Aufgabe 1 – *Broken servers*

Sie sind mit einem Netzwerk verbunden, das aus n Servern besteht, die von 1 bis n nummeriert sind. Sie können jeden Server i in Zeit $\mathcal{O}(1)$ kontaktieren und erhalten als Anwort entweder eine '0' oder eine '1'. Leider sind einige der Server kaputt uns Sie sollen herausfinden welche Server betroffen sind.

- Falls Server i kaputt ist, sendet er bei jeder Anfrage ein unabhängig gleichverteiltes Bit.
- Falls Server i intakt ist, dann antwortet er auf jede Anfrage mit dem gleichen Bit $a_i \in \{0, 1\}$. Allerdings ist der Wert a_i unbekannt und kann von Server zu Server variieren.

(a) Seien $\delta > 0$ und $i \in [n]$ gegeben. Beschreiben Sie einen Monte-Carlo Algorithmus, der herausfindet, ob Server i kaputt ist. Berechnen Sie die Fehlerwahrscheinlichkeiten (abhängig davon ob der Server kaputt/intakt ist) Ihres Algorithmus und stellen Sie sicher, dass Ihr Algorithmus Fehlerwahrscheinlichkeit höchstens $\frac{\delta}{n}$ hat.

(b) Falls ein Server i kaputt (bzw. intakt) ist, ist es dann sicher, dass Ihr Algorithmus aus (a) dies herausfindet? Umgekehrt, wenn Ihr Algorithmus ausgibt, dass ein Server i kaputt (bzw. intakt) ist, können Sie sich sicher sein, dass diese Ausgabe korrekt ist?

(c) Sei $\delta > 0$ gegeben. Beschreiben Sie einen Monte-Carlo Algorithmus, der eine Liste aller kaputten Server erstellt und Fehlerwahrscheinlichkeit höchstens δ hat. Hierbei sagen wir, dass der Algorithmus erfolgreich ist, wenn die Liste alle kaputten Server und keinen intakten Server enthält.

Lösung zu Aufgabe 1 – *Broken Servers*

Die Aufgabe besteht darin, herauszufinden, ob ein Server kaputt ist, also uniform zufällig Bits zurückschickt, oder der Server intakt ist, also immer mit demselben Bit antwortet. Um dies herauszufinden, machen wir mehrere Anfragen an den gleichen Server und sehen, ob er immer die gleiche Antwort ausgibt, oder ob seine Antworten sich unterscheiden. Es bleibt zu überprüfen mit welchen Wahrscheinlichkeiten dies erfolgreich ist abhängig von der Anzahl Anfragen.

Wir beobachten, wenn wir zwei Anfragen an einen Server stellen, dann hat ein intakter Server die Wahrscheinlichkeit 1 dass die beiden Antworten gleich sind nach Aufgabenstellung. Wenn wir die zwei Anfragen jedoch an einen kaputten Server stellen, dann ist die Wahrscheinlichkeit jeweils ein Viertel für jede der möglichen Antworten $\{(0,0),(0,1),(1,0),(1,1)\}$. Insbesondere ist die Wahrscheinlichkeit nur $1/2$ dass wir die gleiche Antwort erhalten.

(a) Wir verallgemeinern dies von zwei auf t Anfragen. Sei Y_t das Ereignis, immer die selbe Antwort von einem Server zu erhalten. Dann ist diese Wahrscheinlichkeit $Pr[Y_t|\text{Server intakt}] = 1$ während $Pr[Y_t|\text{Server ist kaputt}] = 2^{1-t}$, weil wir uniform und unabhängige Bits erhalten. Wir schlagen folgenden Monte-Carlo Algorithmus vor:

Algorithm 1 BROKEN SERVER TEST(δ, i)

```
1: a_i ← Anfrage Server i
2: for j from 1 to t − 1 do
3:     b ← Anfrage Server i
4:     if b ≠ a_i then
5:         return "Server kaputt"                ▷ überprüfen ob Antworten gleich
6: return "Server funktioniert"                  ▷ Falls nie eine falsche Antwort kam
```

Wenn eine Anfrage an den Server in Zeit $\mathcal{O}(1)$ abgeschlossen wird, ist die Laufzeit $\mathcal{O}(t)$. Der Algorithmus terminiert also immer und ist deshalb ein Monte-Carlo Algorithmus.

Für die Korrektheit benutzten wir die oben berechneten Wahrscheinlichkeiten. Wenn der Server funktioniert ist die Wahrscheinlichkeit 1, dass alle Anfragen, also alle b's, gleich sind, und entsprechend gibt der Algorithmus immer die richtige Antwort. Wenn der Server kaputt ist, dann ist die Wahrscheinlichkeit, dass nicht alle Antworten gleich sind mindestens $1-2^{1-t}$. Also ist die Fehlerwahrscheinlichkeit höchstens 2^{1-t}. Wenn wir nun t wählen als $t \geq \log_2(\frac{n}{\delta}) + 1$, dann ist die Fehlerwahrscheinlichkeit höchstens $\frac{\delta}{n}$.

(b) Wir haben einen Monte Carlo Algorithmus mit einseitigem Fehler beschrieben. Das heisst wenn wir die Wahrscheinlichkeiten von (a) übernehmen, dann gilt

$$Pr[\text{Ausgabe "Server intakt"}|\ \text{Server intakt}] = 1$$

während

$$Pr[\text{Ausgabe "Server kaputt"}|\ \text{Server kaputt}] = 1 - 2^t$$

Also heisst dass der Algorithmus ist immer korrekt wenn der Server intakt ist und umgekehrt wenn die Ausgabe "Server kaputt" ist dann liegt der Algorithmus immer richtig. In den andern beiden Fällen können wir nie ganz sicher sein ob die Ausgabe korrekt war.

(c) Für diese Teilaufgabe wenden wir den Algorithmus (a) auf jeden Server an. Sei Z_i die Indikatorvariable für das Ereignis, dass der Algorithmus beim i-ten Server eine falsche Ausgabe gibt und $Z = \sum_i Z_i$. Um fehlzuschlagen, muss auf mindestens einem der Server eine falsche Ausgabe gemacht worden sein, also ist diese Wahrscheinlichkeit $\Pr[\text{Liste inkorrekt}] \leq \Pr[Z \geq 1] \leq n \cdot \frac{\delta}{n} = \delta$, wobei die Ungleichung aus der Markov Ungleichung oder einer Union Bound (boolsche Ungleichung) folgt: Aus (a) folgt $\mathbb{E}[Z_i] \leq \frac{\delta}{n}$, sodass mit Linearität des Erwartungswerts $\mathbb{E}[Z] \leq \delta$ gilt, und wir Markov anwenden können. Alternativ gilt dank der Union Bound:

Korrektheit des Algorithmus folgt aus der selben Argumentation wie (a).