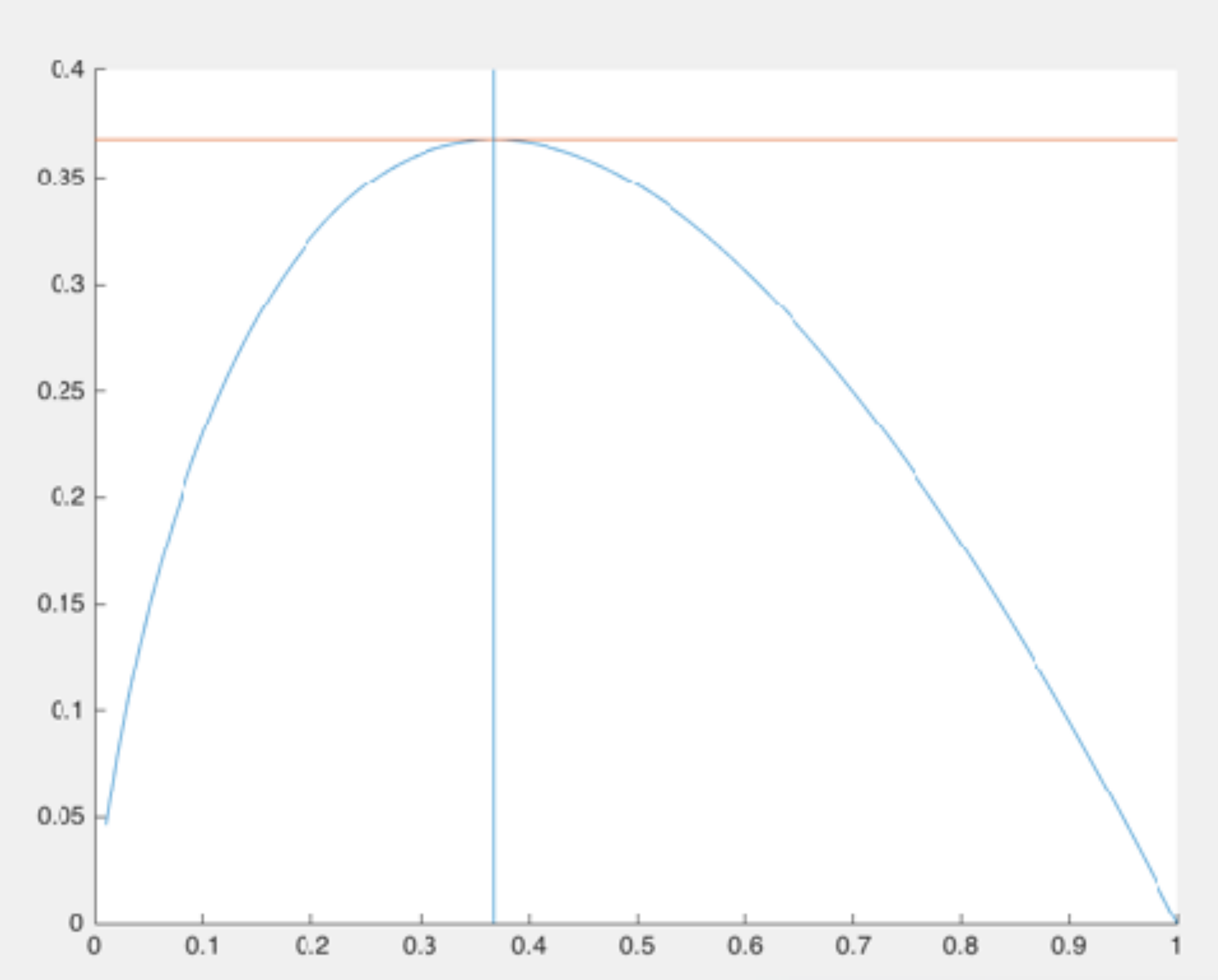
$$P_{best}(k) = \sum_{j=k+1}^{n} \left(\frac{k}{j-1}\right)\left(\frac{1}{n}\right)$$

Dating Problem - Analysis

$$P_{best}(k) = \sum_{j=k+1}^{n} \left(\frac{k}{j-1}\right)\left(\frac{1}{n}\right)$$

$$= \left(\frac{k}{n}\right) \sum_{j=k+1}^{n} \left(\frac{1}{j-1}\right)$$

$$\lim_{n\to\infty} \approx -\left(\frac{k}{n}\right) \ln\left(\frac{k}{n}\right)$$



In the above graph, what's the k/n value that maximizes p_{best}?
And what's the maximum value?

Dating Problem - Analysis

- **1/e**, for both the maximum value of P_{best} and the maximizing input for k/n

$$\frac{1}{e} = \frac{k}{n} \implies k = \frac{n}{e}$$

- So, with 1/e = 36.79% probability, if your strategy is to date the first person better than everyone in the first 36.79% of dates, you'll end up with the best person!

Dating Problem - Improvements

- **1/e** probability of not ending up with anyone :(
  - Strategy: Be desperate
    - Pick the last person, if you get that far
    - With probability 1/e, we pick the last person who will have, on average, rank n/2, so we'll probably be ok
  - Strategy: Gradually lower expectations
    - Pick a series of timesteps, t_0, t_1, t_2, t_k…
    - Reject the first t0 dates as before
    - Look for the best person we've seen so far between dates t_0 and t_1
    - If we find them, great!
    - Otherwise, between dating the (t_1+1)th and t_2th people, look for either the first or the second best we haven't yet dated
    - Repeat the above, gradually accepting a larger "pool"
    - We'll probably do better than the "be desperate" strategy, though by how much is hard to say without hardcore math

Recap

- An ***online algorithm*** is an algorithm where input is fed to you piece by piece, which makes writing a fast and optimal algorithm much more difficult
- Competitive analysis frames an online algorithm's efficiency in terms of an offline solution

CS Applications

- CPUs and memory caches (CS33, CS157)
  - Intel pays major $$$ for good caching strategies
- Artificial intelligence (CS141)
  - Heuristics, search, genetic algorithms
- Machine learning (CS142)
- Statistics

More Applications (continued)

- Economics
  - Stocks and trading
  - Game theory
  - Gambling
- Biology (featuring 2 authors of our textbook, Papadimitriou and Varizani)
  - https://www.quantamagazine.org/20140618-the-game-theory-of-life/
  - Our textbook: Dasgupta, Papadimitriou, and Varizani
  - Evolution as a balance between fitness and diversity, given an unknown future