Randomized Algorithms Notes

Wilson Pan

February 9, 2026

Contents

1 Jan 4

1.1 Background

Before analyzing the main problem, we briefly review two important concepts that will be central to our solution: the cycle decomposition of permutations and Markov's Inequality.

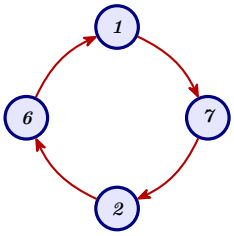
1.1.1 Permutations and Cycle Notation

A permutation σ of the set $\{1, \ldots, n\}$ is a bijection from the set to itself. We can decompose any permutation into disjoint cycles.

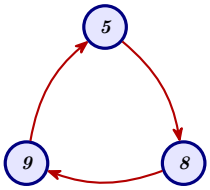
Example 1.1. *Consider the permutation defined by:*

$$\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 7 & 6 & 4 & 3 & 8 & 1 & 2 & 9 & 5 & 10 \end{pmatrix}$$

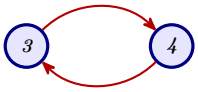
In disjoint cycle notation, this is written as $(1\ 7\ 2\ 6)(3\ 4)(5\ 8\ 9)(10)$. *We can visualize this decomposition below:*



(1 7 2 6)



(5 8 9)



(3 4)

(10)

1.1.2 Probabilistic Tools

We will also make use of Markov's Inequality to bound probabilities.

Theorem 1.2 (Markov's Inequality)**.** *If X is a non-negative random variable and $k > 0$ then:*

$$\mathbb{P}[X \geq k \cdot \mathbb{E}[X]] \leq \frac{1}{k}.$$

1.2 100 Prisoners Problem:

In a prison, there are 100 prisoners and there is a warden who want to retire. He decides to play a game.

1. In a room there are 100 boxes, In these boxes, is a permutation of the numbers $1, 2, ..., 100$. The boxes are closed and the prisoners cannot see the numbers within the boxes.
2. The prisoners will each enter and play a mini-game. For the i-th prisoner, they open 50 boxes and wins by finding the number i.
3. After a prisoner leaves, the boxes are closed again, and the state of the room is restored to exactly how it was before they entered. Prisoners cannot communicate after the game begins.
4. Prisoners win the game (and survive) if and only if **every** prisoner finds their own number. Otherwise, they are all executed.

Naive Algorithm: Each prisoner opens 50 random boxes then

$$\mathbb{P}[\text{Victory}] = \left(\frac{1}{2}\right)^{100} \approx 8 \times 10^{-31}.$$

1.3 Cycle Following Algorithm

We propose a strategy that utilizes the permutation structure inside the boxes.
Cycle Algorithm: Consider the algorithm where person i opens box i, sees j, then open box j, see k and so on. This continues until they find their number or exhaust their 50 tries.

Lemma 1.3. *Under the Cycle Algorithm, prisoner i succeeds if and only if i belongs to a cycle of length less than or equal to 50. Consequently, the team achieves victory if and only if the permutation contains no cycle of length strictly greater than 50.*

1.3.1 A Warm-up Bound using Markov's Inequality

Before calculating the exact probability, let us try to bound the failure probability of a slightly easier version of the game using Markov's Inequality.

Lemma 1.4. *Consider a variant where each prisoner is allowed to open 75 boxes. Using the Cycle Algorithm, the probability of failure is at most* $1/3$.

Proof. Let X be the number of prisoners who lose. We write

$$X = \sum_{i=1}^{100} I_i \text{ where } I_i = \begin{cases} 1 & \text{if prisoner } i \text{ loses} \\ 0 & \text{otherwise} \end{cases}.$$

First, observe that for any prisoner i, the probability that their specific number lies in the 25 unopened boxes is exactly $25/100 = 1/4$. Thus, $\mathbb{E}[I_i] = 1/4$. By linearity of expectation:

$$\mathbb{E}[X] = \sum_{i=1}^{100} \mathbb{E}[I_i] = 100 \cdot \frac{1}{4} = 25.$$

Now, observe the Cycle Algorithm: If prisoner i loses, they must be part of a cycle of length greater than 75. As cycles are disjoint, anyone else with a number in that same cycle will also lose. Therefore, if anyone loses like when $X > 0$, then strictly more than 75 people must lose so $X > 75$.
Using Markov's Inequality with $k = 3$:

$$\mathbb{P}[\text{Non-victory}] = \mathbb{P}[X > 0] = \mathbb{P}[X \geq 76] \leq \mathbb{P}[X \geq 75] \leq \frac{\mathbb{E}[X]}{75} = \frac{25}{75} = \frac{1}{3}.$$

□

1.3.2 Exact Probability Calculation

Theorem 1.5. *The Cycle Algorithm achieves a victory probability of* $\mathbb{P}[\textit{Victory}] \approx 0.31$.

Proof. The prisoners fail if and only if there exists a cycle of length $\ell > 50$. Since the sum of cycle lengths is 100, there can be at most one cycle of length greater than 50. Thus, the events of having a cycle of length ℓ (for $\ell > 50$) are mutually exclusive.

$$\mathbb{P}[\text{Non-victory}] = \mathbb{P}[\exists \text{ cycle of length } > 50] = \sum_{\ell=51}^{100} \mathbb{P}[\exists \text{ cycle of length } \ell].$$

To calculate the number of permutations of $N = 100$ elements that contain a cycle of length exactly ℓ.

1. Choose ℓ elements to be in the cycle: $\binom{N}{\ell}$ ways
2. Arrange these ℓ elements into a cycle: $(\ell - 1)!$ ways
3. Permute the remaining $N - \ell$ elements arbitrarily: $(N - \ell)!$ ways

The total number of such permutations is:

$$\binom{N}{\ell} \cdot (\ell - 1)! \cdot (N - \ell)! = \frac{N!}{\ell!(N - \ell)!}(\ell - 1)!(N - \ell)! = \frac{N!}{\ell}.$$

The probability that a random permutation contains a cycle of length ℓ is therefore:

$$\frac{N!/\ell}{N!} = \frac{1}{\ell}.$$

Summing these probabilities:

$$\mathbb{P}[\text{Non-victory}] = \sum_{\ell=51}^{100} \frac{1}{\ell} = H_{100} - H_{50} \approx \ln(100) - \ln(50) = \ln(2) \approx 0.69.$$

Thus, the probability of victory is $1 - 0.69 \approx 0.31$. □

Theorem 1.6. *The previous algorithm is optimal.*

Proof. Consider a second version of the game:

1. Prisoner 1 enters, opens 50 boxes
2. The difference between the games is that the boxes **remain open** for subsequent prisoners
3. Prisoner i enters. They win immediately if their box was already opened. Otherwise they may open more boxes until 50 total boxes (including those opened by previous prisoners) have been opened in the room

We rely on the following two lemmas to complete the theorem. □

Lemma 1.7. *The Cycle Algorithm achieves victory in Game 2 if and only if there are no cycles of length greater than 50.*

Proof. Consider when there are no cycles of length > 50, the Cycle Algorithm wins Game 1, and since Game 2 provides more information as boxes stay opened, it also wins Game 2. Conversely, suppose there is a cycle of length > 50. The first prisoner in that cycle to enter the room will be forced to open the boxes in that cycle, since the boxes were closed before they entered or if open, did not contain their number, they must follow the cycle. They will run out of their 50 openings before finding their number. Thus, the team fails. Therefore, the performance of the Cycle Algorithm is identical in Game 1 and Game 2. □

Lemma 1.8. *In Game 2, all strategies yield the same probability of victory.*

Proof. In Game 2, the boxes remain open, this means the decision of which box to open next depends on the set of boxes currently revealed. However, since the permutation inside the boxes is chosen uniformly, the values hidden in the unopened boxes are a random permutation of the remaining numbers.
Consequently, the probability that the *next* box opened contains a specific required number depends only on the number of boxes remaining, not on which specific box is chosen. Regardless of the strategy used, the probability that the union of opened boxes contains the set of all prisoners' numbers is not changed. □

Conclusion of Optimality: We have established the following chain of inequalities:

$$\mathbb{P}(\text{Cycle wins Game 1}) \leq \mathbb{P}(\text{Optimal Strategy wins Game 1}) \leq \mathbb{P}(\text{Optimal Strategy wins Game 2}).$$

The second inequality is true as any strategy for Game 1 is a strategy for Game 2 if we ignore the extra information, so Game 2 is strictly easier. However, by Lemma 4.3, all strategies in Game 2 are equivalent.

$$\mathbb{P}(\text{Optimal Strategy wins Game 2}) = \mathbb{P}(\text{Cycle wins Game 2}).$$

And by Lemma 4.2:

$$\mathbb{P}(\text{Cycle wins Game 2}) = \mathbb{P}(\text{Cycle wins Game 1}).$$

Combining these we get:

$$\mathbb{P}(\text{Cycle wins Game 1}) \leq \mathbb{P}(\text{Optimal Strategy wins Game 1}) \leq \mathbb{P}(\text{Cycle wins Game 1}).$$

Therefore, the inequalities must be equalities, and the Cycle Algorithm is optimal for Game 1.

2 Jan 21

2.1 Mcdiarmid's Theorem

Theorem 2.1. *Mcdiarmid's Theorem (1993) Given 3-colorable graph, we can efficiently compute a triangle-free partition.*
The algorithm is in each round

1. *Find some triangle in some part*
2. *Pick random vertex in the triangle*
3. *Move vertex to other side*

Proof. Pick a specific 3 coloring of the graph then let $X = \#$ color 1 vertices on left and $Y = \#$ color 2 vertices on right side. Observe that the value $X + Y$ changes by $\{-1, 0, 1\}$ with equal probability. Then observe a random walk in $[0, n]$

Lemma 2.2. *Let $T(n) = \mathbb{E}[$time to get to $\pm n]$ then $T(n) = n^2$, but we restrict it to powers of 2.*
We can easily see the following recurrence holds and it implies the lemma

$$T(n) = T\left(\frac{n}{2}\right) + T\left(\frac{n}{2}\right) + \frac{1}{2}T(n).$$

□

So the expected number of rounds is $O\left(n^2\right)$

2.2 Some theorem

Theorem 2.3. *Given $G = (V, E)$ we want for each pair $x, y \in V$ to calculate $d(x, y) \pm 2$. We will do it in $O(n^{2.5} \ln n)$.*

Proof. Consider the shortest path $x \to y$

Option 1: Path has no vertices with degree $> \sqrt{n}$. We can run BFS from each node, but exclude high degree nodes from search. So $O(|V| \cdot n^{1.5}) = O(n^{2.5})$

Option 2: There exist a high degree vertex on $x \to y$ path. Define $S = 100\sqrt{n} \log n$ vertices, each of which is an independent uniform random vertex.

Claim: With good probability, every high-degree vertex is incident to at least one vertex in S.
Observe that there exist $s \in S$ such that

$$d(x, s) + d(s, y) \leq d(x, y) + 2.$$

For each $s \in S$ do BFS and this runs at $O(n^{2.5} \log n)$ Then for each x, y compute

$$\min_{s \in S} d(x, s) + d(s, y).$$

This also is $O(n^{2.5} \log n)$.

$$\begin{aligned}\Pr[\text{given high degree vertex } w \not\sim S] &\leq \Pr[\text{every sample misses neigh}(c_v)] \\ &\leq \left(1 - \frac{1}{\sqrt{n}}\right)^{100\sqrt{n} \log n} \\ &\approx \left(\frac{1}{e}\right)^{100 \log n} \leq \frac{1}{n^{100}}\end{aligned}$$

□

3 Chernoff Bound (Feb 4)

Theorem 3.1. *Let $X_1, ..., X_n$ be i.i.d uniformly randomly in $\{0, 1\}$, let $X = \sum_{i=1}^{n} X_i$ then*

$$\Pr\left[X \geq \frac{n}{2} + k\sqrt{n}\right] \leq e^{-\frac{k^2}{2}}.$$

Theorem 3.2. *Let $X_1, ..., x_n$ be i.i.d random variables $\{-1, 1\}$ and $X = \sum_{i=1}^{n} X_i$ then*

$$\Pr\left[X \geq k\sqrt{n}\right] \leq e^{-\frac{k^2}{2}}.$$

Theorem 3.3. *Quick Sort:*

$$\Pr[\textit{depth} \geq 100 \log n] \leq \sum_{\textit{elts } v} \Pr[\textit{elt } v \textit{has recursion depth } 100 \log n] \leq \frac{1}{n^2}.$$

Let $P_1, ...,$ be subproblems containing v at levels $1, 2, ...$ If pivot in middle half ($\Pr \geq \frac{1}{2}$) then $|P_{i+1}| \leq \frac{3}{4}|P_i|$.
Define

$$X_i = \begin{cases} 1 & \textit{if } P_i \textit{ exist and } |P_i| > \frac{3}{4}|P_{i-1}| \\ 0 & \textit{otherwise} \end{cases}.$$

We have

$$\Pr\left[X_i = 1 \;\middle|\; X_1, ..., X_{i-1}\right] \leq \frac{1}{2}.$$

So

$$\Pr\left[\sum_{i=1}^{100 \log n} X_i \geq 750 \log n\right] \leq e^{-1000 \log n/32} = e^{-30 \log n}.$$

4 Feb 9

Lemma 4.1. *(Poor Man's Chernoff Bound)*

$$\Pr[X \geq 2k\sqrt{n}] \leq \frac{1}{2^k}.$$

Proof. Note:

$$\Pr[X \geq 2\sqrt{n}] \leq \frac{1}{4} \implies \Pr[\text{any prefix of } X \text{ is } \geq 2\sqrt{n}] \leq \frac{1}{2}. \qquad \text{(Chebyshev's Inequality)}$$

Let t_i be the event that the prefix of X reaches $2i\sqrt{n}$ then

$$\Pr[t_i \text{ exists} | t_{i-1} \text{ exist}] \leq \frac{1}{2}.$$

So

$$\Pr[X \geq 2k\sqrt{n}] \leq \Pr[t_k \text{ exists}] \leq \frac{1}{2^k}.$$

□

Lemma 4.2. *Let $X_1, ..., X_n$ be i.i.d random variables where each X_i satisfies*

$$\Pr[X_i \geq j] \leq p^j.$$

Then $X = \sum_{i=1}^{n} X_i$ satisfies

$$\Pr[X \geq 2n] \leq (4p)^n.$$

Proof.

$$\sum X_i \geq 2n \implies \sum \lfloor X_i \rfloor \geq n \implies \exists \vec{Y} = (Y_1, ..., Y_n) \text{ such that } X_i \geq Y_i \text{ for all } i \text{ and } \sum Y_i = n.$$

FOr a given $\vec{Y} = (Y_1, ..., Y_n)$ we have

$$\Pr[\vec{Y} \text{ occurs}] = \Pr[X_i \geq Y_i \ \forall i] = \prod_i \Pr[X_i \geq Y_i] \leq \prod_i p^{Y_i} = p^n.$$

We can encode $\vec{Y}$ as a binary string by stars and $\overline{s}$ □