Exercise 1. (Prisoners and Boxes) There are $2n$ prisoners whose names are placed in $2n$ boxes in a room, and they are led into the room one by one. Each prisoner may look into n boxes (that he chooses) and must leave the room exactly as he found it. The prisoners are allowed to agree on a (possibly randomly chosen) strategy in advance, but are permitted no further communication. If some prisoner does not find his own name in one of the boxes he opens, *all* of them will be executed.

1. Assume that each prisoner chooses his n boxes uniformly and independently at random. What is the chance of success (i.e., the probability that they won't be executed)?
2. Let π be a permutation over k elements. Define the associated graph G_π in the natural way, i.e., G_π is a directed graph that has k vertices, corresponding to the k elements, and each vertex v is connected to $\pi(v)$. A *cycle* in the permutation is a cycle in the associated graph. Show that if π is chosen uniformly at random, the probability that is has a cycle of length greater than $k/2$ is at most $\ln 2 \approx 0.693$.
3. Using the result from the last part, show that there is a strategy for the prisoners that has success probability at least 30%.

solution :

1. Each prisonner has a chance over two to find is own name. The probability is thus $\frac{1}{2^n}$.
2. The number of permutations on n elements is $n!$. The number of cycles on n elements is $(n-1)!$. For $\ell > k/2$, the number of permutations having a cycle of length exactly ℓ is given by
$$\binom{k}{\ell}(\ell-1)!(n-\ell)! = \frac{k!}{\ell} \; .$$
This is the number of way to choose ℓ elements to form a cycle, times the number of such cycles, times the number of possible permutations on the other elements. Notice that since $\ell > k/2$ we do not count any such permutation twice, since there can be only 1 cycle of this size in any permutation.
We conclude that the wanted probability is
$$\frac{1}{k!}\sum_{\ell>k/2}\frac{k!}{\ell} = \sum_{\ell>k/2}\frac{1}{\ell} \; .$$
We can upper bound this sum by
$$\int_{\lfloor k/2\rfloor}^{k}\frac{1}{t} = \ln k - \ln k/2 = \ln 2 \; .$$
3. The strategy works has follows. Each prisonners is assigned to a box and remember the box assigned to each of the others. On the experiment day, each prisonner open is own boxe, and if his name is not inside, open the box corresponding to the name he just read and so on.
This way, each prisonner actually follow the same permutation, and his able to find his name, if his boxes is not an a cycle of length grater than n. The result follows.

Exercise 2. Consider the $\sharp$DNF problem seen in class: we have a DNF formula ϕ and we want to compute
$$\sharp\phi = |\{x \in \{0,1\}^n \mid \phi(x) = 1\}|$$
One idea to get a FPRAS for this problem could be to use the following sampler

- choose an assigment τ in $\{0,1\}^n$ uniformly at random.
- output 2^n if $\phi(\tau) = 1$ and 0 otherwise.

We have seen in class that the expectation of this sampler is indeed $\sharp\phi$, show however that its variance is in general too large to be useful.

solution : The variance of this sampler is

$$E(X^2) - E(X)^2 = 2^{2n} \Pr(\phi(\tau) = 1) - (\sharp\phi)^2 ,$$

that is,

$$(2^n - \sharp\phi)\sharp\phi .$$

So the number of sample needed is in the order of

$$\frac{\sigma(X)^2}{\varepsilon^2(\sharp\phi)^2} = \frac{2^n - \sharp\phi}{\varepsilon^2\sharp\phi} .$$

This number can be exponential if $\sharp\phi$ is too small compared to 2^n.

Exercise 3. (von Neumann's extractor) Most randomized algorithms either require a source of unbiased and independent random bits to work properly, or are easier to analyze under such an assumption. However, obtaining pure random bits is very difficult in practice. This gives rise to the problem of *randomness extraction*, namely, that of producing purely random bits given a *weak* source of randomness.

A special and practically important case is when we have a source of *biased* but independent random bits (i.e., each bits is independently 1 with some unknown probability p). Show how to extract unbiased coin flips from such a source and compute the expected number of biased bits required for generating one unbiased bit.

solution : The idea is to throw 2 coins at the time and look for events $(1,0)$ or $(0,1)$. Both events will have the same probability, that is $p(1-p)$, and we can output 0 or 1 depending on which one we see first.

One of this event will happen with probability $2p(1-p)$, so the expected number of coin flip before observing one is $\frac{1}{p(1-p)}$. Notice that the 2 disapeared because we throw two coins at a time.