Randomized Algorithms

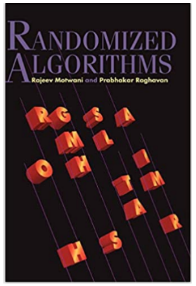
Introduction

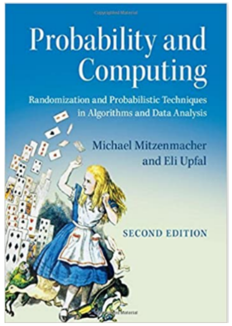
Joseph Chuang-Chieh Lin

Dept. CSE,
National Taiwan Ocean University

Textbooks and Materials

- Textbooks:
  - *Randomized Algorithms*. Motwani, R. and Raghavan, P., 1995. Cambridge University Press.
  - Probability and Computing: Randomization and Probabilistic Techniques in Algorithms and Data Analysis. Second Edition. M. Mitzenmacher and E. Upfal, Cambridge University Press, 2017.




Probability and Computing
Randomization and Probabilistic Techniques in Algorithms and Data Analysis
Michael Mitzenmacher and Eli Upfal
SECOND EDITION

- Other materials:
  - Prepared slides.

Prerequisites

- Basic undergraduate courses in
  - Algorithms
  - Data structures
  - Probability theory
  - Discrete mathematics
  - Linear Algebra
- Motivation.
- Curiosity.

Topics

- Introduction & the Min-Cut Problem
- Examples of Probability Paradoxes
- Probability Prerequisites
- Coupon Collector’s Problem
- The Secretary Problem
- Randomized Quicksort
- Moments and Deviations
- Chernoff Bounds and Hoeffding Bounds
- Balls and Bins
- Continuous Distributions and the Poisson Process
- Markov Chains and Random Walks
- Monte Chains Monte Carlo (MCMC)

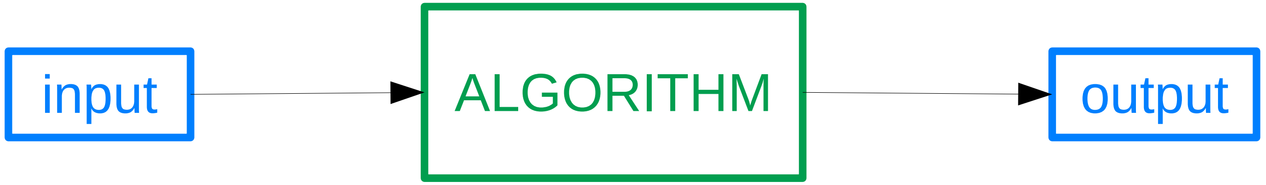
Grading Policy

- Attendance (10%)
- Assignments and Quizzes (60%)
- Final Presentation (30%)

Grading Policy for the Presentations

- Complete the presentation: 60 point
  - Duration for each presentation: 30~50 minutes.
- Raising questions: +4 point for each one (maximum +20 point)
- Clearly answering the teacher's 4 questions: +5 point for each one.

Traditional deterministic algorithms



Randomized algorithms

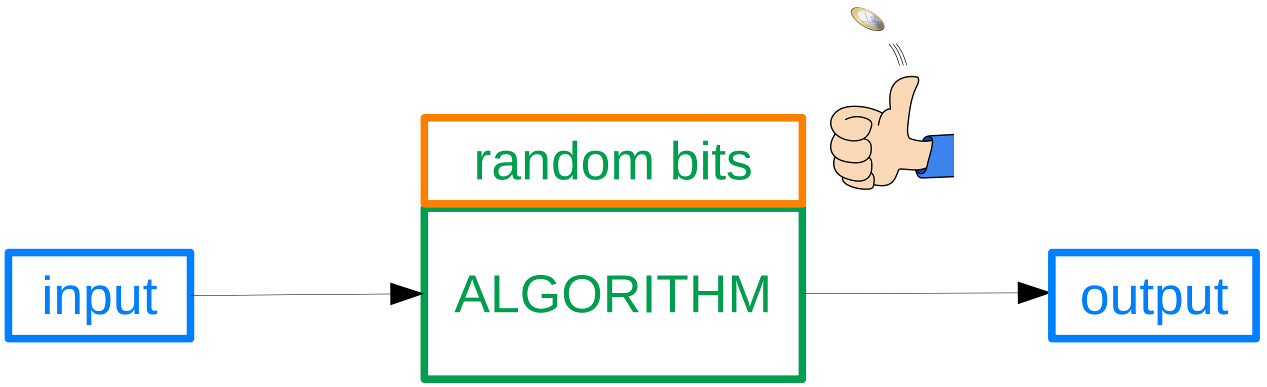


figure freely downloaded from https://pixabay.com/illustrations/coin-flipping-coin-hand-flip-flick-5822271/

Why?

- Randomized algorithms are
  - often much ***simpler*** than the best known deterministic ones.
  - often much ***more efficient*** (faster or using less space) than the best known deterministic ones.

- Sometimes ideas from the randomized algorithms lead to good deterministic algorithms.

Comparisons

- It’s different from the *average-case* analysis of deterministic algorithms.
  - e.g., expected running time of a deterministic algorithm on input sampled from a distribution.

- In most scenarios, it’s NOT a heuristic algorithm.
  - The accuracy is guaranteed, or
  - The running time is guaranteed.

An illustrating example:

- **Problem:** find a grade-‘*A*’ student in a class of n students where half of them get ‘A’.

- What is the time complexity for the best deterministic algorithm?
  - I mean, in the “worst case”.

A randomized algorithm (from Wikipedia)

```
findingA_LV(array $L$, $n$)
begin
    repeat
        Randomly select one element out of $n$ elements.
    until '$A$' is found
end
```

Assignment: Prove that the expected number of iterations is $\lim_{n\to\infty} \sum_{i=1}^{n} \frac{i}{2^i} \le 2.$

A randomized algorithm (from Wikipedia)

```
findingA_MC(array $L$, $n$, $k$)
begin
    $i \leftarrow 0$
    repeat
        Randomly select one element out of $n$ elements.
        $i \leftarrow i + 1$
    until $i = k$ or '$A$' is found
end
```

After k iterations, $\Pr[\text{find } A] = 1 - (1/2)^k$.

Birthday problem (paradox)

- There are n randomly chosen people in a room.
- How ***large*** should n be such that there is at least one pair of them having the same birthday (mm/dd)?
- By the pigeonhole principle, $n = 367$? or 366?

- Let us consider this problem in the other way around.

  How ***large*** should n be such that there is at least one pair of them having the same birthday (mm/dd) with probability ≥ 0.5?

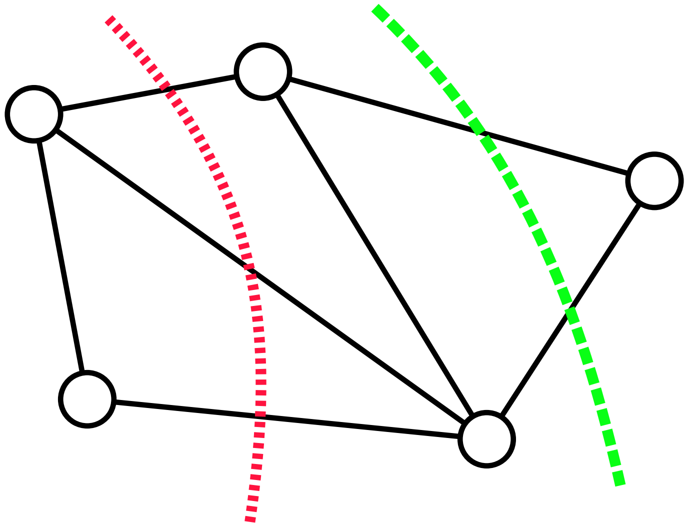
Birthday problem (paradox)

- n people: $x_1, x_2, \ldots, x_n$
- Event i: some pair of $x_1, x_2, \ldots, x_i$ have the same birthday.
- $\Pr[\text{Event2}] = 1 - \frac{364}{365}$
- $\Pr[\text{Event3}] = 1 - \frac{364}{365} \cdot \frac{363}{365}$
- ...
- $\Pr[\text{Event23}] = 1 - \frac{364}{365} \cdot \frac{363}{365} \cdots \frac{343}{365} \approx 0.507297.$
- 23 is much less than 366 or 367.

Birthday problem (paradox)

- **Assignment:** Compute n such that there is at least one pair of them having the same birthday with probability ≥ 0.9.

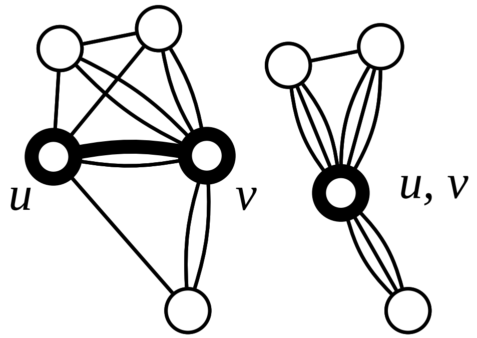
Min-Cut



$|V| = n, |E| = m$

- A graph $G = (V, E)$ and its two "cuts".
  - **Cut**: a partition of the vertices in V into two non-empty, disjoint sets S and T such that
  - $S \cup T = V$
- The **cutset** of a cut:
  - $\{uv \in E \mid u \in S, v \in T\}.$
- The size of the cut:
  - the cardinality of its cutset.

Edge contraction



$e = (u, v)$

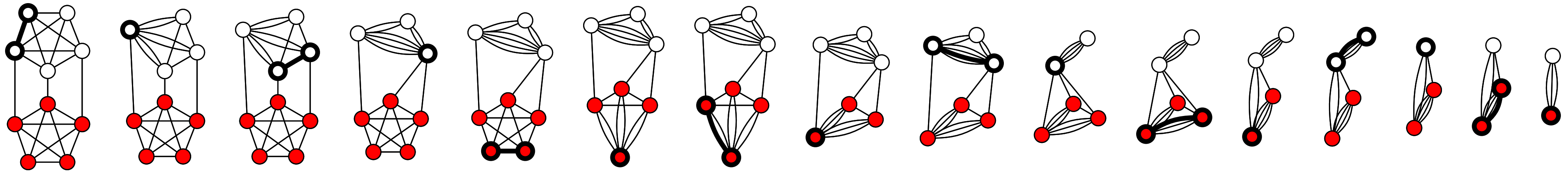
$G \rightarrow G/e$

By Thore Husfeldt - Python, networkx, neato and TikZ, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=21117500

Karger’s edge-contraction algorithm (1993)

```
Procedure contract ($G = (V, E)$):
while $|V| > 2$:
    choose $e \in E$ uniformly at random
    $G \leftarrow G/e$
return the only cut in $G$
```

Time complexity: $O(m)$ or $O(n^2)$.



By Thore Husfeldt - Created in python using the networkx library for graph manipulation, neato for layout, and TikZ for drawing., CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=21103489

Analysis

- C: a specific cut of G.
- k: the size of the cut C.
- The minimum degree of G must be $\geq k$. (WHY?)
  - So, $|E| \geq nk/2$.
- The probability that the algorithm picks an edge from C to contract is

$$\frac{k}{|E|} \leq \frac{k}{nk/2} = \frac{2}{n}.$$

Analysis (contd.)

- Let p_n be the probability that the algorithm on an *n*-vertex graph avoids *C*.
- Then,

$$p_n \geq \left(1 - \frac{2}{n}\right) \cdot p_{n-1}$$

- The recurrence can be expanded as

$$p_n \geq \prod_{i=0}^{n-3} \left(1 - \frac{2}{n-i}\right) = \frac{n-2}{n} \cdot \frac{n-3}{n-1} \cdots \frac{2}{4} \cdot \frac{1}{3} = \frac{1}{\binom{n}{2}}.$$

Analysis (contd.)

- The probability of "success" is $\binom{n}{2}^{-1}$
  - A bit too low, isn't it?

- How about repeating it for $T = \binom{n}{2} \ln n$ times, and then choose the minimum of them?

- The probability of NOT finding a min-cut is $\left[1 - \binom{n}{2}^{-1}\right]^T \le \frac{1}{e^{\ln n}} = \frac{1}{n}.$

- Total running time: $O(Tm)$ or $O(Tn^2)$.

A bonus project (5%)

- Implement Karger’s edge-contraction algorithm.

Assignment 1 (5%)

A randomized algorithm (from Wikipedia) (3%)

```
findingA_LV(array $L$, $n$)
begin
   repeat
      Randomly select one element out of $n$ elements.
   until '$A$' is found
end
```

Assignment: Prove that the expected number of iterations is $\lim_{n\to\infty}\sum_{i=1}^{n}\frac{i}{2^i} \le 2.$

Birthday problem (paradox) (2%)

- **Assignment:** Compute n such that there is at least one pair of them having the same birthday with probability ≥ 0.9.