COMP 3011
DESIGN AND ANALYSIS OF ALGORITHMS
FALL 2024

Randomized Algorithms & Advanced Data Structure

LI Bo
Department of Computing
The Hong Kong Polytechnic University

RANDOMIZED ALGORITHMS

RANDOM VARIABLES

RANDOM VARIABLES

- Let X be a discrete random variable.
- In particular, for every real number a, there is some value $\Pr[X = a]$ that says what is the total probability of all events where X takes value a. These values satisfy:

$$\Pr[X = a] \geq 0 \text{ and } \sum_a \Pr[X = a] = 1$$

- Saying that X is discrete means $\Pr[X = a] > 0$ for only finitely (or countably) many values a.

Definition (Expected Value, Expectation, Mean)

$$E[X] = \sum_a a \cdot \Pr[X = a]$$

Proposition (Probabilistic Method)

- There is some outcome such that X takes value $\geq E[X]$.
- There is some outcome such that X takes value $\leq E[X]$.

RANDOM VARIABLES

We can construct new random variables from old ones.

- For example, if X and Y are random variables then so to is $X + Y$.
- It is the random variable that takes the value of X plus the value of Y on each outcome.

Proposition (**Linearity of Expectation**)

For two random variables X and Y, over the same probability space, we have

$$E[X + Y] = E[X] + E[Y].$$

Furthermore, for a random variable X and a constant α we have

$$E[\alpha \cdot X] = \alpha \cdot E[X].$$

RANDOM VARIABLES

Definition

joint probability of $X = a$ and $Y = b$

X and Y are **independent** if for all values,

$$\Pr[X = a] \cdot \Pr[Y = b] = \Pr[X = a \text{ and } Y = b]$$

Proposition

If X and Y are independent, then $E[X \cdot Y] = E[X] \cdot E[Y]$.

More generally, if $X_1, \ldots, X_n$ are independent (meaning any two of them are independent) then

$$E[\Pi_i X_i] = \Pi_i E[X_i].$$

MARKOV'S INEQUALITY

Definition

Say that a random variable is nonnegative if $\Pr[X = a] > 0$ only for $a \geq 0$.

Theorem (Markov's Inequality)

- If X is a nonnegative random variable, then for any $\alpha > 0$ we have $\Pr\left[X \geq \alpha \cdot E[X]\right] \leq \frac{1}{\alpha}$.
- Equivalently, $\Pr[X \geq \alpha] \leq \frac{E[X]}{\alpha}$.

Proof.

by setting α to be $\alpha \cdot E[X]$.

The first statement follows immediately from the second statement.

$$E[X] = \sum_{a \geq \alpha} a \cdot \Pr[X = a] + \sum_{a < \alpha} a \cdot \Pr[X = a] \geq \sum_{a \geq \alpha} \alpha \cdot \Pr[X = a] = \alpha \cdot \Pr[X \geq \alpha]$$

UNION BOUND

- Sometimes we want to avoid a collection of bad events that may not be independent.
- The probability that some bad event happens is upper bounded by the sum of the individual probabilities of the bad events.

Theorem (Union Bound)

Consider any collection $X_1, X_2, \ldots, X_n$ of $\{0,1\}$ random variables. Then

$$\Pr[X_i = 1 \text{ for some } 1 \leq i \leq n] \leq \sum_{i=1,\ldots,n} \Pr[X_i = 1].$$

LAW OF TOTAL PROBABILITY

Theorem (Law of Total Probability)

If $\{B_1, \dots, B_n\}$ is a finite (or countably infinite) partition of a sample space (in other words, a set of pairwise disjoint events whose union is the entire sample space), then for any event A of the same probability space

joint probability:
the probability of A happens and B_i happens

$$\Pr[A] = \sum_{i=1}^{n} \Pr[A \cap B_i]$$

or, alternatively,

conditional probability:
the probability of A happens, given B_i happens

$$\Pr[A] = \sum_{i=1}^{n} \Pr[\,A \mid B_i\,] \cdot \Pr[B_i]\,.$$

WAITING FOR A FIRST SUCCESS

- We have a coin: come up head with probability $p > 0$, and tail $1 - p$.
- Different flips have independent outcomes.
- We flip the coin until we first get a head. What's the expected number of flips we perform?
- Let X be the random variable equal to the number of flips performed.
- For $j > 0$, we have $\Pr[X = j] = (1 - p)^{j-1}p$:

  The first $j - 1$ flips must come up tails, and the j-th must come up head. Thus

$$E[X] = \sum_{j=1}^{\infty} j \cdot \Pr[X = j] = \sum_{j=1}^{\infty} j \cdot (1 - p)^{j-1}p = \frac{1}{p}$$

Theorem (Waiting for a First Success)

If we repeatedly perform independent trials of an experiment, each of which succeeds with probability $p > 0$, then the expected number of trials we need to perform until the first success is $\frac{1}{p}$.

THE MAX-SAT PROBLEM

THE MAX-SAT PROBLEM

- In the maximum satisfiability problem (MAX-SAT), we are given clauses $C_1, \dots, C_m$, each a disjunction of literals over variables $x_1, \dots, x_n$ (e.g. $C_j = (x_1 \vee \overline{x_2} \vee x_3)$).
- Each of the variables x_i may be set to either true or false. The objective of the problem is to find a truth assignment that satisfies the maximum possible number of clauses.

The optimization version of SAT

Algorithm 1 (Flipping Coins)

Independently for each i, set $x_i = \begin{cases} true, \ with\ probability \frac{1}{2} \\ false, \ with\ probability \frac{1}{2} \end{cases}$

THE MAX-SAT PROBLEM

Lemma

For each clause C with, say, k literals,

$$\Pr[C \text{ is satisfied }] \geq 1 - \frac{1}{2^k}.$$

Proof: Instead of computing $\Pr[C \text{ is satisfied}]$, we compute $\Pr[C \text{ is not satisfied}]$.

$$\Pr[C \text{ is not satisfied}] = \Pi_{x_i \in C} \Pr[x_i \text{ is false}] \cdot \Pi_{\overline{x}_i \in C} \Pr[x_i \text{ is true}] = \left(\frac{1}{2}\right)^k$$

follows since x_i's are sampled independently.

Corollary

For MAX 3-SAT problem and any clause C,

$$\Pr[C \text{ is satisfied }] \geq \frac{7}{8}.$$

THE MAX-3SAT PROBLEM

α–approximation randomized algorithm:
$E[ALG] \geq \alpha \cdot OPT$

Theorem

For MAX 3-SAT problem, the expected number of satisfied clauses is at least $\frac{7}{8}m$, where m is the number of clauses.

$E[ALG1] \geq \frac{7}{8}m \geq \frac{7}{8}OPT$: Algorithm 1 is $\frac{7}{8}$–approximation

Proof:

$$E[\#\text{ satisfied clauses}] = \sum_C \Pr[C \text{ is satisfied}] * 1 \geq \frac{7}{8}m.$$

- For any random variable, there must be some point at which it assumes some value at least as large as its expectation.

Theorem

For every instance of 3-SAT, there is a truth assignment that satisfies at least a $\frac{7}{8}$ fraction of all clauses.

PROBABILISTIC METHOD

Theorem

For every instance of 3-SAT, there is a truth assignment that satisfies at least a $\frac{7}{8}$ fraction of all clauses.

- We have arrived at a nonobvious fact about 3-SAT:

  The existence of an assignment satisfying many clauses, whose statement has nothing to do with randomization; but we have done so by a randomized construction.

- This is a fairly widespread principle in the area of combinatorics:

  One can show the existence of some structure by showing that a random construction produces it with positive probability.

- Constructions of this sort are said to be applications of the **probabilistic method**.

THE MAX-3SAT PROBLEM

- Suppose we are not satisfied with a "one-shot" algorithm that produces a single assignment with a large number of satisfied clauses in expectation.
- Rather, we would like a randomized algorithm whose expected running time is polynomial and that is guaranteed to output a truth assignment satisfying at least a $\frac{7}{8}$ fraction of all clauses.
- A simple way to do this is to generate random truth assignments until one of them satisfies at least $\frac{7}{8}m$ clauses.
- How long will it take until we find one by random trials?

Waiting for a First Success?

THE MAX-3SAT PROBLEM

- If we can show that the probability a random assignment satisfies at least $\frac{7}{8}m$ clauses is at least p,

  then the expected number of trials performed by the algorithm is $\frac{1}{p}$.
- What is this quantity p?
- For $j = 1, \ldots, m$, let p_j denote the probability that a random assignment satisfies exactly j clauses.
- So the expected number of clauses satisfied, by the definition of expectation, is equal to

$$\sum_{j=1}^{m} j \cdot p_j$$

  By the previous analysis, this is equal to $\frac{7}{8}m$.

- We are interested in the quantity $p = \sum_{j \geq \frac{7}{8}m} p_j$.

THE MAX-3SAT PROBLEM

$$\frac{7}{8}m = \sum_{j=1}^{m} j \cdot p_j \quad = \sum_{j<\frac{7}{8}m} j \cdot p_j + \sum_{j\geq\frac{7}{8}m} j \cdot p_j$$

- If we can show that the probability a random assignment satisfies at least $\frac{7}{8}m$ clauses is at least p, then the expected number of trials performed by the algorithm is $\frac{1}{p}$.
- Let k' denote the largest natural number that is strictly smaller than $\frac{7}{8}m$.
- The right-hand side of the above equation only increases if we replace the terms in the first sum by $k'p_j$ and the terms in the second sum by mp_j. We have

$$\frac{7}{8}m = \sum_{j<\frac{7}{8}m} jp_j + \sum_{j\geq\frac{7}{8}m} jp_j \leq \sum_{j<\frac{7}{8}m} k'p_j + \sum_{j\geq\frac{7}{8}m} mp_j = k'(1-p) + mp \leq k' + mp$$

THE MAX-3SAT PROBLEM

$$\frac{7}{8}m \leq k' + mp \quad \Rightarrow \quad mp \geq \frac{7}{8}m - k'$$

- Since k' is a natural number strictly smaller than $\frac{7}{8}$ times another natural number,

$$\frac{7}{8}m - k' \geq \frac{1}{8}$$

- Thus,

$$p \geq \frac{\frac{7}{8}m - k'}{m} \geq \frac{1}{8m}.$$

- By the waiting-time bound, we see that the expected number of trials needed to find the satisfying assignment we want is at most $8m$.

Summary: Improvement of Algorithm 1 (Flipping Coins)

- Repeat Algorithm 1 until we research a correct solution: **Las Vegas Algorithms**
  - A Las Vegas algorithm is a randomized algorithm whose output is always correct.
  - The expected running time of the algorithm is polynomial.

- Repeat Algorithm 1 polynomial times: **Monte Carlo Algorithms**
  - A Monte Carlo algorithm is a randomized algorithm whose output may be incorrect with a certain (typically small) probability.
  - The running time of the algorithm is always polynomial.

DE-RANDOMIZATION

DE-RANDOMIZATION

Law of Total Probability:

$$\Pr[A] = \sum_{i=1}^{n} \Pr[\,A \mid B_i\,] \cdot \Pr[B_i]\,.$$

E[# satisfied clauses]

$$= E[\,\#\text{satisfied clauses} \mid x_1 = \mathit{true}\,] \cdot \Pr[x_1 = \mathit{true}] +$$

$$E[\,\#\text{satisfied clauses} \mid x_1 = \mathit{true}\,] \cdot \Pr[x_1 = \mathit{false}]$$

$$= \frac{1}{2}(E[\,\#\text{satisfied clauses} \mid x_1 = \mathit{true}\,] + E[\,\#\text{satisfied clauses} \mid x_1 = \mathit{false}\,])$$

$$\max\begin{cases} E[\,\#\text{satisfied clauses} \mid x_1 = \mathit{true}\,] \\ E[\,\#\text{satisfied clauses} \mid x_1 = \mathit{false}\,] \end{cases} \geq E[\#\text{ satisfied clauses}] \geq \frac{7}{8}m$$

If $E[\,\#\text{satisfied clauses} \mid x_1 = \mathit{true}\,] \geq E[\,\#\text{satisfied clauses} \mid x_1 = \mathit{false}\,]$, set $x_1 = \mathit{true}$;

If $E[\,\#\text{satisfied clauses} \mid x_1 = \mathit{true}\,] < E[\,\#\text{satisfied clauses} \mid x_1 = \mathit{false}\,]$, set $x_1 = \mathit{false}$

DE-RANDOMIZATION

Suppose we set $x_1 = b_1$

$$E[\,\#\text{satisfied clauses} \mid x_1 = b_1\,] \longrightarrow \geq E[\#\text{ satisfied clauses}] \geq \frac{7}{8}m$$

$$= E[\,\#\text{satisfied clauses} \mid x_1 = b_1, x_2 = true\,] \cdot \Pr[x_2 = true] +$$

$$E[\,\#\text{satisfied clauses} \mid x_1 = b_1, x_2 = true\,] \cdot \Pr[x_2 = false]$$

$$= \frac{1}{2}(E[\,\#\text{satisfied clauses} \mid x_1 = b_1 x_1 = true\,] +$$

$$E[\,\#\text{satisfied clauses} \mid x_1 = b_1 x_1 = false\,])$$

$$\max \begin{cases} E[\,\#\text{satisfied clauses} \mid x_1 = b_1, x_2 = true\,] \\ E[\,\#\text{satisfied clauses} \mid x_1 = b_1 x_1 = false\,] \end{cases} \geq E[\,\#\text{satisfied clauses} \mid x_1 = b_1\,]$$

DE-RANDOMIZATION

Suppose we set $x_1 = b_1$

$E[\,\#\text{satisfied clauses} \mid x_1 = b_1\,] \longrightarrow \geq E[\#\text{ satisfied clauses}] \geq \frac{7}{8}m$

$= E[\,\#\text{satisfied clauses} \mid x_1 = b_1, x_2 = true\,] \cdot \Pr[x_2 = true] +$

$E[\,\#\text{satisfied clauses} \mid x_1 = b_1, x_2 = true\,] \cdot \Pr[x_2 = false]$

$= \frac{1}{2}(E[\,\#\text{satisfied clauses} \mid x_1 = b_1 x_1 = true\,] +$

$E[\,\#\text{satisfied clauses} \mid x_1 = b_1 x_1 = false\,])$

If $E[\,\#\text{satisfied clauses} \mid x_1 = b_1, x_2 = true\,] \geq E[\,\#\text{satisfied clauses} \mid x_1 = b_1, x_2 = false\,]$,

set $x_2 = true$;

If $E[\,\#\text{satisfied clauses} \mid x_1 = b_1, x_2 = true\,] < E[\,\#\text{satisfied clauses} \mid x_1 = b_1, x_2 = false\,]$,

set $x_2 = false$.

DE-RANDOMIZATION

$$\max\begin{cases} E[\,\#\text{satisfied clauses} \mid x_1 = b_1, \dots, x_i = b_i, x_{i+1} = true\,] \\ E[\,\#\text{satisfied clauses} \mid x_1 = b_1, \dots, x_i = b_i, x_{i+1} = false\,] \end{cases} \geq E[\#\text{satisfied clauses}] \geq \frac{7}{8}m$$

Suppose we have set $x_1 = b_1, \dots, x_i = b_i$.

Case 1: If $E[\,\#\text{satisfied clauses} \mid x_1 = b_1, \dots, x_i = b_i, x_{i+1} = true\,] \geq$

$$E[\,\#\text{satisfied clauses} \mid x_1 = b_1, \dots, x_i = b_i, x_{i+1} = false\,],$$

set $x_{i+1} = true$;

Case 2: If $E[\,\#\text{satisfied clauses} \mid x_1 = b_1, \dots, x_i = b_i, x_{i+1} = true\,] <$

$$E[\,\#\text{satisfied clauses} \mid x_1 = b_1, \dots, x_i = b_i, x_{i+1} = false\,],$$

set $x_{i+1} = false$.

APPLICATION 1
ADVANCED DATA STRUCTURE

BLOOM FILTER

BLOOM FILTER

Bloom filter is a space-efficient probabilistic data structure, conceived by Burton Howard Bloom in 1970, that is used to test whether an element is a member of a set.

A typical application of Bloom Filters is **Web caching**

- An ISP may keep several levels of carefully located caches to speed up the loading of commonly viewed web pages, in particular large data objects such as images and videos.
- If a client requests a particular URL, then the service needs to determine quickly if the requested page is in one of its caches.
- If it turns out that a page thought to be in one of its caches is not there, it will be loaded from its native URL, and the penalty is not much worse than not having the cache in the first place.

False positives, while undesirable, are acceptable!

BLOOM FILTER

Bloom filter is a space-efficient probabilistic data structure, conceived by Burton Howard Bloom in 1970, that is used to test whether an element is a member of a set.

To maximize space efficiency, correctness is sacrificed:

- If a given key is not in the set, then a Bloom filter may give the wrong answer (this is called a **false positive**), but the probability of such a wrong answer can be made small.
- But **false negatives** are not possible: if a given key is in the set, then a Bloom filter must give the correct answer.
- Thus, a query returns either "possibly in set" or "definitely not in set".

BLOOM FILTER

A formal set-up:

- We want to represent n-element sets $S = \{s_1, \cdots, s_n\}$ from a very large universe U, with $|U| = u \gg |S| = n$.

  (Think of U as the set of URLs, n as the cache size, and S as the URLs of those web pages that are currently in the cache.)

- We want to support insertions and membership queries "Given $x \in U$, is $x \in S$?" so that:
  - If the answer is **No**, then $x \notin S$.
  - If the answer is **Yes**, then x may or may not be in S, but the probability that $x \notin S$ (false positive) is low.
- Bloom Filter can also be made to support deletions, but we won't worry about those in the current lecture.

BLOOM FILTER

- A Bloom filter is a bit vector B of m bits, with k independent hash functions $h_1, \cdots, h_k$ that map each key in U to the set $R_m = \{0,1, \cdots, m-1\}$.

- We assume that each hash function h_i maps a uniformly at random chosen key $x \in U$ to each element of R_m with equal probability.

- Since the hash functions are independent, it follows that the vector $(h_1(x), \cdots, h_k(x))$ is equally likely to be any of the m^k k-tuples of elements from R_m.

- Initially, all m bits of B are set to 0.
  - **Insert x into S**. Compute $h_1(x), \cdots, h_k(x)$ and set
  $$B[h_1(x)] = \cdots = B[h_k(x)] = 1.$$
  - **Query if $x \in S$**. Compute $h_1(x), \cdots, h_k(x)$. If
  $$B[h_1(x)] = \cdots = B[h_k(x)] = 1 \text{ then answer Yes, else answer No.}$$

- The running times depend only on the number k of hash functions.

BLOOM FILTER

An Example

- Let $m = 5$ (number of bits) and $k = 2$ (number of hash functions):
  - $h_1(x) = x \bmod 5$
  - $h_2(x) = (2x + 3) \bmod 5$
- We initialize the Bloom filter $B[0,1,2,3,4]$ and then insert 9 and 11:

	$h_1(x)$	$h_2(x)$	B				
Initialize:			0	0	0	0	0
Insert 9:	4	1	0	1	0	0	1
Insert 11:	1	0	1	1	0	0	1

- Now let us attempt some membership queries:

	$h_1(x)$	$h_2(x)$	Answer
Query 15:	0	3	No, not in B (correct answer)
Query 16:	1	0	Yes, in B (wrong answer: false positive)

Note that 16 was never inserted into the filter!

BLOOM FILTER

Analysis

- For simplicity, assume all hash functions are independent and perfectly random.
- The probability that one hash fails to set a given a bit is $1 - \frac{1}{m}$.
- Hence, after all n elements of S have been inserted into the Bloom filter, the probability that a specific bit is still 0 is

$$\left(1 - \frac{1}{m}\right)^{kn} \approx e^{-\frac{kn}{m}}$$

- The probability of a false positive is the probability that a specific set of k bits are 1, which is

$$\left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)^{k} \approx \left(1 - e^{-\frac{kn}{m}}\right)^{k} \approx (1-p)^k,$$

where $p = e^{-\frac{kn}{m}}$.

Task: **Suppose we are given the ratio $\frac{m}{n}$ and want to optimize the number k of hash functions to minimize the false positive rate f.**

BLOOM FILTER

Analysis

- The probability of a false positive is the probability that a specific set of k bits are 1, which is

$$f = \left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)^{k} \approx \left(1 - e^{-\frac{kn}{m}}\right)^{k} \approx (1-p)^{k},$$

where $p = e^{-\frac{kn}{m}}$.

- We can find the minimum by taking the derivative of f.
- To simplify the math, we minimize the logarithm of f with respect to k.

$$g = \ln f = k \cdot \ln(1-p) = k \cdot \ln\left(1 - e^{-\frac{kn}{m}}\right)$$

Thus

$$\frac{dg}{dk} = \ln\left(1 - e^{-\frac{kn}{m}}\right) + \frac{kn}{m} \cdot \frac{e^{-\frac{kn}{m}}}{1 - e^{-\frac{kn}{m}}}$$

- We find the optimal k, or right number of hash functions to use, when the derivative is 0.
- This occurs when $k = \ln 2 \cdot \frac{m}{n}$, and can be shown to be a global minimum.

BLOOM FILTER

Analysis

- For the optimal value of k, the false positive rate is

$$\left(\frac{1}{2}\right)^k = 0.6185^{\frac{m}{n}}.$$

- As m grows in proportion to n, the false positive rate decreases.
- For example, for $m = 8n$, ($k^* \approx 5.54$)
  - if $k = 3$, then $f = 0.0306$;
  - if $k = 4$, then $f = 0.0240$;
  - if $k = 5$, then $f = 0.0217$;
  - if $k = 6$, then $f = 0.0216$;
  - if $k = 7$, then $f = 0.0229$.

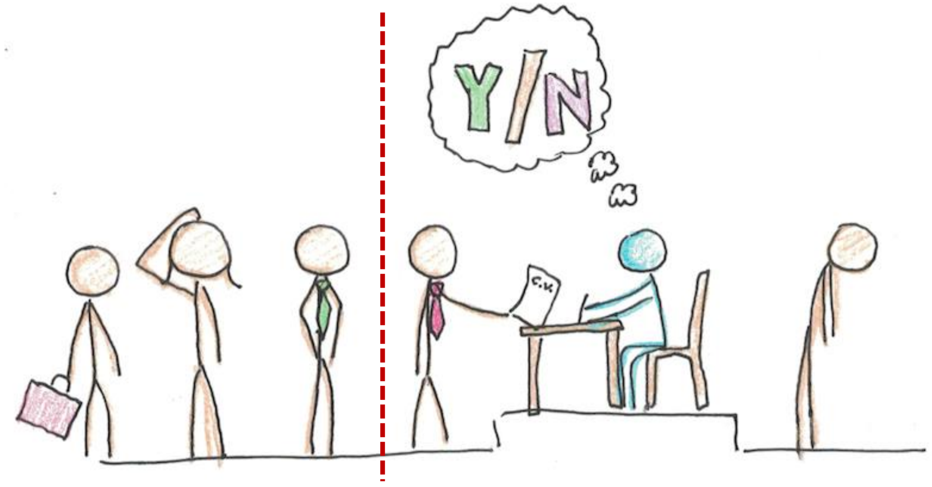
APPLICATION 2
ONLINE STOCHASTIC DECISION-MAKING

Secretary Problem

Secretary Problem

Problem

- Choosing one candidate out of n rankable candidates who arrive online.
- As long as one candidate arrives, we need to irrevocably decide whether to hire her/him.



If the decision can be deferred to the end, this can be solved by taking the maximum.
The difficulty is that the decision must be made immediately.

Optimal Stopping Algorithm

- The applicants are interviewed one by one in random order.
- A decision about each particular applicant is to be made immediately after the interview.
- Once rejected, an applicant cannot be recalled.
- During the interview, we gain information sufficient to rank the applicant among all applicants interviewed so far, but is unaware of the quality of yet unseen applicants.

Secretary Problem

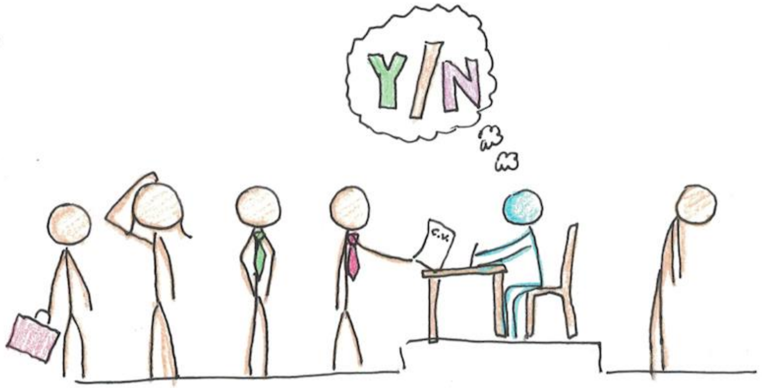
Problem:

- Choosing one candidate out of n rankable candidates who arrive online.
- As long as one candidate arrives, we need to irrevocably decide whether to hire her/him.

Setting:

- n different positive integers (the scores of candidates) that we do not know what they are
- n is known to the algorithm
- The numbers arrive in a uniformly random distribution.

$\Rightarrow$ Maximize the probability that the best candidate is hired.



Strategy I: Take the first candidate.

- P(finding best candidate) $= \frac{1}{n}$

Secretary Problem

Problem

- Choosing one candidate out of n rankable candidates who arrive online.
- As long as one candidate arrives, we need to irrevocably decide whether to hire her/him.

Setting:

- n different positive integers (the scores of candidates) that we do not know what they are
- n is known to the algorithm
- The numbers arrive in a uniformly random distribution.

Strategy II:

- Interview the first $\frac{n}{2}$ candidates, but do not select.
- Interview the second $\frac{n}{2}$ candidates, and select the first that is better than all the first $\frac{n}{2}$ candidates.

P(finding best candidate) $\geq$ P(second best candidate in first half) $\cdot$ P(best candidate in second half)

$$= \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}$$

Can you compute P(finding best candidate)?

Secretary Problem

Problem

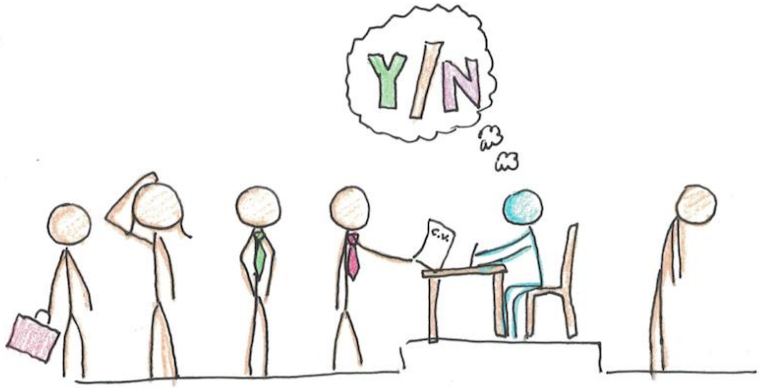
- Choosing one candidate out of n rankable candidates who arrive online.
- As long as one candidate arrives, we need to irrevocably decide whether to hire her/him.

Setting:

- n different positive integers (the scores of candidates) that we do not know what they are
- n is known to the algorithm
- The numbers arrive in a uniformly random distribution.

Strategy III (optimal):

- Interview the first $\frac{n}{e}$ candidates, but do not select.
- Interview the others, and select the first that is better than all the first $\frac{n}{e}$ candidates.
- P(finding best person) $= \frac{1}{e}$.



Note: The analysis on this page will not be examined.

Secretary Problem

Observation. Our algorithm should never pick an element that is not the best so far.

Observation. Given the current candidate i is the best so far,

$$f(i) = \Pr[i^{\text{th}} \text{ item is global best} \mid i^{th} \text{ is best so far}] = \frac{\Pr[ith \text{ item is global best}]}{\Pr[ith \text{ is best so far}]} = \frac{\frac{1}{n}}{\frac{1}{i}} = \frac{i}{n}.$$

Then $f(i)$ is increasing.

Observation. Letting $g(i) = Pr[\text{picking global best using optimal strategy from item } i \text{ onwards}]$,

then $g(i)$ is a non-increasing function (i.e., $g(i) \geq g(i+1)$).

Proof. Otherwise, we could use the strategy for $g(i+1)$ also for $g(i)$ by just ignoring item i.

So any optimal strategy should

- pass at times i where $f(i) < g(i+1)$ and
- pick at other times if we see an element that is best so far.

Note: The analysis on this page will not be examined.

Secretary Problem

So any optimal strategy should

- pass at times i where $f(i) < g(i+1)$ and
- pick at other times if we see an element that is best so far.

→ A strategy similar to the "1/4" is optimal (with a different threshold)!

- If we reject the first τ items, then the probability we succeed in picking the global best is

$$\sum_{t=\tau+1}^{n} \Pr[t^{\text{th}} \text{ item is global best}] \cdot \Pr[\text{ best of first } t - 1 \text{ items is in first } \tau \text{ positions}]$$

$$= \sum_{t=\tau+1}^{n} \frac{1}{n} \cdot \frac{\tau}{t-1} = \frac{\tau}{n}(H_n - H_{\tau-1}) = g(\tau+1).$$

$$H_n = 1 + \frac{1}{2} + \cdots + \frac{1}{n} \to \ln n$$

- So the first position where $f(\tau) = \frac{\tau}{n} \geq g(\tau+1)$ is given by the smallest τ such that $H_n - H_{\tau-1} \leq 1$, i.e., $\tau \approx n/e$ for large n, and hence $g(\tau+1) \approx 1/e$.

Prophet Inequality

Prophet Inequality

- There are n random variables $X_1, \cdots, X_n$.
- We know their distributions upfront, but not their realizations.
- These realizations are revealed one-by-one (say in the order $1, \cdots, n$).
- We want to give a strategy (a stopping rule) that, upon seeing the value X_i (and all the values before it) decides either to choose i, in which case we get reward X_i and the process stops.
- Or we can pass, in which case we move on to the next item, and are not allowed to come back to i.
- We want to maximize our **expected reward**.

$$X_{max} = \max\{X_1, \cdots, X_n\}$$

The prophet knows everything.

Expected Profit = $E[X_{max}]$

Our expected reward cannot exceed $E[X_{max}]$.

But how close can we get?

Prophet Inequality

$X_{max} = \max\{X_1, \cdots, X_n\}$
Benchmark = $E[X_{max}]$

A Hard Example:
- $X_1 = 1$ w.p. 1
- $X_2 = \frac{1}{\epsilon}$ w.p. ϵ, and $X_2 = 0$ w.p. $1 - \epsilon$
- $E[X_{max}] = 2 - \epsilon$

What can we do?
- Accept the first item: profit of 1 w.p. 1
- Pass the first item and accept the second:

$$E[X_2] = 1$$

Observation:
No stopping rule can achieve better than 1/2-approximation.

Theorem:
There is a strategy with expected reward $\frac{1}{2}E[X_{max}]$.

There are several strategies achieves 0.5-approximation.

A Simple Strategy
- Consider the distribution of X_{max}: $\Pr[X_{max} \geq c] = 1 - \Pi_i \Pr[X_i < c]$ for all c.
- Let τ be the median of the distribution of X_{max}: $\Pr[X_{max} \geq \tau] = 0.5$.
- Assume there is no point mass at τ.
- **Strategy**: pick the first X_i which exceeds τ.

$X_{max} = \max\{X_1, \cdots, X_n\}$
Benchmark $= E[X_{max}]$

Note: The analysis on this page will not be examined.

Prophet Inequality

Theorem:

There is a strategy with expected reward $\frac{1}{2}E[X_{max}]$.

A Simple Strategy

- Consider the distribution of X_{max}: $\Pr[X_{max} \geq c] = 1 - \Pi_i \Pr[X_i < c]$ for all c.
- Let τ be the median of the distribution of X_{max}: $\Pr[X_{max} \geq \tau] = 0.5$.
- Assume there is no point mass at τ.
- **Strategy**: pick the first X_i which exceeds τ.

An imaginary experiment:

If $X_{max} < \tau$, the profit is rounded up to τ.
If $X_{max} \geq \tau$, the profit is X_{max}.

$$E[X_{max}] \leq \tau + E[(X_{max} - \tau)^+])$$
$$\leq \tau + \sum_{i=1}^{n} E[(X_i - \tau)^+])$$

$$ALG \geq \tau \cdot \Pr[X_{max} \geq \tau] + \sum_{i=1}^{n} \Pr\left[\bigwedge_{j<i} x_j < \tau\right] \cdot E[(X_i - \tau)^+]$$
$$\geq \tau \cdot \Pr[X_{max} \geq \tau] + \sum_{i=1}^{n} \Pr[X_{max} < \tau] \cdot E[(X_i - \tau)^+]$$
$$= \Pr[X_{max} < \tau] \cdot \left(\tau + \sum_{i=1}^{n} E[(X_i - \tau)^+]\right) \geq \frac{1}{2}E[X_{max}] \blacksquare$$

Thank You!