Randomised Algorithms
Winter term 2022/2023, Exercise Sheet No. 4

Authors:
Ben Ayad, Mohamed Ayoub
Kamzon, Noureddine

November 13, 2022

Exercise 1.

(a) *Every deterministic algorithm has a predefined list of S that it checks in the same order, hence is s^* was the last item in the algorithm's list, it would be forced to try all words in S. To know this input we can try a naive approach, try all words of S as input, and collect the time it took the algorithm to break the lock, the input we are looking for would take the longest time.*

(b) *For $|S| = 1$ there is only one input and hence, $\mathbb{P}[T = 1] = 1 = 1/|S|$. Let's suppose that for some set S' of size $n \geq 1$ we have $\mathbb{P}[T = k] = \frac{1}{|S'|}$ for all $1 \leq k \leq n$.*

Let S be a set of size $n + 1$, we have the following for some $k \in \{1, \dots, n + 1\}$:

$$\mathbb{P}[T = k] = \mathbb{P}[T = k|T \leq n]\mathbb{P}[T \leq n] + \mathbb{P}[T = k|T = n + 1]\mathbb{P}[T = n + 1]$$

For $k \leq n$:
$\mathbb{P}[T = k|T \leq n] = \frac{1}{n}$ *(using the hypothesis, knowing that $T \leq n$, gives us one less choice and puts us back to the hypothesis n), and $\mathbb{P}[T = k|T = n + 1] = 0$, which yields, $\mathbb{P}[T = k] = \frac{1}{n}\mathbb{P}[T \leq n] = \frac{1}{n}\frac{n}{n+1} = \frac{1}{n+1}$*

For $k = n + 1$:
$\mathbb{P}[T = k] = \mathbb{P}[T = k|T = n + 1]\mathbb{P}[T = n + 1] = 1\frac{1}{n+1} = \frac{1}{n+1}$

Hence, for all $k \in \{1, \dots, n + 1\}$: $\mathbb{P}[T = k] = \frac{1}{n+1}$ which completes our induction.

For $|S| = n$, let's compute $\mathbb{E}[T]$:

$$\begin{aligned}
\mathbb{E}[T] &= \sum_{k=1}^{n} k\mathbb{P}[T = k] \\
&= \frac{1}{n}\frac{n(n + 1)}{2} \\
&= \frac{n + 1}{2}
\end{aligned}$$

(c) The hardest distribution p is a uniform one, otherwise (if p favoured some combinations), then there are always some deterministic algorithms that would check for those combinations first, and hence make the expected numbers of checks smaller in average.

Let p be the uniform distribution over words of S, let A be any optimal determinitic algorithm, hence, for each $k \in \{1, \dots, |S|\}$, there is one and only one input I_j such that $k = C(I_j, A)$, this observation justifies the equality [*] below.

$$\begin{aligned}\mathbb{E}[C(I_p, A)] &= \sum C(I_k, A)\mathbb{P}[I_k] \\ &= \frac{1}{|S|}\sum k \qquad \text{[*]} \\ &= \frac{|S|+1}{2}\end{aligned}$$

Now let q be a probability distribution over the set of deterministic algorithms $\mathcal{A}$, using Yao's minmax theorem we get:

$$\frac{|S|+1}{2} \leq \max_{I \in S} \mathbb{E}[C(I, A_q)]$$

From the last inequality, we can conclude that no randomized algorithm can do better in average that $\frac{|S|+1}{2}$, and hence the the algorithm in **(b)** is optimal.

Exercise 2.

Let $C = \{x_1, \dots, x_N\}$ be a random cut of the graph, where $\{x_i\}_{1 \leq i \leq N}$ representes the edges. We are obviously interested in $\mathbb{E}[N]$, i.e., the expected number of edges in a cut. Let $E = \{e_1, \dots, e_{|E|}\}$ and let the RV X_i be the indicator of edge e_i in C, i.e., $X_i = \delta(e_i \in C)$.

Clearly $N = \sum_{i=1}^{|E|} X_i$, and hence, $\mathbb{E}[N] = \sum_{i}^{|E|} \mathbb{E}[X_i]$

Now we prove that $\mathbb{E}(X_i) = 1/2$. Suppose the edge e_i connects the vertices A and B.

$$\begin{aligned}\mathbb{E}[X_i] &= \mathbb{P}[X_i = 1] \\ &= \mathbb{P}[\{\text{A random cut } S_1/S_2 \text{ contains } e_i\}] \\ &= \mathbb{P}[\{S_1 \text{ contains } A \text{ alone or } B \text{ alone}\}]\end{aligned}$$

Each cut defines a partition of vertices S_1/S_2, where S_1 selects $j \in \{1, \dots, |V| - 1\}$ vertices at random from V. Each vertex has 1/2 probability to be in S_1 (resp. S_2), which yields:

$$\begin{aligned}\mathbb{E}[X_i] &= \mathbb{P}[\{(A, B) \in (S_1, S_2) \vee (A, B) \in (S_2, S_1)\}] \\ &= \mathbb{P}[\{(A, B) \in (S_1, S_2)\}] + \mathbb{P}[\{(A, B) \in (S_2, S_1)\}] \\ &= \mathbb{P}[\{A \in S_1 \wedge B \in S_2\}] + \mathbb{P}[\{A \in S_1 \wedge B \in S_1\}] \\ &= \mathbb{P}[\{A \in S_1\}]\mathbb{P}[\{B \in S_2\}] + \mathbb{P}[\{A \in S_1\}]\mathbb{P}[\{B \in S_1\}] \\ &= \frac{1}{2}\frac{1}{2} + \frac{1}{2}\frac{1}{2} = \frac{1}{2}\end{aligned}$$

Now we have: $\mathbb{E}[N] = \sum_i \mathbb{E}[X_i] = \frac{|E|}{2} \geq \frac{|E|}{2}$. Hence, there must be a cut that has at least $\frac{|E|}{2}$ edges.

Exercise 3.

(a) *The probability that ModeratelyFastCut outputs a given minimum cut, is the same as the probablity of the contraction algorithm (the while loop of MODERATELYFASTCUT) not cutting any edge from the minimum cut, which is, according to the notes:* $\frac{t(t-1)}{n(n-1)}$. *(Assumig the deterministic algrithm always outputs the correct answer)*

(b) *The running time could be expressed as follows:* $M(t,n) = (n-t)\mathcal{O}(n) + \mathcal{O}(t^3)$

(c) *If we run the algorithm N times, the running time would be:* $T_{Amp}(t,n,N) = N\mathcal{O}((t^3 - nt + n^2)$

To make it effictient, each run has to be efficient first, we find t that minimizes the polynomial $P(t) = t^3 - nt + n^2$ *given that* $t \in \{2, \ldots, n\}$. *A quick derivation would give the value* $[\sqrt{\frac{n}{3}}]$, *assuming n is large enough* ($n \geq 12$). *The new runtime: would be* $T(n,N) = N\mathcal{O}(n^2)$

And we would get the following upper bound:

$$
\begin{aligned}
\mathbb{P}[\{Error\}] &\leq \left(1 - \frac{t(t-1)}{n(n-1)}\right)^N \\
&\leq e^{-\frac{t(t-1)N}{n(n-1)}} \\
\text{(for } t = \sqrt{\frac{n}{3}}\text{)} \qquad &= e^{\left(-\frac{(\sqrt{n}-\sqrt{3})N}{3\sqrt{n}(n-1)}\right)} \\
&\leq e^{-\frac{N}{\sqrt{n}(\sqrt{n}-\sqrt{3})}} \\
&\leq e^{-\frac{N}{n}}
\end{aligned}
$$

(d) *We have the following results:*

- *For Fast cut:* $\mathcal{O}(n^2 \log^2(n))$
- *For Randomized Contraction:* $\Theta(n^4)$
- *ModeratelyFastCut: for* $t = [\sqrt{\frac{n}{3}}]$ *we would need* $\mathcal{O}(n)$ *repititions to guarentee a constant error (using the upper bound we derived in the past equation), plus, each run would take* $\mathcal{O}(n^2)$, *which yields the following:* $\mathcal{O}(n^3)$