Randomization in computing

(CSE 6512)

Notes of lecture 2 on Sep 01 2011

By

Seema Munavalli

Quick sort (1967 ,by Hoare)

Let $\Pi_1, \Pi_2, \Pi_3 \ldots\ldots\ldots \Pi_n$ be the sorted order of the input.

Let $X_{i,j} = \begin{cases} 1, & \text{if } \Pi_i \text{ and} \Pi_j \text{ will be compared} \\ 0, & \text{otherwise} \end{cases}$

Total # of comparison made= $\sum_{j=i+1}^{n} \sum_{i=1}^{n} X_{i,j}$

Let $P_{i,j}$ be the probability that Π_i and Π_j will be compared

$\Rightarrow$ Expected # of comparisons

$= E[\sum_{j>i} \sum_{i=1}^{n} X_{i,j}]$

$= \sum_{j>i} \sum_{i=1}^{n} \mathbf{E}(X_{i,j})$

*Since $X_{i,j}$ is a binary variable, it can take only values 0 or 1

$\Rightarrow E(X_{i,j}) = 1.P_{i,j} + (1 - P_{i,j}).0$

$= P_{i,j}$

$\Rightarrow$ Expected # of comparisons

$= \sum_{j>i} \sum_{i=1}^{n} P_{i,j}$ ---------- (1)

*Every input key acts as partition element at some point of time.

Consider $\Pi_1, \Pi_2, \Pi_3 \ldots \Pi_{i-1}, \boxed{\Pi_i, \Pi_{i+1} \ldots\ldots \Pi_{j-1}, \Pi_j}, \Pi_{j+1} \ldots\ldots \Pi_n$

*<u>Observation</u> : If Π_i or Π_j is picked as a pivot before any of the keys $\Pi_{i+1}, \ldots\ldots \Pi_{j+1}$,then

Π_i and Π_j will be compared, otherwise they will not be compared.

$\Rightarrow P_{i,j} = 2/(j-i+1)$----------(2)

$\Rightarrow$ Expected # of comparisons

$= \sum_{j=i+1}^{n} \sum_{i=1}^{n} 2/(j-i+1)$

$=2\sum_{i=1}^{n} [\frac{1}{2}+\frac{1}{3}+\ldots\ldots\frac{1}{n-i+1}]$

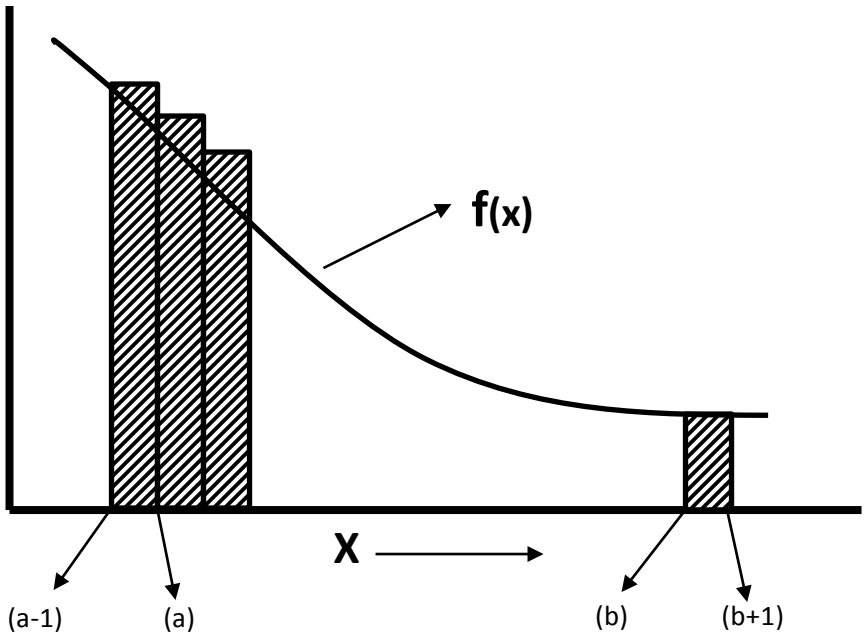
$<=2\sum_{i=1}^{n} [\frac{1}{2}+\frac{1}{3}+\ldots\frac{1}{n}]$

*Note that [1/2 +1/3+1/4+.....1/n] is a simple harmonic series ,that is

[1/2 +1/3+1/4+.....1/n] =Θ (log n)

$<=2\sum_{i=1}^{n} \Theta(\log n)$----------(3)

*Note that we can get very good upper and lower bounds on summations using integrals. Consider the below figure:



We can write

$$\int_a^{b+1} f(x)dx \le \sum_{i=a}^{b} f(i) \le \int_{a-1}^{b} f(x)dx \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (4)$$

Therefore using eqn 3 and eqn 4 we can write the runtime of this algorithm =O (n log n)

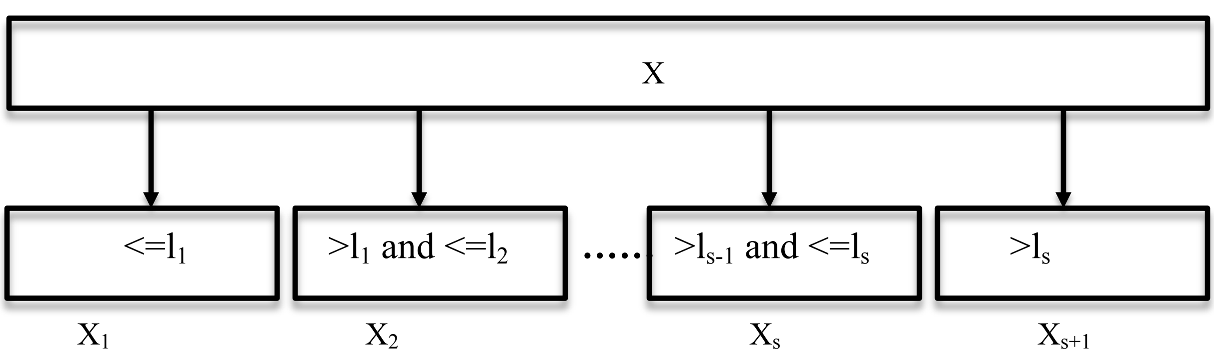
In the quicksort algorithm proposed by Hoare, the partitioning element is picked deterministically. The above average case analysis is done assuming that the rank of the pivot element could be any integer in the range [1,n] all with equal probability.

We can convert the above deterministic algorithm into a randomized algorithm by picking the pivot key with a coin flip. In this case the same analysis can be used to show that the average run time of this Las Vegas algorithm is O(n log n). Note that in the average case analysis of the deterministic algorithm the probability space under consideration is the space of all possible inputs whereas in the analysis of the randomized algorithm the probability space is the space of all possible outcomes for the coin flips.

A better randomized algorithm was proposed by Frazer and McKellar

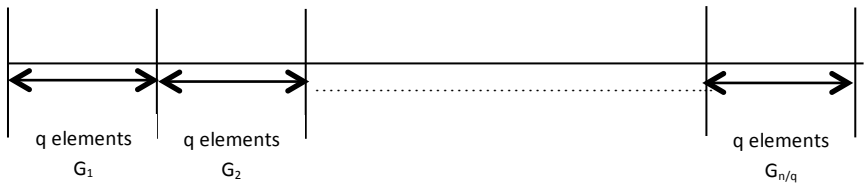
Let's prove a sampling lemma

- Let X be the a sequence of n elements $X=k_1,k_2\ldots\ldots k_n$.
  Pick a random sample S of size s.
  Sort S to get $l_1,l_2\ldots\ldots l_s$ and partition X into $X_1,X_2\ldots\ldots X_{s+1}$ as shown in the figure below:



LEMMA: The size of each X_i is $\tilde{O}(n/s \log n)$

Proof: Let $\Pi_1, \Pi_2, \Pi_3 \ldots\ldots\ldots \Pi_q, \ldots\ldots \Pi_{2q} \ldots\ldots \Pi_n$ be the sequence X in sorted order.



Partition the sequence into groups of size q each. Probability that a specific element of G_1 is in the sample S is s/n

⇒ probability that this specific element is not in S=(1-s/n)

Probability that G_1 has no representative in S is $<=(1-s/n)^q$.

⇒Probability that ∃ a group with no representative in S is $<=n/q(1-s/n)^q$.

$<=n(1-s/n)^q$.

We want this probability to be $<=n^{-\alpha}$

Fact : $\boxed{(1-x)^{1/x}<=1/e}$ **for any 1>x>0**

Using the above fact, the above probability is $<=n(1-s/n)^{n/s.s/n.q}$.

$<=n.e^{-s/n.q}$

Equate this with $n^{-\alpha}$

$n.e^{-s/n.q} = n^{-\alpha}$

⇒ $e^{-s/n.q} = n^{-(\alpha+1)}$

⇒ $-(s/n).q = -(\alpha+1)\log_e n$.

⇒ $q=n/s(\alpha+1)\log_e n$

⇒ The size of each X_i is $<= 2\ n/s(\alpha+1)\log_e n$ with probability $>=1- n^{-\alpha.}$

⇒ The size of each X_i is $\tilde{O}(n/s \log n)$.

Sorting Algorithm :-Input is $X=k_1,k_2\ldots k_n$.

1) Pick a Random Sample S with |S| =s

This step takes O(s) time [using uniform cost model, we can treat all the basic operations (including the coin flip) as unit time operations]

2) Sort S to get $l_1, l_2, \ldots\ldots l_s$.

This step takes O(s log s) time

3)Partition X to $X_1, X_2 \ldots X_{s+1}$ S.T

$X_1 = \{q \in X : q <= l_1\}$;

$X_i = \{q \in X : l_{i-1} < q <= l_i\}$; For i=2,.....s;

$X_{s+1} = \{q \in X : q > l_s\}$.

This step takes n log s time and can be done using binary search. I.e., for each input key we perform a binary search in the sorted sample to figure out which part this key belongs to.

4)For i=1…s+1 do

Sort X_i (using any asymptotically optimal sorting algorithm such as heapsort)

5)For i=1 to (s+1) do

Output X_i in sorted order;

Analysis :-

Pick s to be $n/(\log^2 n)$.

⇒ Size of each X_i is $\tilde{O}(\log^3 n)$.

Time for step 4 =

$$\sum_{i=1}^{s+1} O(|X_i| \log |X_i|)$$

$= O[Max_{i=1}^{s+1} \log |X_i| \sum_{i=1}^{s+1} |X_i|]$.

$= \tilde{O}(n.\log \log n)$.

⇒Runtime of this algorithm is

$O(n/\log^2 n)+O(n/\log n)+ n\log s+ \tilde{O}(n.\log\log n)$.

$=n \log s+ \tilde{O}(n.\log\log n)$.

$= n \log n+ \tilde{O}(n.\log\log n)$.

- A lower bound on the run time of any comparison based sorting algorithm is $\Omega(\log |n!|)$

  Stirling's approximation: $n! \approx (n/e)^n . \sqrt{2n\pi} .(1+1/11n)$. If we use this approximation, the lower bound for sorting run time is $\approx n \log n - n \log e$. As a result, the run time of Frazer and McKellar's algorithm is very nearly equal to the lower bound!