Advanced Algorithms

April 14, 2026

Logistics

- Assignment due next Tuesday, April 21

- Meet with me to discuss / lock in final project topic
  - Will have time after class today (until 4)
  - Keep in mind **scope**

- Tuesday and Thursday next week **I will be gone**
  - **Tues**: workshop day! with Kartika in class. Attendance taken
  - **Thurs**: Online recorded lecture, no need to come to class.
  - Will be available by email, Zoom!

Online Algorithms

- Until now: primary concern computational power
  - “Efficiently” solvable problems
  - NP-hard problems

- Next two weeks: difficulty is lack of information

- Online Model
  - Input / data arrives over time
  - Need to make irrevocable decisions without knowing future

Ski Rental Problem

- You want to go skiing as much as possible this season.
- But you don’t have skis, and the season could end any day

- Every day you have the option of:
  - **Renting** skis: $50 per day
  - **Buying** skis: $500 one time cost

- Should you **rent** or **buy** skis? When?

Secretary Problem

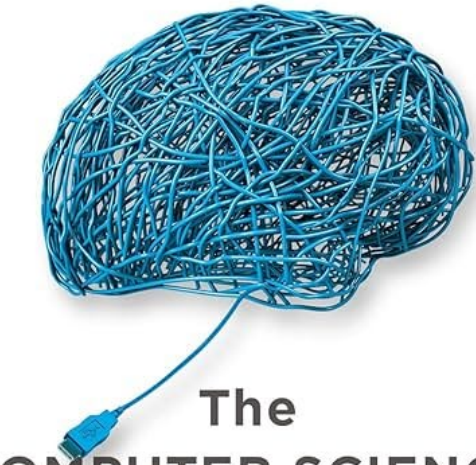
- You are hiring a secretary, and you want the best person for the job

- N candidates arrive one by one, revealing their quality after being interviewed

- After each candidate arrives, you must either:
  - **Hire** that candidate
  - **Reject** that candidate forever

Challenge?

Algorithms to Live By

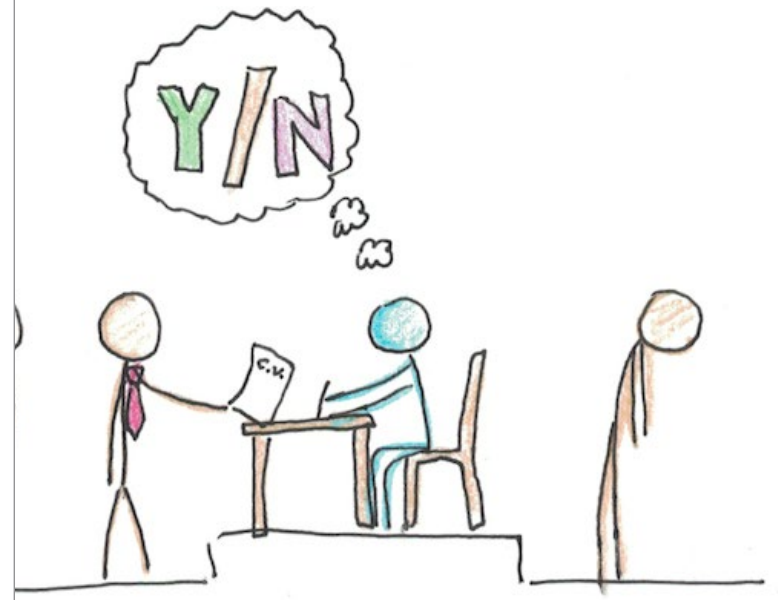


The COMPUTER SCIENCE of HUMAN DECISIONS

Brian Christian and Tom Griffiths

"Compelling and entertaining…Whether you want to optimize your to-do list, organize your closet, or understand human memory, this is a great read." —Charles Duhigg, author of *Smarter Faster Better*

PICADOR



Ski Rental Problem

- You want to go skiing as much as possible this season.
- But you don’t have skis, and the season could end any day

- Every day you have the option of:
  - **Renting** skis: $50 per day
  - **Buying** skis: $500 one time cost

- Should you **rent** or **buy** skis?

- Algorithms / policies?

Possible Algorithms

- Long-term view: **buy** on day 1
- Short-term view: **rent** until the season ends
- Mixture?
    - **Rent** for 4 days, then **buy** if the season is still going on

How do we evaluate these algorithms?

Analyzing Online Algorithms

- We define the optimal clairvoyant algorithm OPT
  - This algorithm knows the entire input in advance and can make decisions accordingly
  - The cost of this algorithm on input σ is OPT(σ)

- The competitive ratio c of an online algorithm ALG is:

  - Maximum over all inputs σ: $\mathrm{ALG}(\sigma)/OPT(\sigma)$

- This is very similar to approximation algos (consider $c = 2$)

Buying cost = $500
Renting cost = $50

Competitive Ratios

- Long-term view: **Buy** immediately on day 1 of the season
  - Worst-case scenario?
  - Competitive ratio?

- Short-term view: **Rent** until season ends
  - Bad scenario?
  - Competitive ratio?

- Mixture: **Rent** for 4 days, then **buy** if the season is still going on
  - Bad scenario? Competitive Ratio?

Optimal Strategy

- Observation: every deterministic algorithm is "buy on day k"
  - Worst case situation if we buy on day k is season ends after k days
- Buy on day 5:
  - Bad scenario – season ends after 5 days.
  - Clairvoyant Optimum just rents. Ratio is 2.8
- Buy on day 15:
  - Bad scenario – season ends after 15 days
  - Clairvoyant Optimum buys immediately! Ratio = 2.4

What is the best day to buy on?

The Better Late than Never algorithm

- We buy the skis exactly when we realize we should have done so in hindsight

- In our setup, that is on day $k = 10$

- **Theorem**: BLTN never pays more than 2 times the clairvoyant optimum. That is, BLTN is 2-competitive.

Broadly useful idea!

Your turn!

The Elevator Problem

Goal: get to the second floor of Edmunds as fast as possible

- Elevator takes 15 seconds once it arrives.

- Stairs takes 45 seconds, can go anytime.

You don’t know how long it will take the elevator to show up

Generalized Ski-rental

- Costs $\$r$ to rent and $\$b$ to buy. Assume r divides b for simplicity.
- BTLN algorithm:
  - Buy on day B := b/r
- **Theorem:** BTLN has competitive ratio $2 - \frac{r}{b}$
- Zoo passes, Rock climbing membership, Disney Land . . .

Summary

- We saw a deterministic 2-competitive algorithm for Ski Rental
- Turns out can’t do better than this using **deterministic** algorithms
- Must utilize randomness

Optimal Algorithm

Theorem (Karlin, Manasse, McGeoch, Owicki (SODA '90))

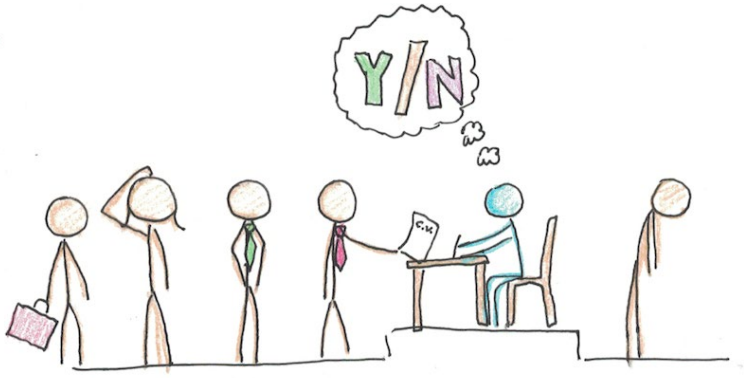
If we set $p_t = \left(\frac{b-1}{b}\right)^{b-t} \frac{1}{b\left(1-\left(1-\frac{1}{b}\right)^b\right)}$ *for all* $t \leq b$, *then*

- $\frac{E[ALG(\sigma)]}{OPT(\sigma)} \leq \frac{e}{e-1} \approx 1.58$ *for all* σ, *and*
- *this is optimal.*

A simpler random protocol

- Flip a coin in the beginning:
  - With probability 1/2 , run same algorithm as before (buy on day B)
  - With probability 1/2 , buy on day ¾ B
- Expected cost of our algo: at most $\frac{15}{8}$ times OPT for any B

Secretary Problem

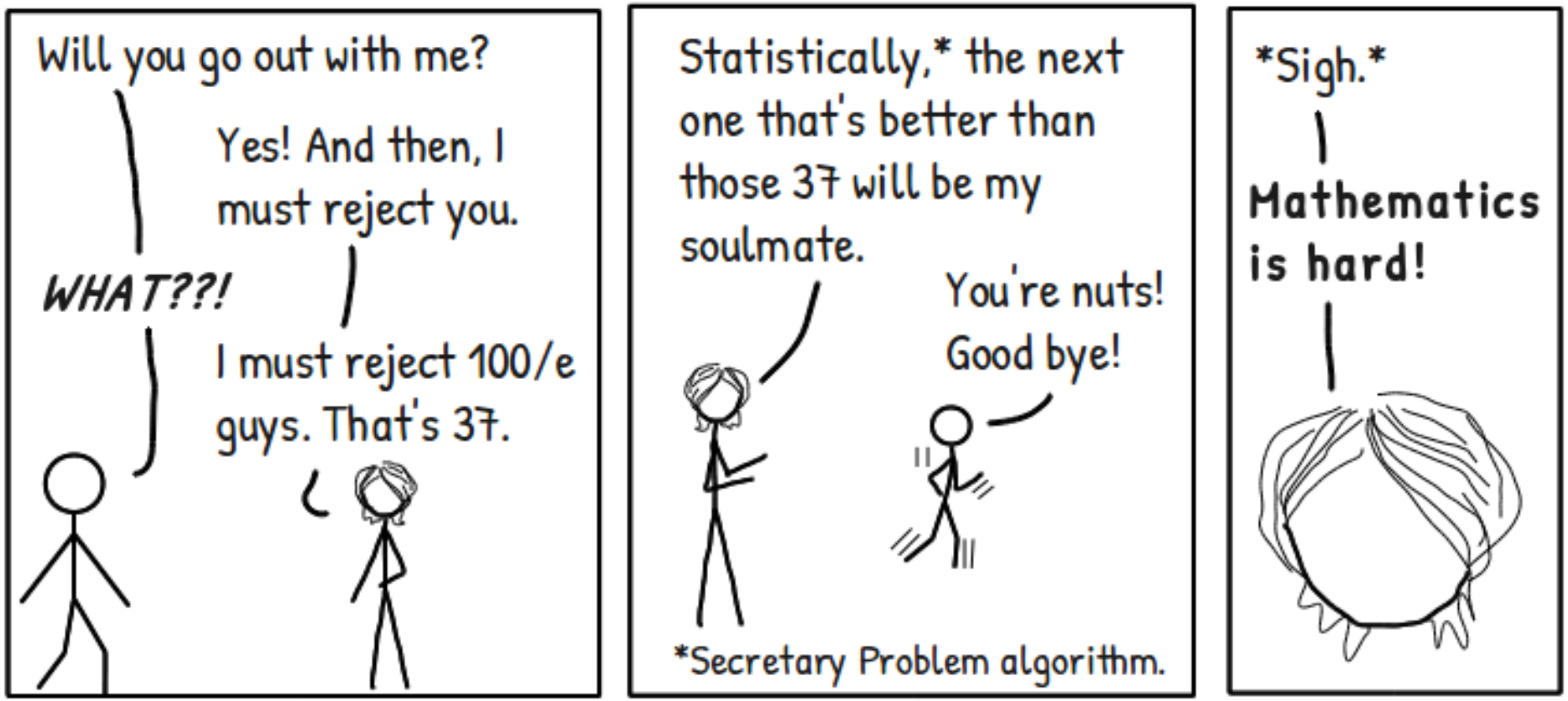


- N arrivals, each with a score
- Accept or reject each candidate as they arrive
- Goal: Pick top candidate with greatest probability
- **Key assumption: the candidates arrive in random order**

Optimal Algorithm

- Threshold policy:
  - Reject the first N / e candidates (this is roughly the first 37%)
  - Keep track of the best candidate you saw in that set, say with score T
  - Henceforth, pick any candidate with score at least T

Probability of picking best candidate → 1/e ≈ 0.37



Soulmate Algorithm -- I.

icanbarelydraw.com CC BY-NC-ND 3.0

Optimal Algorithm

- Threshold policy:
  - Reject the first **N / e** candidates (this is roughly the first 37%)
  - Keep track of the best candidate you saw in that set, say with score T
  - Henceforth, pick any candidate with score at least T

Probability of picking best candidate → 1/e ≈ 0.37

Simplified Algorithm

- Threshold policy:
  - Reject the first **N / 2** candidates (first 50%)
  - Keep track of the best candidate you saw in that set, say with score T
  - Henceforth, pick any candidate with score at least T

Probability of picking best candidate ≥ ___1/4___ Not bad!

More Online Problems

- You already know
  - Load Balancing
  - Online Bipartite Matching

- Many more
  - Caching problems – what to evict?
  - Online flows, bin packing, Steiner tree, set cover, knapsack . . .

Randomness helps in Online Algos!

More Online Problems

- You already know
  - Load Balancing
  - Online Bipartite Matching

- Many more
  - **Caching problems** – what to evict?
  - Online flows, bin packing, Steiner tree, set cover, knapsack . . .