Chapter 14

Applications of Chernoff's Inequality

By Sariel Har-Peled, March 19, 2024[1]

14.1. QuickSort is Quick

We revisit **QuickSort**. We remind the reader that the running time of **QuickSort** is proportional to the number of comparisons performed by the algorithm. Next, consider an arbitrary element u being sorted. Consider the ith level recursive subproblem that contains u, and let S_i be the set of elements in this subproblems. We consider u to be *successful* in the ith level, if $|S_{i+1}| \leq |S_i|/2$. Namely, if u is successful, then the next level in the recursion involving u would include a considerably smaller subproblem. Let X_i be the indicator variable which is 1 if u is successful.

We first observe that if **QuickSort** is applied to an array with n elements, then u can be successful at most $T = \lceil \lg n \rceil$ times, before the subproblem it participates in is of size one, and the recursion stops. Thus, consider the indicator variable X_i which is 1 if u is successful in the ith level, and zero otherwise. Note that the X_is are independent, and $\mathbb{P}[X_i = 1] = 1/2$.

If u participates in v levels, then we have the random variables $X_1, X_2, \ldots, X_v$. To make things simpler, we will extend this series by adding independent random variables, such that $\mathbb{P}[`]X_i = 1 = 1/2$, for $i \geq v$. Thus, we have an infinite sequence of independent random variables, that are 0/1 and get 1 with probability 1/2. The question is how many elements in the sequence we need to read, till we get T ones.

Lemma 14.1.1. *Let $X_1, X_2, \ldots$ be an infinite sequence of independent random 0/1 variables. Let M be an arbitrary parameter. Then the probability that we need to read more than $2M + 4t\sqrt{M}$ variables of this sequence till we collect M ones is at most $2\exp\left(-t^2\right)$, for $t \leq \sqrt{M}$. If $t \geq \sqrt{M}$ then this probability is at most $2\exp\left(-t\sqrt{M}\right)$.*

Proof: Consider the random variable $Y = \sum_{i=1}^{L} X_i$, where $L = 2M + 4t\sqrt{M}$. Its expectation is $L/2$, and using the Chernoff inequality, we get

$$\begin{aligned}
\alpha = \mathbb{P}[Y \leq M] &\leq \mathbb{P}[|Y - L/2| \geq L/2 - M] \leq 2\exp\left(-\tfrac{2}{L}(L/2 - M)^2\right) \\
&\leq 2\exp\left(-2\left(M + 2t\sqrt{M} - M\right)^2/L\right) \leq 2\exp\left(-2\left(2t\sqrt{M}\right)^2/L\right) = 2\exp\left(-\frac{8t^2 M}{L}\right),
\end{aligned}$$

by Corollary 14.7.4. For $t \leq \sqrt{M}$ we have that $L = 2M + 4t\sqrt{M} \leq 8M$, as such in this case $\mathbb{P}[Y \leq M] \leq 2\exp\left(-t^2\right)$.

If $t \geq \sqrt{M}$, then $\alpha = 2\exp\left(-\dfrac{8t^2 M}{2M + 4t\sqrt{M}}\right) \leq 2\exp\left(-\dfrac{8t^2 M}{6t\sqrt{M}}\right) \leq 2\exp\left(-t\sqrt{M}\right)$. ■

Going back to the **QuickSort** problem, we have that if we sort n elements, the probability that u will participate in more than $L = (4 + c)\lceil \lg n \rceil = 2\lceil \lg n \rceil + 4c\sqrt{\lg n}\sqrt{\lg n}$, is smaller than $2\exp\left(-c\sqrt{\lg n}\sqrt{\lg n}\right) \leq 1/n^c$, by Lemma 14.1.1. There are n elements being sorted, and as such the probability that any element would participate in more than $(4 + c + 1)\lceil \lg n \rceil$ recursive calls is smaller than $1/n^c$.

Lemma 14.1.2. *For any $c > 0$, the probability that* **QuickSort** *performs more than $(6 + c)n \lg n$, is smaller than $1/n^c$.*

14.2. How many times can the minimum change?

Let $\Pi = \pi_1 \ldots \pi_n$ be a random permutation of $\{1, \ldots, n\}$. Let $\mathcal{E}_i$ be the event that π_i is the minimum number seen so far as we read Π; that is, $\mathcal{E}_i$ is the event that $\pi_i = \min_{k=1}^{i} \pi_k$. Let X_i be the indicator variable that is one if $\mathcal{E}_i$ happens. We already seen, and it is easy to verify, that $\mathbb{E}[X_i] = 1/i$. We are interested in how many times the minimum might change[2]; that is $Z = \sum_i X_i$, and how concentrated is the distribution of Z. The following is maybe surprising.

Lemma 14.2.1. *The events $\mathcal{E}_1, \ldots, \mathcal{E}_n$ are independent (as such, the variables $X_1, \ldots, X_n$ are independent).*

Proof: Exercise. ∎

Theorem 14.2.2. *Let $\Pi = \pi_1 \ldots \pi_n$ be a random permutation of $1, \ldots, n$, and let Z be the number of times, that π_i is the smallest number among $\pi_1, \ldots, \pi_i$, for $i = 1, \ldots, n$. Then, we have that for $t \geq 2e$ that $\mathbb{P}[Z > t \ln n] \leq 1/n^{t \ln 2}$, and for $t \in [1, 2e]$, we have that $\mathbb{P}[Z > t \ln n] \leq 1/n^{(t-1)^2/4}$.*

Proof: Follows readily from Chernoff's inequality, as $Z = \sum_i X_i$ is a sum of independent indicator variables, and, since by linearity of expectations, we have

$$\mu = \mathbb{E}[Z] = \sum_i \mathbb{E}[X_i] = \sum_{i=1}^{n} \frac{1}{i} \geq \int_{x=1}^{n+1} \frac{1}{x} \mathrm{d}x = \ln(n+1) \geq \ln n.$$

Next, we set $\delta = t - 1$, and use Theorem 14.7.2. ∎

14.3. Routing in a parallel computer

Let G be a graph of a network, where every node is a processor. The processor communicate by sending packets on the edges. Let $[0, \ldots, N-1]$ denote be vertices (i.e., processors) of G, where $N = 2^n$, and G is the hypercube. As such, each processor is identified by a binary string $b_1 b_2 \ldots b_n \in \{0, 1\}^n$. Two nodes are connected if their binary string differs only in a single bit. Namely, G is the binary ***hypercube*** over n bits.

We want to investigate the best routing strategy for this network topology. We assume that every processor need to send a message to a single other processor. This is represented by a permutation π, and we would like to figure out how to send the messages encoded by the permutation while creating minimum delay/congestion.

Specifically, in our model, every edge has a FIFO queue[3] of the packets it has to transmit. At every clock tick, the message in front of the queue get sent. All the processors start sending their packets (to the destination specified by the permutation) in the same time.

A routing scheme is ***oblivious*** if every node that has to forward a packet, inspect the packet, and depending only on the content of the packet decides how to forward it. That is, such a routing scheme is local in nature, and does not take into account other considerations. Oblivious routing is of course a bad idea – it ignores congestion in the network, and might insist routing things through regions of the hypercube that are "gridlocked".

[2] The answer, my friend, is blowing in the permutation.

[3] First in, first out queue. I sure hope you already knew that.

RandomRoute($v_0, \ldots, v_{N-1}$)

```
// v_i:  Packet at node i to be routed to node d(i).
```

(i) Pick a *random* intermediate destination $\sigma(i)$ from $[1, \ldots, N]$. Packet v_i travels to $\sigma(i)$.

```
// Here random sampling is done with replacement.
// Several packets might travel to the same destination.
```

(ii) Wait till all the packets arrive to their intermediate destination.

(iii) Packet v_i travels from $\sigma(i)$ to its destination $d(i)$.

Figure 14.1: The routing algorithm

Theorem 14.3.1 ([KKT91]). *For any deterministic oblivious permutation routing algorithm on a network of N nodes each of out-degree n, there is a permutation for which the routing of the permutation takes $\Omega\left(\sqrt{N/n}\right)$ units of time (i.e., ticks).*

Proof: (Sketch.) The above is implied by a nice averaging argument – construct, for every possible destination, the routing tree of all packets to this specific node. Argue that there must be many edges in this tree that are highly congested in this tree (which is NOT the permutation routing we are looking for!). Now, by averaging, there must be a single edge that is congested in "many" of these trees. Pick a source-destination pair from each one of these trees that uses this edge, and complete it into a full permutation in the natural way. Clearly, the congestion of the resulting permutation is high. For the exact details see [KKT91]. ■

How do we send a packet? We use ***bit fixing***. Namely, the packet from the i node, always go to the current adjacent node that have the first different bit as we scan the destination string $d(i)$. For example, packet from (0000) going to (1101), would pass through (1000), (1100), (1101).

The routing algorithm. We assume each edge have a FIFO queue. The routing algorithm is depicted in Figure 14.1.

14.3.1. Analysis

We analyze only step (i) in the algorithm, as (iii) follows from the same analysis. In the following, let ρ_i denote the route taken by v_i in (i).

Exercise 14.3.2. Once a packet v_j that travel along a path ρ_j can not leave a path ρ_i, and then join it again later. Namely, $\rho_i \cap \rho_j$ is (maybe an empty) path.

Lemma 14.3.3. *Let the route of a message* $\mathbf{c}$ *follow the sequence of edges $\pi = (e_1, e_2, \ldots, e_k)$. Let S be the set of packets whose routes pass through at least one of $(e_1, \ldots, e_k)$. Then, the delay incurred by* $\mathbf{c}$ *is at most $|S|$.*

Proof: A packet in S is said to leave π at that time step at which it traverses an edge of π for the last time. If a packet is ready to follow edge e_j at time t, we define its *lag* at time t to be $t - j$. The lag of $\mathbf{c}$ is initially zero, and the delay incurred by $\mathbf{c}$ is its lag when it traverse e_k. We will show that each step at which the lag of $\mathbf{c}$ increases by one can be charged to a distinct member of S.

We argue that if the lag of $\mathbf{c}$ reaches $\ell + 1$, some packet in S leaves π with lag ℓ. When the lag of $\mathbf{c}$ increases from ℓ to $\ell + 1$, there must be at least one packet (from S) that wishes to traverse the same edge as $\mathbf{c}$ at that time step, since otherwise $\mathbf{c}$ would be permitted to traverse this edge and its lag would not increase. Thus, S contains at least one packet whose lag reach the value ℓ.

Let τ be the last time step at which any packet in S has lag ℓ. Thus there is a packet $\mathbf{d}$ ready to follow edge e_μ at τ, such that $\tau - \mu = \ell$. We argue that some packet of S leaves π at time τ – this establishes the lemma since once a packet leaves π, it would never join it again and as such will never again delay $\mathbf{c}$.

Since $\mathbf{d}$ is ready to follow e_μ at time τ, some packet ω (which may be $\mathbf{d}$ itself) in S traverses e_μ at time τ. Now ω must leave π at time τ – if not, some packet will follow $e_{\mu+1}$ at step $\mu + 1$ with lag ℓ. But this violates the maximality of τ. We charge to ω the increase in the lag of $\mathbf{c}$ from ℓ to $\ell + 1$. Since ω leaves π, it will never be charged again. Thus, each member of S whose route intersects π is charge for at most one delay, establishing the lemma. ■

Let H_{ij} be an indicator variable that is 1 if ρ_i and ρ_j share an edge, and 0 otherwise. The total delay for v_i is at most $\leq \sum_j H_{ij}$.

Crucially, for a fixed i, the variables $H_{i1}, \ldots, H_{iN}$ are independent. Indeed, imagine first picking the destination of v_i, and let the associated path be ρ_i. Now, pick the destinations of all the other packets in the network. Since the sampling of destinations is done with replacements, whether or not the path ρ_j of v_j intersects ρ_i, is independent of whether ρ_k intersects ρ_i. Of course, the probabilities $\mathbb{P}[H_{ij} = 1]$ and $\mathbb{P}[H_{ik} = 1]$ are probably different. Confusingly, however, $H_{11}, \ldots, H_{NN}$ are not independent. Indeed, imagine k and j being close vertices on the hypercube. If $H_{ij} = 1$ then intuitively it means that ρ_i is traveling close to the vertex v_j, and as such there is a higher probability that $H_{ik} = 1$.

Let

$$\rho_i = (e_1, \ldots, e_k),$$

and let $T(e)$ be the number of packets (i.e., paths) that pass through e. We have that

$$\sum_{j=1}^{N} H_{ij} \leq \sum_{j=1}^{k} T(e_j) \quad \text{and thus} \quad \mathbb{E}\left[\sum_{j=1}^{N} H_{ij}\right] \leq \mathbb{E}\left[\sum_{j=1}^{k} T(e_j)\right].$$

Because of symmetry, the variables $T(e)$ have the same distribution for all the edges of G. On the other hand, the expected length of a path is $n/2$, there are N packets, and there are $Nn/2$ edges[4]. We conclude

$$\mathbb{E}[T(e)] = \frac{\text{Total length of paths}}{\text{\# of edges in graph}} = \frac{N(n/2)}{N(n/2)} = 1$$

$= 1$. Thus, for $k = |\rho_i|$, we have

$$\mu = \mathbb{E}\Big[\sum_{j=1}^{N} H_{ij}\Big] \leq \mathbb{E}\Big[\sum_{j=1}^{k} T(e_j)\Big] = \mathbb{E}\Big[\mathbb{E}\Big[\sum_{j=1}^{k} T(e_j) \,\Big|\, \rho_i\Big]\Big] = \mathbb{E}\Big[\sum_{j=1}^{|\rho_i|} \mathbb{E}\Big[T(e_j) \,\Big|\, \rho_i\Big]\Big] = \mathbb{E}\Big[\sum_{j=1}^{|\rho_i|} 1\Big] = \mathbb{E}\Big[|\rho_i|\Big] = \frac{n}{2}.$$

By the Chernoff inequality, specifically Lemma 14.7.3, we have

$$\mathbb{P}\left[\sum_j H_{ij} > 7n\right] \leq \mathbb{P}\left[\sum_j H_{ij} > (1 + 13)\mu\right] < 2^{-13\mu} \leq 2^{-6n}.$$

Since there are $N = 2^n$ packets, we know that with probability $\leq 2^{-5n}$ all packets arrive to their temporary destination in a delay of most $7n$.

Theorem 14.3.4. *Each packet arrives to its destination in $\leq 14n$ stages, in probability at least $1 - 1/N$ (note that this is very conservative).*

[4] Indeed, the hypercube has N vertices, all of degree n. As such, the number of edges is $Nn/2$.

14.4. Faraway Strings

Consider the Hamming distance between binary strings. It is natural to ask how many strings of length n can one have, such that any pair of them, is of Hamming distance at least t from each other. Consider two random strings, generated by picking at each bit randomly and independently. Thus, $\mathbb{E}[d_H(x, y)] = n/2$, where $d_H(x, y)$ denote the hamming distance between x and y. In particular, using the Chernoff inequality, specifically Corollary 14.7.4, we have that

$$\mathbb{P}[d_H(x, y) \leq n/2 - \Delta] \leq \exp\left(-2\Delta^2/n\right).$$

Next, consider generating M such string, where the value of M would be determined shortly. Clearly, the probability that any pair of strings are at distance at most $n/2 - \Delta$, is

$$\alpha \leq \binom{M}{2} \exp\left(-2\Delta^2/n\right) < M^2 \exp\left(-2\Delta^2/n\right).$$

If this probability is smaller than one, then there is some probability that all the M strings are of distance at least $n/2 - \Delta$ from each other. Namely, there exists a set of M strings such that every pair of them is far. We used here the fact that if an event has probability larger than zero, then it exists. Thus, set $\Delta = n/4$, and observe that

$$\alpha < M^2 \exp\left(-2n^2/16n\right) = M^2 \exp(-n/8).$$

Thus, for $M = \exp(n/16)$, we have that $\alpha < 1$. We conclude:

Lemma 14.4.1. *There exists a set of* $\exp(n/16)$ *binary strings of length n, such that any pair of them is at Hamming distance at least $n/4$ from each other.*

This is our first introduction to the beautiful technique known as the probabilistic method — we will hear more about it later in the course.

This result has also interesting interpretation in the Euclidean setting. Indeed, consider the sphere $\mathbb{S}$ of radius $\sqrt{n}/2$ centered at $(1/2, 1/2, \ldots, 1/2) \in \mathbb{R}^n$. Clearly, all the vertices of the binary hypercube $\{0, 1\}^n$ lie on this sphere. As such, let P be the set of points on $\mathbb{S}$ that exists according to Lemma 14.4.1. A pair p, q of points of P have *Euclidean* distance at least $\sqrt{d_H(p, q)} = \sqrt{n/4} = \sqrt{n}/2$ from each other. We conclude:

Lemma 14.4.2. *Consider the unit hypersphere $\mathbb{S}$ in $\mathbb{R}^n$. The sphere $\mathbb{S}$ contains a set Q of points, such that each pair of points is at (Euclidean) distance at least one from each other, and $|Q| \geq \exp(n/16)$.*

Proof: Take the above point set, and scale it down by a factor of $\sqrt{n}/2$. ■

14.5. Bibliographical notes

Section 14.3 is based on Section 4.2 in [MR95]. A similar result to Theorem 14.3.4 is known for the case of the wrapped butterfly topology (which is similar to the hypercube topology but every node has a constant degree, and there is no clear symmetry). The interested reader is referred to [MU05].

14.6. Exercises

Exercise 14.6.1 (More binary strings. More!). To some extent, Lemma 14.4.1 is somewhat silly, as one can prove a better bound by direct argumentation. Indeed, for a fixed binary string x of length n, show a bound on the number of strings in the Hamming ball around x of radius $n/4$ (i.e., binary strings of distance at most $n/4$ from x). (Hint: interpret the special case of the Chernoff inequality as an inequality over binomial coefficients.)

Next, argue that the greedy algorithm which repeatedly pick a string which is in distance $\geq n/4$ from all strings picked so far, stops after picking at least $\exp(n/8)$ strings.

14.7. From previous lectures

Corollary 14.7.1. *Let $X_1, \ldots, X_n$ be n independent* coin flips*, such that $\mathbb{P}[X_i = 0] = \mathbb{P}[X_i = 1] = \frac{1}{2}$, for $i = 1, \ldots, n$. Let $Y = \sum_{i=1}^{n} X_i$. Then, for any $\Delta > 0$, we have $\mathbb{P}[|Y - n/2| \geq \Delta] \leq 2\exp\left(-2\Delta^2/n\right)$.*

Theorem 14.7.2. *Let $X_1, \ldots, X_n$ be n independent variables, where $\mathbb{P}[X_i = 1] = p_i$ and $\mathbb{P}[X_i = 0] = q_i = 1 - p_i$, for all i. Let $X = \sum_{i=1}^{b} X_i$. $\mu = \mathbb{E}[X] = \sum_i p_i$. For any $\delta > 0$, we have*

$$\mathbb{P}[X > (1+\delta)\mu] < \left(e^{\delta} \big/ (1+\delta)^{1+\delta}\right)^{\mu}.$$

Lemma 14.7.3. *Let $X_1, \ldots, X_n$ be n independent Bernoulli trials, where $\mathbb{P}[X_i = 1] = p_i$, and $\mathbb{P}[X_i = 0] = 1 - p_i$, for $i = 1, \ldots, n$. Let $X = \sum_{i=1}^{b} X_i$, and $\mu = \mathbb{E}[X] = \sum_i p_i$. For $\delta > 2e - 1$, we have $\mathbb{P}[X > (1+\delta)\mu] < 2^{-\mu(1+\delta)}$.*

Corollary 14.7.4. *Let $X_1, \ldots, X_n$ be n independent* coin flips*, such that $\mathbb{P}[X_i = 0] = \mathbb{P}[X_i = 1] = \frac{1}{2}$, for $i = 1, \ldots, n$. Let $Y = \sum_{i=1}^{n} X_i$. Then, for any $\Delta > 0$, we have $\mathbb{P}[|Y - n/2| \geq \Delta] \leq 2\exp\left(-2\Delta^2/n\right)$.*

References

[KKT91] C. Kaklamanis, D. Krizanc, and T. Tsantilas. Tight bounds for oblivious routing in the hypercube. *Math. sys. theory*, 24(1): 223–232, 1991.

[MR95] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge, UK: Cambridge University Press, 1995.

[MU05] M. Mitzenmacher and U. Upfal. *Probability and Computing – randomized algorithms and probabilistic analysis*. Cambridge, 2005.