Randomized sorting

Matching lower and upper bounds for sorting

In what follows, we consider black-box Las Vegas algorithms for sorting and obtain essentially matching lower and upper bounds of about $n \log n$ on the expected number of comparisons in worst case (where n is the number of items in the list to be sorted).

The upper bound is obtained by a probabilistic argument that shows that randomized quicksort requires at most $1.4\, n \log n$ comparisons.

The lower bounds is obtained by arguing that

- any deterministic black-box algorithms requires roughly $n \log n$ comparisons on average when the inputs are chosen uniformly at random from the set of all inputs of size n,
- where the lower bound for deterministic algorithms extends to Las Vegas algorithms by Yao's Minimax Principle.

Randomized sorting

Algorithm $\mathrm{RandQuicksort}$ (Randomized Quicksort)

(Suppose that a strict linear ordering $<$ is understood.)

```
Input:   A list $S = (s_{\pi(1)}, \dots, s_{\pi(n)})$ of $n$ pairwise distinct items
         ($(s_1, \dots, s_n)$ is the ordered list, $\pi$ is a permutation).
  Pick an item s of $S$ uniformly at random.
  $S_{\mathrm{small}} = (s_{\pi(i)})_{s_{\pi(i)} < s}$
  $S_{\mathrm{large}} = (s_{\pi(i)})_{s_{\pi(i)} > s}$
  If $|S_{\mathrm{small}}| > 1$, then $S_{\mathrm{small}} = \mathrm{RandQuicksort}(S_{\mathrm{small}})$.
  If $|S_{\mathrm{large}}| > 1$, then $S_{\mathrm{large}} = \mathrm{RandQuicksort}(S_{\mathrm{large}})$.
Output:  $(S_{\mathrm{small}} \circ s \circ S_{\mathrm{large}})$          ($\circ$ is concatenation).
```

In order to sort a list S, $\mathrm{RandQuicksort}$ is invoked with input S.

Randomized sorting

Black-box sorting

Recall that we are considering black-box sorting, i.e., the only way an algorithm may obtain information about an item in the input list is to compare the item to another item.

Randomized and deterministic quicksort

- Algorithm $\mathrm{RandQuicksort}$ differs from deterministic quicksort precisely by the choice of the pivot elements that are used to split the list that is currently processed.
- For deterministic quicksort there are (rare) bad inputs on which the algorithm always uses about n^2 comparisons.
- There are no particular bad inputs for randomized quicksort, however, on any input, with small probability, randomized quicksort may use about n^2 comparisons.

Randomized sorting: an upper bound

Proposition

When algorithm $\mathrm{RandQuicksort}$ is run on any list of n pairwise distinct items, the expected number of comparisons required to sort the list is at most $1.4\, n \log n$.

Proof.

Let $S = (s_1, \dots, s_n)$ be any *ordered* list of n pairwise distinct items, and consider the application of $\mathrm{RandQuicksort}$ to any permutation $(s_{\pi(1)}, \dots, s_{\pi(n)})$ of S.

Any pair of items is compared at most once because

- the recursive calls to $\mathrm{RandQuicksort}$ are always for lists of items that have not yet been compared,
- during such a call, any pair of items is compared at most once.

The number of comparisons is just the number of pairs (s_i, s_j) with $i < j$ that are compared at all.

Randomized sorting: an upper bound

Proof (continued).

For any pair i, j where $1 \leq i < j \leq n$, consider the event that s_i is ever compared to s_j, and let p_{ij} be the probability of this event.

Furthermore, let X_{ij} be the corresponding indicator variable, (i.e., $X_{ij} = 1$ if the event occurs and $X_{ij} = 0$, otherwise).

The number of comparisons is just the sum over the X_{ij}, where

$$\mathbf{E}[X_{ij}] = p_{ij} \cdot 1 + (1 - p_{ij}) \cdot 0 = p_{ij} .$$

Hence the expected number of comparisons is

$$\mathbf{E}\left[\sum_{1 \leq i < j \leq n} X_{ij}\right] = \sum_{1 \leq i < j \leq n} \mathbf{E}[X_{ij}] = \sum_{1 \leq i < j \leq n} p_{ij} .$$

It remains to bound the sum of the probabilities p_{ij}.

Randomized sorting: an upper bound

Proof (continued).

Fix any pair of indices i and j where $i < j$ and consider the $j - i + 1$ items $s_i, \ldots, s_j$.

During the recursive calls, these items always stick together until for the first time one of these items is picked for splitting.

Each of these items has the same chance of being picked first.

In case the item from this list that is picked first

- differs from s_i and s_j, the two latter items are assigned to different sublists and are never compared.
- is equal to s_i or s_j, the two items are compared.

In summary, the probability that s_i and s_j are compared at all is

$$p_{ij} = \frac{2}{j - i + 1} .$$

Randomized sorting: an upper bound

Proof (continued).

The expected number of comparisons then is equal to

$$\begin{aligned}\sum_{1 \leq i < j \leq n} p_{ij} &= \sum_{1 \leq i < j \leq n} \frac{2}{j - i + 1} = 2\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{1}{j - i + 1} \\ &= 2\sum_{i=1}^{n-1} \sum_{k=2}^{n-i+1} \frac{1}{k} \leq 2n\sum_{k=2}^{n} \frac{1}{k} \leq 2n(\mathrm{H}_n - 1),\end{aligned}$$

where $\mathrm{H}_n = 1/1 + 1/2 + \ldots + 1/n$ is the nth Harmonic number.

Recall from the section on stable marriages that for all $n \geq 2$,

$$\mathrm{H}_n - 1 \leq \ln n = \ln 2 \cdot \log n < 0.7 \log n.$$

Hence the expected number of comparisons of $\mathrm{RandQuicksort}$ on any fixed input of size n is strictly less than $1.4\, n \log n$. $\square$

Randomized sorting: a lower bound

The upper bound of $1.4 n \log n$ comparisons for randomized Quicksort is essentially matched by the following lower bound.

Proposition

For any real number $\varepsilon > 0$ and for almost all n, when sorting lists of n pairwise distinct items by any black-box Las Vegas algorithm, in worst case the expected number of comparisons is at least

$$(1 - \varepsilon) n \log n .$$

Proof.

Fix any $\varepsilon > 0$ and n. By the discussion of black-box Las Vegas algorithms for sorting preceding Yao's Minimax Principle,

- there is a finite set $\mathcal{I}$ of inputs of size n,
- there is a finite set $\mathcal{A}$ of all correct deterministic black-box algorithms for sorting inputs of size n.

Randomized sorting: a lower bound

Proof (continued).

Furthermore, any black-box Las Vegas algorithm that correctly sorts all inputs in $\mathcal{I}$ can be identified with a probability distribution σ on $\mathcal{A}$, where then we refer to the algorithm by A_σ.

If we let $k(A, I)$ be the number of comparisons that algorithm A makes on input I, we have to show that for almost all n,

$$(1-\varepsilon)n\log n \le \max_{I\in\mathcal{I}} \mathbf{E}\left[k(A_\sigma, I)\right] . \tag{1}$$

By Yao's Minimax Principle, we have for any probability distribution σ on $\mathcal{A}$ and for the uniform distribution τ on $\mathcal{I}$,

$$\min_{A\in\mathcal{A}} \mathbf{E}\left[k(A, I_\tau)\right] \le \max_{I\in\mathcal{I}} \mathbf{E}\left[k(A_\sigma, I)\right] . \tag{2}$$

So we are done by the next propostion, which shows that for almost all n the left-hand side of (1) is a lower bound for the left-hand side of (2). □

Randomized sorting: a lower bound

Proposition

For any real number $\varepsilon > 0$ and for almost all n, when sorting inputs that are chosen uniformly at random from all permutations of the set $\{1, \dots, n\}$, for any correct deterministic black-box algorithm the average number of comparisons is at least

$$(1-\varepsilon)n\log n .$$

Proof.

Fix any $\varepsilon > 0$ and any natural number $n > 0$, and let $N = n!$.

We identify the set $\mathcal{I}$ of input lists with n elements with the set $\{S_1, \dots, S_N\}$ of all permutations of the set $\{1, \dots, n\}$.

Fix any deterministic black-box algorithm A that correctly sorts all lists in $\mathcal{I}$.

Randomized sorting: a lower bound

Proof (continued).

Let the word w_i be equal to the sequence of "answer bits" that algorithm A receives on input S_i when successively asking queries of the form "x < y?".

Then the average number of comparisons is $(|w_1| + \dots + |w_N|)/N$ and it suffices to show that this value is at least $(1-\varepsilon)n\log n$.

Claim 1 The set $\{w_1, \dots, w_N\}$ has size N and is prefix-free.

For a proof, first observe that there cannot be indices $i \ne j$ such that $w_i = w_j$. For such indices, the distinct inputs S_i and S_j could not be distinguished by A, hence at least one of these inputs would not be sorted correctly.

Similarly, there cannot be indices i and j where w_i is a proper prefix of w_j. For such indices, the distinct inputs S_i and S_j cannot be distinguished by A based on the first $|w_i|$ queries, whereas A asks further queries on input S_i but not on input S_j, a contradiction.

Randomized sorting: a lower bound

Proof (continued).

We say a prefix-free set $\{w_1, \dots, w_N\}$ of size N has minimum length sum if the sum $|w_1| + \dots + |w_N|$ is minimum among all prefix-free sets of size N.

Claim 2 There is a prefix-free set of size N that has minimum length sum such that the lengths of any two words in the set differ at most by 1.

For a proof, it suffices to consider a prefix-free set $\{w_1, \dots, w_N\}$ of size N that has minimum sum $\sum_{i=1}^{N} 2^{-|w_i|}$ among all prefix-free sets of size N that have minimum length sum.

If this set contains words w and w' such that $|w'| - |w| \ge 2$, then replacing w and w' by $w0$ and $w1$ yields as a contradiction another prefix-free set of size N that has minimum length sum and where the sum of the terms $2^{-|w_i|}$ has become strictly smaller due to

$$2^{-|w|} + 2^{-|w'|} > 2^{-|w|} = 2^{-|w0|} + 2^{-|w1|} .$$

Randomized sorting: a lower bound

Proof (continued).

By Claim 2, choose a prefix-free set $\{w_1, \ldots, w_N\}$ of size N that has mimimum length sum and where in addition all the w_i have length t or $t+1$ for some natural number t.

Accodingly, the minimum length sum is at least Nt and in order to show the proposition, it suffices to show $t \geq (1-\varepsilon) n \log n$.

There are 2^t and 2^{t+1} words of length t and $t+1$, respectively, and whenever the chosen prefix-free set contains a string w of length t, it contains neither $w0$ nor $w1$, hence 2^{t+1} is an upper bound for the size N of the set, i.e.,

$$t + 1 \geq \log N = \log n! \ . \tag{3}$$

Randomized sorting: a lower bound

Proof (continued).

For any natural number $k \geq 1$, among the n factors of $n!$ there are at most n/k that are less than or equal to n/k, hence we have

$$n! > (n/k)^{(n-n/k)} = (n/k)^{(1-1/k)n} \ . \tag{4}$$

Putting together (3) and (4), we get for any $k \geq 1$

$$t + 1 \geq \log n! > \log(n/k)^{(1-1/k)n} = (1 - 1/k)n(\log n - \log k) \ .$$

Now pick k where $1/k < \varepsilon/2$. Then it holds for all sufficiently large n that

$$\begin{aligned} t &> (1 - 1/k)\ n\ (\log n - \log k) - 1 \\ &\geq (1 - \varepsilon/2)\ n \log n - 1 \geq (1-\varepsilon) n \log n \ . \qquad \Box \end{aligned}$$

In the proof above, in place of inequality (4) one could also work with approximations of $n!$ or $\log n!$ related to Stirling's formula.