1 Frazer and Mc Kellar’s Randomized Sort

Let the input sequence be $X = k_1, k_2, k_3, \ldots k_n$. There are five steps in the algorithm:

1. Pick a random sample $S$ of $s$ elements from $X$;
2. Sort the sample. Let the sorted sample be $\ell_1, \ell_2, \ell_3, \ldots \ell_s$;
3. Partition the input into $(s+1)$ parts. $X_1 = \{q \in X : q \leq \ell_1\}; X_i = \{q \in X : \ell_{i-1} < q \leq \ell_i\}$, for $2 \leq i \leq s$; and $X_{s+1} = \{q \in X : q > \ell_s\}$;
4. for $i := 1$ to $(s + 1)$ do sort($X_i$);
5. Output sorted($X_1$), sorted($X_2$), ..., sorted($X_{s+1}$).

Lemma 1.1 If we pick a sample $S$ of $s$ elements, sort the sample and partition $X$ using the sample keys as splitters, then the size of each part is $\tilde{O}\left(\frac{n}{s}\log n\right)$.

Proof: Consider the elements of $X$ in sorted order. Let the group of the first $q$ elements be $G_1$; Let the group of the next $q$ elements be $G_2$; and so on.

If we can show that each of these groups ($G_1, G_2, \ldots$) has at least one sample key in it with high probability, it will mean that the size of each $X_i$ is no more than $2q$ with high probability.

Let us figure out the (smallest) value of $q$ for which each group ($G_j$) will have at least one sample key with high probability.

Probability that a specific element of $G_1$ is in the sample = $\frac{s}{n}$
Probability that a specific element of $G_1$ is not in the sample = $(1 - \frac{s}{n})$. 

1
Thus, probability that no element of $G_1$ is in the sample = $(1 - \frac{s}{n})^q$.

Since there are $\frac{n}{q}$ groups, $\text{Prob.} [\exists \text{ a group with the no representatives in the sample}] \leq \frac{n}{q} (1 - \frac{s}{n})^q$. We have used the following:

**Fact:**

$$\text{Prob}[A \cup B] \leq \text{Prob}(A) + \text{Prob}(B)$$

**Fact:**

For any $0 < x < 1$, $(1 - x)^{\frac{1}{x}} \leq \frac{1}{e}$

Using the above fact, $\text{Prob.} [\exists \text{ a group with the no representatives in the sample}] \leq n \left(1 - \frac{s}{n}\right)^{\frac{n}{q} \cdot \frac{s}{n}} \leq n e^{-\frac{qs}{n}}$.

We want the above probability to be $\leq n^{-\alpha}$.

Equating the two we get, $n e^{-\frac{qs}{n}} = n^{-\alpha} \Rightarrow e^{-\frac{qs}{n}} = n^{-(\alpha+1)} \Rightarrow \frac{qs}{n} = -(\alpha + 1) \log_e n \Rightarrow q = \frac{n}{s} (\alpha + 1) \log_e n$.

### 1.1 Analysis of the Run Time

1. Step 1 takes $s$ time.

2. Step 2 takes $O(s \log s)$ time.

3. Step 3 can be done using a binary search for each input key. In particular, if $k$ is any input key, we perform a binary search for $k$ in the sorted sample and figure out the part that $k$ belongs to. Total time taken is $n \log s$.

4. Step 4: Fix the value of $s$ to be $\frac{n}{\log^5 n}$. For this choice of $s$, the size of each $X_i$ is $\tilde{O}(\log^5 n)$. Sort each $X_i$ using heapsort, for example. The total time needed to sort all the $X_i$’s is $O(\sum_{i=1}^{s+1} |X_i| \log |X_i|) = \max_{i=1}^{s+1} \log(|X_i|) O(\sum_{i=1}^{s+1} |X_i|) = 5 \log \log n \ O(n) = O(n \log \log n)$.

Therefore the total number of comparisons made by the algorithm equals $n \log s + \tilde{O}(n \log \log n) = n \log n + \tilde{O}(n \log \log n)$. 

2
Fact

Any comparison sorting algorithm needs \(\log(n!)\) comparisons in the worst case.

Using Stirling’s approximation, \(\log(n!) \approx \log\left(\left(\frac{n}{e}\right)^n\right) = n \log n - n \log e\).

Thus the number of comparisons made by Frazer and McKellar’s algorithm is very close to the information theoretical lower bound.

2 Selection Problem

Inputs for the selection problem are a sequence \(X = k_1, k_2, \ldots, k_n\) and an integer \(i\) \((1 \leq i \leq n)\). The problem is to identify the \(i\)th smallest element of \(X\).

3 Floyd and Rivest’s Algorithm

1. Pick a random sample \(S\) of \(s\) elements from \(X\). Possible values for \(s\) are \(n^{\frac{2}{3}}, n^{\frac{3}{4}}, \ldots\), etc.

   Let \(K\) be the element to be selected.

   The number of elements \(\leq K\) in \(X = i\);

   The expected number of elements \(\leq K\) in \(S = i(n^{\frac{a}{n}})\).

   Definition: \(\text{Rank}(K, X) = |\{q \in X : q \leq K\}|\).

2. Pick two elements \(\ell_1\) and \(\ell_2\) from \(S\), whose ranks in \(S\) are \(i n^{\frac{a}{n}} - \delta\) and \(i n^{\frac{a}{n}} + \delta\), respectively, for some appropriate value \(\delta\).

   The elements \(\ell_1\) and \(\ell_2\) are such that they will bracket \(K\) (the element to be selected) and \(|\{q \in X : \ell_1 \leq q \leq \ell_2\}|\) will be ”small” with high probability.

   If \(\ell_1\) and \(\ell_2\) do not bracket \(K\) or if \(|\{q \in X : \ell_1 \leq q \leq \ell_2\}|\) is not ”small” we will start the algorithm all over again. However the probability of this happening is very low.
3.1 Analysis:

Pick \( \delta = c\alpha \sqrt{s \log n} \). The following Lemma will be used in the analysis.

Lemma 3.1 (Rajasekaran and Reif 1987) Let \( q \) be an element of \( S \) whose rank in \( S \) is \( j \). If \( r_j \) is the rank of \( q \) in \( X \), then,

\[
\text{Prob} \left( \left| r_j - j \frac{n}{s} \right| > \sqrt{3\alpha \frac{n}{\sqrt{s} \sqrt{\log n}}} \right) \leq n^{-\alpha}.
\]