Tutorials: NDMI025 - Randomized Algorithms

Karel Král

March 11, 2021

This text is a work in progress, do not distribute. All errors in this text are on purpose. Please report them to my email kralka@iuuk.mff.cuni....

Contents

Chapter 1

Exercises

1.1 Tutorial

1. - Can you all hear me?
   - If you are uncomfortable asking a question in English, just ask in Czech/Slovak and I will translate.
   - Have you all taken:
     (a) a probability course (discrete probability, random variables, expected value, variance, Markov, Chernoff)
     (b) a linear algebra course (matrix operations, linear maps, eigenvectors and eivenvalues, discriminant)
     (c) a graph theory course (what a combinatorial graph is, bipartite, complete, coloring)
     (d) a combinatorics course (factorial, binomial coefficients)
     (e) an algorithms / programming course (big-O notation, possibly understanding Python based on the other question)
   - This class is heavy on theory. Are you interested in computer simulations and or implementations? If so:
     (a) Python
     (b) R
     (c) C++

   Solution: 1

2. You are presented with two sealed envelopes. There are k\$ in one of those and ℓ\$ in the other ($k, \ell \in \mathbb{N}$ but you do not know k, ℓ in advance). You may open an envelope and (based on what you see) decide to take this one or the other (without looking into both).
   (a) Is there a way how to walk away with the larger amoung of money with probability strictly larger than 0.5?
   (b) What is the expected value you walk away with (in terms of k, ℓ)?
   (c) Simulate.

   Solution: 2

3. Graph isomorphism. You have seen an interactive proof of graph non-isomorphism on the class. Can you come up with an interactive proof of graph isomorphism?

   Solution: 3

4. We will focus on random walks and their properties a lot.

   (a) Random walks are useful when analysing algorithms – "two coloring without monochromatic triangle" of three-colorable graph.

   (b) Random numbers in the computer are often expensive to generate, can we reduce number of used random bits (expanders)? Or even get a deterministic algorithm?

   (c) To sample from extremely large spaces.

   Let $n \in \mathbb{N}$, say $n = 30$. Let us the following problem we start with $X_0 = \lfloor n/2 \rfloor$ and do the following process:

   - if $X_i \in \{0, n\}$ we stop
   - we set $X_{i+1} = X_i + \delta$ where δ is picked uniformly at random from $\{-1, 1\}$

   (a) Is this a Markov chain (Definition 2.2)? If so can you write it's matrix?

   (b) What is the expected number of steps until stopping?

   Solution: 4

5. Think of some example MCs.

   (a) Create a MC that is irreducible.

   (b) Create a MC that is not irreducible.

   (c) Create a MC that is periodic.

   (d) Create a MC that is not periodic.

   (e) Compute a stationary distribution of the following MC:

   $$\begin{pmatrix} 1/2 & 1/2 \\ 1/2 & 1/2 \end{pmatrix}$$

   (f) Create a MC that has more stationary distributions.

   Solution: 5

6. We are collectors and we want to collect all n kinds of coupons. Coupons are sold in packages which all look the same. Thus when we buy an coupon, we buy one of n kinds uniformly at random. This is known as the *coupon collector* problem.

   (a) What is the expected number of coupons we need to buy to get all kinds?

   (b) How many coupons do we need to buy to have probability at least $1 - q$ of collecting all kinds?

   (c) What is the Markov chain? Is this similar to a random walk on some graph?

   (d) Simulate.

   Solution: 6

Chapter 2

Theory

2.1 Probability 101

Probability 101

2.2 Markov Chain

Definition. *A* discrete-time Markov chain *is a sequence of random variables* $X_0, X_1, X_2, \ldots$ *with the Markov property:*

$$\Pr[X_{n+1} = x \mid X_0 = x_0, X_1 = x_1, \ldots, X_n = x_n] = \Pr[X_{n+1} = x \mid X_n = x_n]$$
$$\text{(if both are defined, i.e., } \Pr[X_0 = x_0, X_1 = x_1, \ldots, X_n = x_n] > 0)$$

and the possible values of X_i *form a countable set called the state space of the Markov chain.*

The Markov property states that the process has no memory – the next state depends only on the current state. We will deal with a special case where the state space of each random variable will be the same and finite. Moreover we will deal with time-homogenous Markov chains, that is $\Pr[X_{n+1} \mid X_n] = \Pr[X_n \mid X_{n-1}]$ (the transition probabilities are time independent). Thus we will represent Markov chains by their transition matrices – if a Markov chain has n states its transition matrix is $P \in [0,1]^{n \times n}$ such that $P_{i,j} = \Pr[X_{n+1} = i \mid X_n = j]$ (thus column sums are equal to one).

If we take a probability distribution $\pi \in [0,1]^n$ and multiply it by the transition matrix we get the probability distribution after one step $P\pi$.

There are several interesting properties of Markov chains:

- We say that a MC is *irreducible* iff for each pair of states $i, j \in [n]$ there is a time $k \in \mathbb{N}$ such that $(P^k)_{i,j} > 0$ (we can get from any state to any state).
- We say that a MC is *periodic* iff there is a state $i \in [n]$ and a period $p \in \mathbb{N}, p > 1$ such that for each time $k \in \mathbb{N}$ we have $(P^k)_{i,i} > 0 \Rightarrow p \mid k$ that is probability of staying at state i is positive only for multiples of the period.
- We say that $\pi \in [0,1]^n$ is a *stationary distribution* of a given MC iff $P\pi = \pi$ (the distribution is the same after one step).

Theorem 1. *If a MC is aperiodic and irreducible it has a unique stationary distribution* π*. Moreover for all pairs of states* $i, j \in [n]$ *the we know that*

$$\lim_{t \to \infty} (P^t)_{i,j} = \pi_i$$

Chapter 3

Solutions

3.1 Tutorial

1.

- **Can you all hear me?**
- **If you are uncomfortable asking a question in English, just ask in Czech/Slovak and I will translate.**
- **Have you all taken:**
  - (a) **a probability course (discrete probability, random variables, expected value, variance, Markov, Chernoff)**
  - (b) **a linear algebra course (matrix operations, linear maps, eigenvectors and eivenvalues, discriminant)**
  - (c) **a graph theory course (what a combinatorial graph is, bipartite, complete, coloring)**
  - (d) **a combinatorics course (factorial, binomial coefficients)**
  - (e) **an algorithms / programming course (big-O notation, possibly understanding Python based on the other question)**
- **This class is heavy on theory. Are you interested in computer simulations and or implementations? If so:**
  - (a) **Python**
  - (b) **R**
  - (c) **C++**

2. **You are presented with two sealed envelopes. There are k\$ in one of those and ℓ\$ in the other ($k, \ell \in \mathbb{N}$ but you do not know k, ℓ in advance). You may open an envelope and (based on what you see) decide to take this one or the other (without looking into both).**

   (a) Is there a way how to walk away with the larger amoung of money with probability strictly larger than 0.5?

   Solution: Pick an envelope uniformly at random. If you see m\$ toss a fair coin until you get Tails. If the number of tosses was strictly less than m keep the envelope, otherwise take the other. If $k < \ell$ then the probability of keeping the envelope with k\$ is strictly less than the probability of keeping the envelope with ℓ\$.

   (b) What is the expected value you walk away with (in terms of k, ℓ)?

   Solution: Let us recall the sum of geometric series:

$$\begin{aligned} S &= \sum_{j=0}^{n} q^j \\ &= 1 + q + q^2 + \ldots + q^n \\ &= 1 + q\left(1 + q + q^2 + \ldots + q^{n-1}\right) \\ &= 1 + q(S - q^n) \end{aligned}$$

   thus

$$\begin{aligned} S &= 1 + q(S - q^n) \\ S - qS &= 1 - q^{n+1} \\ S &= \frac{1 - q^{n+1}}{1 - q} \qquad \text{(pokud } q \neq 1\text{)} \end{aligned}$$

   and for the infinite case:

$$\begin{aligned} \sum_{j=0}^{\infty} q^j &= \lim_{n\to\infty} \sum_{j=0}^{n} q^j \\ &= \lim_{n\to\infty} \frac{1 - q^{n+1}}{1 - q} \\ &= \frac{1}{1 - q} \qquad \text{(pokud } |q| < 1\text{)} \end{aligned}$$

   Thus exactly n tosses have probability for the general case where Tails has probability p and Heads has probability $1 - p$:

$$\Pr[n \text{ tosses}] = (1-p)^{n-1}p \qquad \text{(for any } n \in \mathbb{N}^+\text{)}$$

   Probability of at most n tosses:

$$\begin{aligned} \Pr[1, 2, \ldots, n \text{ tosses}] &= \sum_{j=1}^{n} p(1-p)^{j-1} \\ &= p\sum_{j=1}^{n} (1-p)^{j-1} \\ &= p\frac{1 - (1-p)^n}{1 - (1-p)} \end{aligned}$$

$$= 1 - (1-p)^n$$

Probability that we keep k\$ (fair coin):

$$\Pr[\text{tosses } < k] = \sum_{j=1}^{k-1} 0.5^j$$
$$= 1 - 0.5^{k-1}$$

Thus probability of walking away with k\$ is

$$\Pr[\text{winning } k\$] = \frac{1}{2}(1 - 0.5^{k-1}) + \frac{1}{2}0.5^{\ell-1}$$
$$= \frac{1}{2} - 0.5^k + 0.5^\ell$$
$$= \frac{1}{2} + (0.5^\ell - 0.5^k)$$

Thus the expected win is

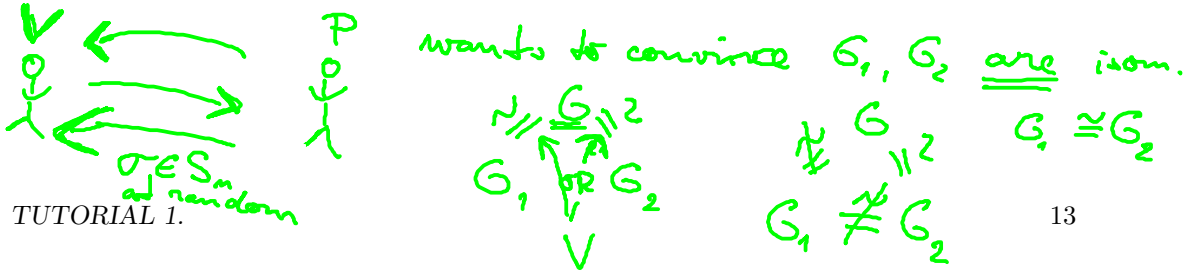
$$\mathbb{E}[\text{win}] = k\left(\frac{1}{2} + (0.5^\ell - 0.5^k)\right) + \ell\left(\frac{1}{2} + (0.5^k - 0.5^\ell)\right)$$

(c) Simulate.

Solution:

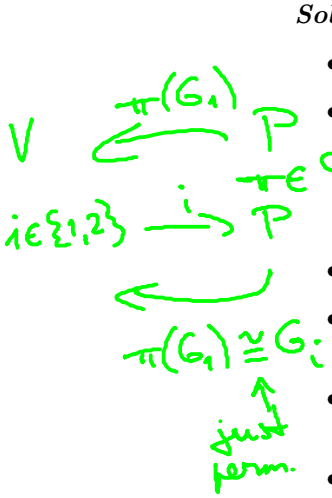
```
# https://docs.python.org/3/library/random.html
# Do not use for cryptography!
from random import randint
from random import random

def geometric(pr: float = 0.5) -> int:
    """pr is success probability, return the number of tosses until
    the first success."""
    assert pr > 0
    sample = 1
    fail_pr = 1 - pr
    while random() < fail_pr:
        sample += 1
    return sample


# Our unknown amounts.
envelopes = [5, 10]

N = 1000000       # Number of samples.
total_amount = 0  # Total sum that we got during all samples.
got_larger = 0    # Number of times we walked away with the larger sum.

for _ in range(N):
    # Pick the first envelope at random.
    chosen = randint(0, 1)
```

```
    if geometric() < envelopes[chosen]:
        # Keep this one.
        pass
    else:
        # Choose the other.
        chosen = 1 - chosen
    if envelopes[chosen] >= envelopes[1 - chosen]:
        got_larger += 1
    total_amount += envelopes[chosen]

k = envelopes[0]
l = envelopes[1]
pr_larger = 0.5 + abs(0.5**k - 0.5**l)
e_win = k * (0.5 + (0.5**l - 0.5**k)) + l * (0.5 + (0.5**k - 0.5**l))

print(f'Pr[selected larger] = {got_larger / N} (={pr_larger})')
print(f'E[win] = {total_amount / N} (={e_win})')

# Possible outcome:
# Pr[selected larger] = 0.529865 (=0.5302734375)
# E[win] = 7.649325 (=7.6513671875)
```



3. **Graph isomorphism. You have seen an interactive proof of graph non-isomorphism on the class. Can you come up with an interactive proof of graph isomorphism?**

Solution:



- Both the prover P and the verifier V know two graphs G_1, G_2.
- The prover knows an isomorphism π such that $\pi(G_1) = G_2$. Formally $\pi\colon V(G_1) \to V(G_2)$ such that
$$(u, v) \in E(G_1) \Leftrightarrow (\pi(u), \pi(v)) \in E(G_2).$$
And by $\pi(G_1)$ we mean the graph $(\pi(V(G_1)), \{(\pi(u), \pi(v)) \mid (u, v) \in E(G_1)\})$.
- For ease of presentation we set $V(G_1) = V(G_2) = [n] = \{1, 2, 3, \ldots, n\}$.
- The prover picks uniformly random permutation $\sigma \in S_n$ and sends the graph $G = \sigma(G_1)$.
- The verifier picks uniformly random number $i \in \{1, 2\}$ and asks verifier to present a permutation τ such that $\tau(G) = G_i$.
- If $i = 1$ then the prover sends $\tau = \sigma^{-1}$. If $i = 2$ then the prover sends $\tau = (\sigma \circ \pi)^{-1}$.

This is indeed an interactive proof:

- If the prover knows the isomorphism π, then all answers are correct.
- If G_1, G_2 are not isomorphic, then the verifier will pick a graph (either G_1 or G_2) which is not isomorphic with G with probability $1/2$.

Again the prover learns nothing about the isomorphism. If you find these interactive proofs interesting, take a look at "Zero Knowledge Proofs".

Also note that our prover can be implemented efficiently as opposed to the case of graph non-isomorphism. In fact in some sense the prover proves that it knows the isomorphism (this can be made formal, see "Zero Knowledge Proofs of Knowledge").

It is natural to repeat this protocol more times in order to boost the probabilities. This is called probability amplification. We will investigate this much more during the semester.

4. **We will focus on random walks and their properties a lot.**

   (a) **Random walks are useful when analysing algorithms – "two coloring without monochromatic triangle" of three-colorable graph.**

   (b) **Random numbers in the computer are often expensive to generate, can we reduce number of used random bits (expanders)? Or even get a deterministic algorithm?**

   (c) **To sample from extremely large spaces.**

   Let $n \in \mathbb{N}$, say $n = 30$. Let us the following problem we start with $X_0 = \lfloor n/2 \rfloor$ and do the following process:

   - **if $X_i \in \{0, n\}$ we stop**
   - **we set $X_{i+1} = X_i + \delta$ where δ is picked uniformly at random from $\{-1, 1\}$**

   (a) **Is this a Markov chain (Definition 2.2)? If so can you write it's matrix?**

   Solution: Yes (see the lecture video).

   (b) **What is the expected number of steps until stopping?**

   Solution: Let us set

   $$S_k = \mathbb{E}[\text{number of steps untill stopping, when starting at } k]$$

   We know the following:

   $$S_0 = S_n = 0$$
   $$S_k = 1 + \frac{1}{2}(S_{k-1} + S_{k+1}) \quad \text{(by linearity of expectation)}$$

   The above is so-called difference equation. It is not terribly complicated, but not super easy to solve (hint try to consider equations for $d(k) = S_k - S_{k-1}$ to get rid of the "1+" term). You may look at `https://en.wikipedia.org/wiki/Recurrence_relation` Luckily when dealing with asymptotics thus we do not need exact estimates. And you will see some nice theoretical results tomorrow.

   But it can be shown that

   $$S_k = k(n-k),$$

   which we can easily check that this is indeed a solution (note that we would also need that this is a unique solution, see solution methods on Wikipedia for this part):

   $$S_k = 1 + \frac{1}{2}(S_{k-1} + S_{k+1})$$
   $$S_k = 1 + \frac{1}{2}((k-1)(n-(k-1)) + (k+1)(n-(k+1)))$$
   $$S_k = 1 + \frac{1}{2}((k-1)n - (k-1)^2 + (k+1)n - (k+1)^2)$$
   $$S_k = 1 + \frac{1}{2}(2kn - (k-1)^2 - (k+1)^2)$$
   $$S_k = 1 + \frac{1}{2}(2kn - 2k^2 - 2)$$
   $$S_k = k(n-k)$$

5. **Think of some example MCs.**

   (a) **Create a MC that is irreducible.**

   Solution: Two states:

$$\begin{pmatrix} 1/2 & 1/2 \\ 1/2 & 1/2 \end{pmatrix}$$

   (with probability 1/2 stay at the current state, with probability 1/2 switch to the other state).

   (b) **Create a MC that is not irreducible.**

   Solution: Two states:

$$\begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}$$

   (always stay at the first state or immediatelly go there).

   (c) **Create a MC that is periodic.**

   Solution: Three states:

$$\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

   (from the first state go always to the third, from the second always to the first and from the third always to the second).

   (d) **Create a MC that is not periodic.**

   Solution: Two states:

$$\begin{pmatrix} 1/2 & 1/2 \\ 1/2 & 1/2 \end{pmatrix}$$

   (with probability 1/2 stay at the current state, with probability 1/2 switch to the other state).

   (e) **Compute a stationary distribution of the following MC:**

$$\begin{pmatrix} 1/2 & 1/2 \\ 1/2 & 1/2 \end{pmatrix}$$

   Solution: One eigenvalue is 1, the only stationary distribution $(1/2, 1/2)^T$. The other eigenvalue is 0 with the corresponding eigenvector $(1, -1)^T$ (this is not a distribution).

   (f) **Create a MC that has more stationary distributions.**

   Solution: Two states:

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

   (always stay where we are).

6. **We are collectors and we want to collect all n kinds of coupons. Coupons are sold in packages which all look the same. Thus when we buy an coupon, we buy one of n kinds uniformly at random. This is known as the *coupon collector* problem.**

   (a) **What is the expected number of coupons we need to buy to get all kinds?**

   Solution: Let t_i be the time to collect the i-th coupon kind after we have collected $i-1$ coupons. The probability of buying the i-th coupon is

   $$\Pr[\text{getting } i\text{-th coupon when already having } i-1 \text{ coupons}] = \frac{n-(i-1)}{n}$$

   Thus t_i has geometric distribution (we are tossing the same probability and waiting for the first success). The expected value of t_i is:

   $$\mathbb{E}[t_i] = \frac{n}{n-(i-1)}$$

   By linearity of expectation:

   $$\begin{aligned}
   \mathbb{E}[\text{collecting}] &= \mathbb{E}[t_1 + t_2 + \ldots + t_n] \\
   &= \mathbb{E}[t_1] + \mathbb{E}[t_2] + \ldots + \mathbb{E}[t_n] \\
   &= \frac{n}{n} + \frac{n}{n-1} + \frac{n}{n-2} + \ldots + \frac{n}{n-(n-1)} \\
   &= nH_n \\
   &= n\log(n) + n \cdot 0.577\ldots + 1/2 + \mathcal{O}(1/n) \qquad \text{(source Wikipedia)}
   \end{aligned}$$

   (b) **How many coupons do we need to buy to have probability at least $1-q$ of collecting all kinds?**

   Solution: We can use Markov inequality $\Pr[T > nH_n/q] \leq q$ (here T is the random variable telling us how many tosses are necessary).

   (c) **What is the Markov chain? Is this similar to a random walk on some graph?**

   Solution: There might be more Markov chains corresponding to this problem. The states could be all subsets of $[n] = \{1, 2, 3, \ldots, n\}$ (too big – not that nice to work with) or how many coupons have we collected so far (much smaller).

   This corresponds to the cover time of a complete graph (when we have loops in each vertes).

   (d) **Simulate.**

   Solution:

```python
import matplotlib.pyplot as plt
from collections import Counter
from random import randint

def catch_them_all(n: int = 50) -> int:
    coupons = [False] * n
    coupons_collected = 0
    coupons_bought = 0
    while coupons_collected < len(coupons):
        new_coupon = randint(0, len(coupons) - 1)
```

```
            coupons_bought += 1
            if not coupons[new_coupon]:
                coupons[new_coupon] = True
                coupons_collected += 1
    return coupons_bought

cnt = Counter(catch_them_all(50) for _ in range(10000))
plt.bar(cnt.keys(), cnt.values())
plt.xlabel("Steps untill collecting all 50 coupons")
plt.ylabel("How many times did we take this many steps")
# plt.show()
plt.savefig('coupon_collector.pdf')
```

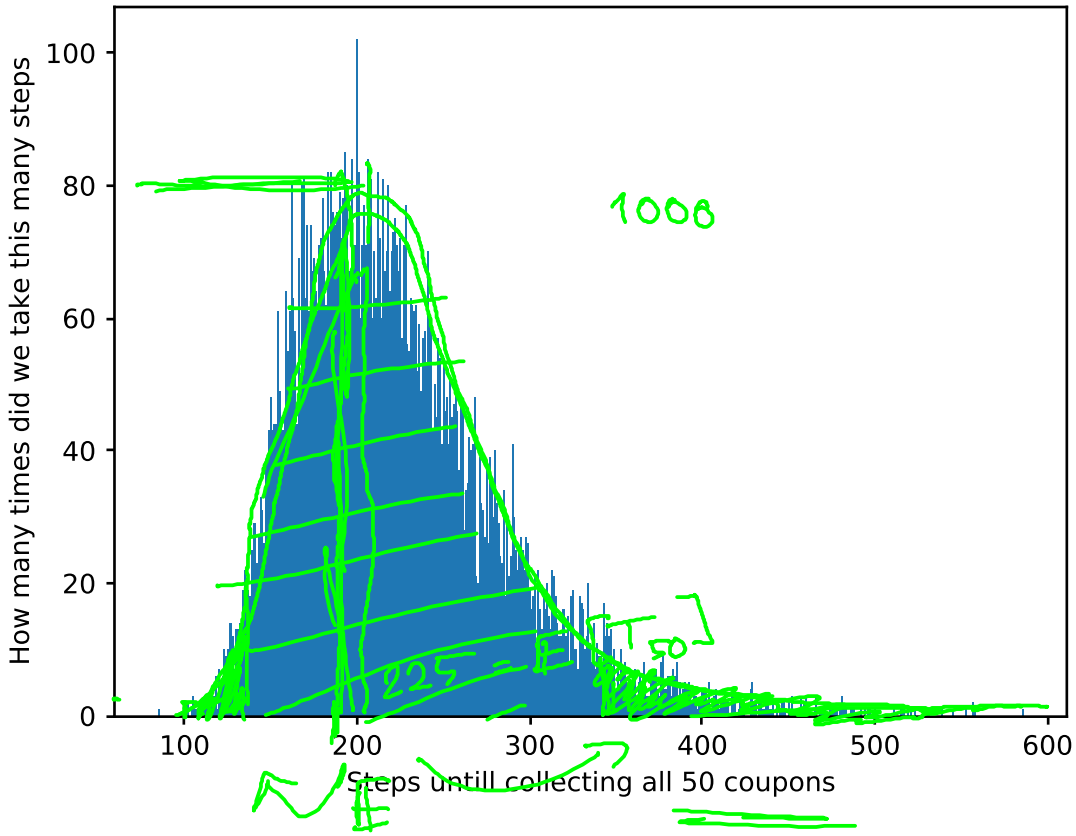


Figure 3.1: A histogram of how many steps were necessary (say 200 steps was necessary around 80 times).