

## 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 the form  $c n \log n$  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  $4n \log n$  comparisons.

The lower bound is obtained by arguing that

any deterministic black-box algorithm 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 RandQuicksort (Randomized Quicksort)

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

Input: A list  $S = (s_1, \dots, s_n)$  of  $n$  pairwise distinct items,

Pick an item  $s$  of  $S$  uniformly at random.

$S_{\text{small}} = (s_i)_{s_i < s}$

$S_{\text{large}} = (s_i)_{s_i > s}$

If  $|S_{\text{small}}| > 1$ , then  $S_{\text{small}} = \text{RandQuicksort}(S_{\text{small}})$ .

If  $|S_{\text{large}}| > 1$ , then  $S_{\text{large}} = \text{RandQuicksort}(S_{\text{large}})$ .

Output:  $(S_{\text{small}} \circ s \circ S_{\text{large}})$  ( $\circ$  is concatenation).

In order to sort a list  $S$ , RandQuicksort is invoked with input  $S$ .

## Randomized sorting

### Randomized and deterministic quicksort

Algorithm RandQuicksort differs from deterministic quicksort precisely by the choice of the pivot items 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 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 RandQuicksort to any permutation of  $S$ .

Any pair of items is compared at most once because

the recursive calls to 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  $s_i, \dots, s_j$ .

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

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

where  $H_n = 1/1 + 1/2 + \dots + 1/n$  is the  $n$ th Harmonic number.

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

$$H_n - 1 \leq \ln n = \ln 2 \cdot \log n < 0.7 \log n.$$

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

## Randomized sorting: a lower bound

Next we consider black-box Las Vegas algorithms for sorting and obtain a lower bound of roughly  $n \log n$  on the average number of comparisons in worst case.

Lemma

Let  $\hat{A}$  be any black-box Las Vegas algorithm for sorting lists of  $n$  pairwise distinct items. Then  $\hat{A}$  can be represented as a probability distribution  $\sigma$  on the set  $\mathcal{A}$  of correct deterministic black-box algorithms for sorting such lists.

Sketch of proof.

By *behavior* of a black-box algorithm for sorting we refer to the mapping that determines for any given situation whether another comparison is made and if so, which one; here situation refers to the previously asked queries " $x_i < x_j$ ?" and their answers,

If we are only interested in the number of comparisons made, a correct deterministic black-box algorithm for sorting can be identified with its behavior.

## Randomized sorting: a lower bound

### Sketch of proof (continued).

Note that for any correct deterministic algorithm (and also for any Las Vegas algorithm) the output must be determined by the comparisons made and by the corresponding answers.

Then any randomized algorithm  $\hat{A}$  can be identified with the probability distribution  $\sigma$  on  $\mathcal{A}$  where the probability of any algorithm  $A$  in  $\mathcal{A}$  is just the probability that  $\hat{A}$  and  $A$  behave the same (e.g., if  $\hat{A}$  uses random coin tosses, then for any fixed sequence of coin tosses,  $\hat{A}$  can be viewed as a deterministic algorithm and has a certain behavior).

Conversely, any probability distribution  $\sigma$  on  $\mathcal{A}$  can be viewed as a randomized algorithm  $\mathcal{A}_\sigma$  where on any given input initially some deterministic algorithm is chosen according to  $\sigma$ .

When going from a randomized algorithm  $\hat{A}$  to the corresponding probability distribution  $\sigma$ , then going back  $\mathcal{A}_\sigma$ , the randomized algorithms  $\mathcal{A}_\sigma$  and  $\hat{A}$  will have the same probabilistic behavior.

## Randomized sorting: a lower bound

### Proposition

*For any any real number  $\varepsilon > 0$  and for almost all  $n$ , any black-box Las Vegas algorithm for sorting lists of  $n$  pairwise distinct items requires on average in worst case a number of comparisons of at least*

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

### Proof.

For given  $n$ , let  $\mathcal{A}$  be the set of all correct deterministic black-box algorithms for sorting lists of  $n$  pairwise distinct items.

We can assume that the set  $\mathcal{I}$  of inputs of size  $n$  is just the set of all permutations of the list  $1, \dots, n$ .

Furthermore, let  $k(A, I)$  be the number of comparisons that algorithm  $A$  makes on input  $I$ .

## Randomized sorting: a lower bound

### Proof (continued).

By the lemma above, any black-box Las Vegas algorithm for sorting lists of  $n$  pairwise distinct items can be identified with a probability distribution  $\sigma$  on  $\mathcal{A}$ .

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

$$\min_{A \in \mathcal{A}} \mathbf{E} [k(A, I_\tau)] \leq \max_{I \in \mathcal{I}} \mathbf{E} [k(A_\sigma, I)] . \quad (1)$$

So it suffices to show for given  $\varepsilon > 0$  and for almost all  $n$  that  $(1 - \varepsilon)n \log n$  is a lower bound on the expression on the left-hand side of inequality (1).

This is just the assertion of the following proposition. □

## Randomized sorting: a lower bound

### Proposition

*For any any real number  $\varepsilon > 0$  and for almost all  $n$ , any deterministic black-box algorithm for sorting lists of  $n$  pairwise distinct items requires on average at least  $(1 - \varepsilon)n \log n$  comparisons when sorting a list that is chosen uniformly at random from all permutations of the list  $1, \dots, n$ .*

### Proof.

Fix any  $\varepsilon > 0$  and let  $n$  be so large that (??) below is true.

Let  $N = n!$  and let  $\mathcal{I} = \{S_1, \dots, S_N\}$  be the set of all permutations of the list  $1, \dots, n$ .

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

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

## Randomized sorting: a lower bound

Proof (continued).

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

**Claim 1** The set  $\{w_1, \dots, w_N\}$  is prefix-free.

For a proof, first observe that there cannot be indices  $i$  and  $j$  where  $w_i$  is a proper prefix of  $w_j$ .

Otherwise, the inputs  $S_i$  and  $S_j$  could not be distinguished based on the first  $|w_i|$  queries, whereas on input  $S_i$  but not on input  $S_j$  additional queries are asked.

If, on the other hand, there were indices  $i \neq j$  such that  $w_i = w_j$ , then 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.

## Randomized sorting: a lower bound

Proof (continued).

**Claim 2** It holds that  $\sum_{i=1}^N 2^{-|w_i|} \leq 1$ .

For a proof, consider the random experiment where the bits of an infinite binary sequence  $\widehat{S}$  are determined by tosses of a fair coin.

For any index  $i$  let  $E_i$  be the event that  $w_i$  is a prefix of  $\widehat{S}$ .

The probability of  $E_i$  is just  $2^{-|w_i|}$ .

Since the set  $\{w_1, \dots, w_N\}$  is prefix-free, the events  $E_1, \dots, E_N$  are pairwise disjoint.

Hence the sum of the probabilities of the events  $E_i$  is at most 1.

## Randomized sorting: a lower bound

Proof (continued).

**Claim 3** There is a prefix-free set of words  $w_1, \dots, w_N$  where

the length sum  $|w_1| + \dots + |w_N|$  is minimum over all prefix-free sets of size  $N$ ,

and for all indices  $i$  and  $j$  holds  $|w_i| - |w_j| \leq 1$ .

For a proof, among all prefix-free sets  $\{w_1, \dots, w_N\}$  such that the length sum  $|w_1| + \dots + |w_N|$  is minimum, choose a set such that in addition  $\sum_{i=1}^N 2^{-|w_i|}$  is minimum.

If for this set there were indices  $i \neq j$  where  $|w_i| - |w_j| \geq 2$ , then replacing  $w_i$  and  $w_j$  by  $w_j0$  and  $w_j1$  yields another prefix-free set of size  $N$  without increasing the length sum, whereas the sum of the terms  $2^{-|w_i|}$  becomes strictly smaller due to

$$2^{-|w_i|} + 2^{-|w_j|} > 2^{-|w_j|} = 2^{-|w_j0|} + 2^{-|w_j1|},$$

which contradicts the choice of  $\{w_1, \dots, w_N\}$ .  $\square$

## Randomized sorting: a lower bound

Proof (continued).

By Claim 3, there is a prefix-free set  $\{w_1, \dots, w_N\}$  of minimum length sum where for some  $t$  all the  $w_i$  have length  $t$  or  $t + 1$ .

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

By Claim 2,  $N2^{-(t+1)}$  is at most 1, thus

$$2^{t+1} \geq N \quad \text{and} \quad t + 1 \geq \log n!.$$

Related to Stirling's formula, one can show  $n! \geq \sqrt{2\pi n}(n/e)^n$  for all  $n$ , and by having chosen  $n$  large enough, we have

$$\begin{aligned} t + 1 &\geq \log n! = \frac{1}{2} \log 2\pi + \left(n + \frac{1}{2}\right) \log n - n \log e \\ &\geq (1 - \varepsilon)n \log n + 1 \end{aligned} \quad (2)$$

$\square$