Collaborators : None

Sources :

1) https://www.perplexity.ai/

Q4 Dealing with error in randomized computation

4-a) If you have a zero-error algorithm $\boldsymbol{C}$ for $\boldsymbol{f}$ with failure probability $\mathbf{90}\%$ (quite high!), show how to convert it to a zero-error algorithm $\boldsymbol{C'}$ for $\boldsymbol{f}$ with failure probability at most $\mathbf{2^{-500}}$. The "slowdown" should only be a factor of a few thousand.

Algorithm for $\boldsymbol{C'}$:

1. Run $\boldsymbol{C}$ repeatedly for $\boldsymbol{k}$ times.
2. If any run produces a non-"?" result, return that result.
3. If all runs produce "?", return "?".

Analysis:

- Probability of all $\boldsymbol{k}$ runs failing: $\boldsymbol{p^k} \leq \mathbf{0.9^k}$
- We want: $\mathbf{0.9^k} \leq \mathbf{2^{-500}}$
- Taking logarithms: $k * log(0.9) \leq -500 * log(2) \Rightarrow k \geq \frac{(-500 * log(2))}{log(0.9)} \Rightarrow k \geq 3289.4067$

Therefore, we need $\boldsymbol{k} = \mathbf{3290}$ runs to achieve the desired failure probability. The slowdown factor is 3290, which is indeed "a factor of a few thousand" as required.

4-b) Alternatively, show how to convert $\boldsymbol{C}$ to an algorithm $\boldsymbol{C''}$ for $\boldsymbol{f}$ which: (i) always outputs the correct answer, meaning $\boldsymbol{C''(x) = f(x)}$; (ii) has expected running time only a few powers of 2 worse than that of $\boldsymbol{C}$. (Hint: look up the mean of a *geometric random variable*.)

Algorithm for $\boldsymbol{C''}$:

1. Repeatedly run $\boldsymbol{C}$ until a non-"?" result is obtained.
2. Return this result.

Analysis:

- The number of runs follows a geometric distribution with $\boldsymbol{p} = \mathbf{1} - \mathbf{0.9} = \mathbf{0.1}$ (success probability). This means that the probability that Algorithm returns the result after $\boldsymbol{k}$ runs is $\boldsymbol{P_k} = \mathbf{0.9}^{\boldsymbol{k}-\mathbf{1}} * \mathbf{0.1}$
- Expected number of runs (Average number of runs Algorithm takes to return the result) $= \frac{\mathbf{1}}{\boldsymbol{p}} = \frac{\mathbf{1}}{\mathbf{0.1}} = \mathbf{10}$

The expected running time of $\boldsymbol{C''}$ is 10 times that of $\boldsymbol{C}$, which is only a few powers of 2 worse, as required.

4-c) If you have a no-false-negatives algorithm $\boldsymbol{C}$ for $\boldsymbol{f}$ with failure probability 90% (quite high!), show how to convert it to a no-false-negatives algorithm $\boldsymbol{C'}$ for $\boldsymbol{f}$ with failure probability at most 2^{-500}. The "slowdown" should only be a factor of a few thousand.

Algorithm for $\boldsymbol{C'}$:

1. Run $\boldsymbol{C}$ for $\boldsymbol{k}$ times.
2. If any run outputs $\mathbf{1}$, return $\mathbf{1}$.
3. Otherwise, return $\mathbf{0}$.

Analysis:

- This preserves the no-false-negatives property.
- For $\boldsymbol{f}(\boldsymbol{x}) = \mathbf{0},\ \boldsymbol{C'}$ fails only if all $\boldsymbol{k}$ runs output $\mathbf{1}$.
- Probability of failure: $\mathbf{0.9}^{\boldsymbol{k}} \leq \mathbf{2}^{-\mathbf{500}}$

Solving for $\boldsymbol{k}$ as in part (a), we get $\boldsymbol{k} = \mathbf{3290}$. The slowdown factor is 3290, which is "a factor of a few thousand" as required.

4-d) If you have a two-sided error algorithm $\boldsymbol{C}$ for $\boldsymbol{f}$ with failure probability 40%, show how to convert it to a two-sided error algorithm $\boldsymbol{C'}$ for $\boldsymbol{f}$ with failure probability at most 2^{-500}. The "slowdown" should only be a factor of a few dozen thousand. (Hint: look up the Chernoff bound.)

Algorithm for $\boldsymbol{C'}$:

1. Run $\boldsymbol{C}$ for $\boldsymbol{k}$ times ($\boldsymbol{k}$ odd).
2. Return the majority vote of the $\boldsymbol{k}$ outputs.

Analysis using Chernoff bound:

Let $\boldsymbol{X}$ be the number of correct outputs in $\boldsymbol{k}$ runs.

$\boldsymbol{\mu} = \boldsymbol{E}[\boldsymbol{X}] = \mathbf{0.6}\boldsymbol{k}$ (as the success probability is 60%)

We want: $\boldsymbol{Pr}\left[\boldsymbol{X} \leq \frac{\boldsymbol{k}}{\mathbf{2}}\right] \leq \mathbf{2}^{-\mathbf{500}}$ (to get the majority votes for successful outcome)

Using the Chernoff bound with $\boldsymbol{\delta} = \frac{\mathbf{1}}{\mathbf{6}}$ for $\boldsymbol{Pr}[\boldsymbol{X} \leq (\mathbf{1} - \boldsymbol{\delta})\boldsymbol{\mu}] \leq \boldsymbol{e}^{\left(-\frac{\delta^2\mu}{2}\right)}$,

We need: $\boldsymbol{e}^{-\left(\frac{\delta^2\mu}{2}\right)} \leq \mathbf{2}^{-\mathbf{500}} \Rightarrow \left(\frac{\delta^2 * 0.6 * k}{2}\right) \geq \mathbf{500} * \boldsymbol{log}(\mathbf{2})$

Solving this inequality:

$\boldsymbol{k} \geq \frac{(2 * 500 * log(2) * 6^2)}{(0.6)} \approx 41588.83$

Therefore, we can use $\boldsymbol{k} = \mathbf{41589}$ (to make it odd) runs of $\boldsymbol{C}$. The slowdown factor is 41589, which is "a factor of a few dozen thousand" as required.