SORTING ALGORITHMS

DISCRETE STRUCTURES II

DARRYL HILL

BASED ON THE TEXTBOOK:

DISCRETE STRUCTURES FOR COMPUTER SCIENCE: COUNTING, RECURSION, AND PROBABILITY

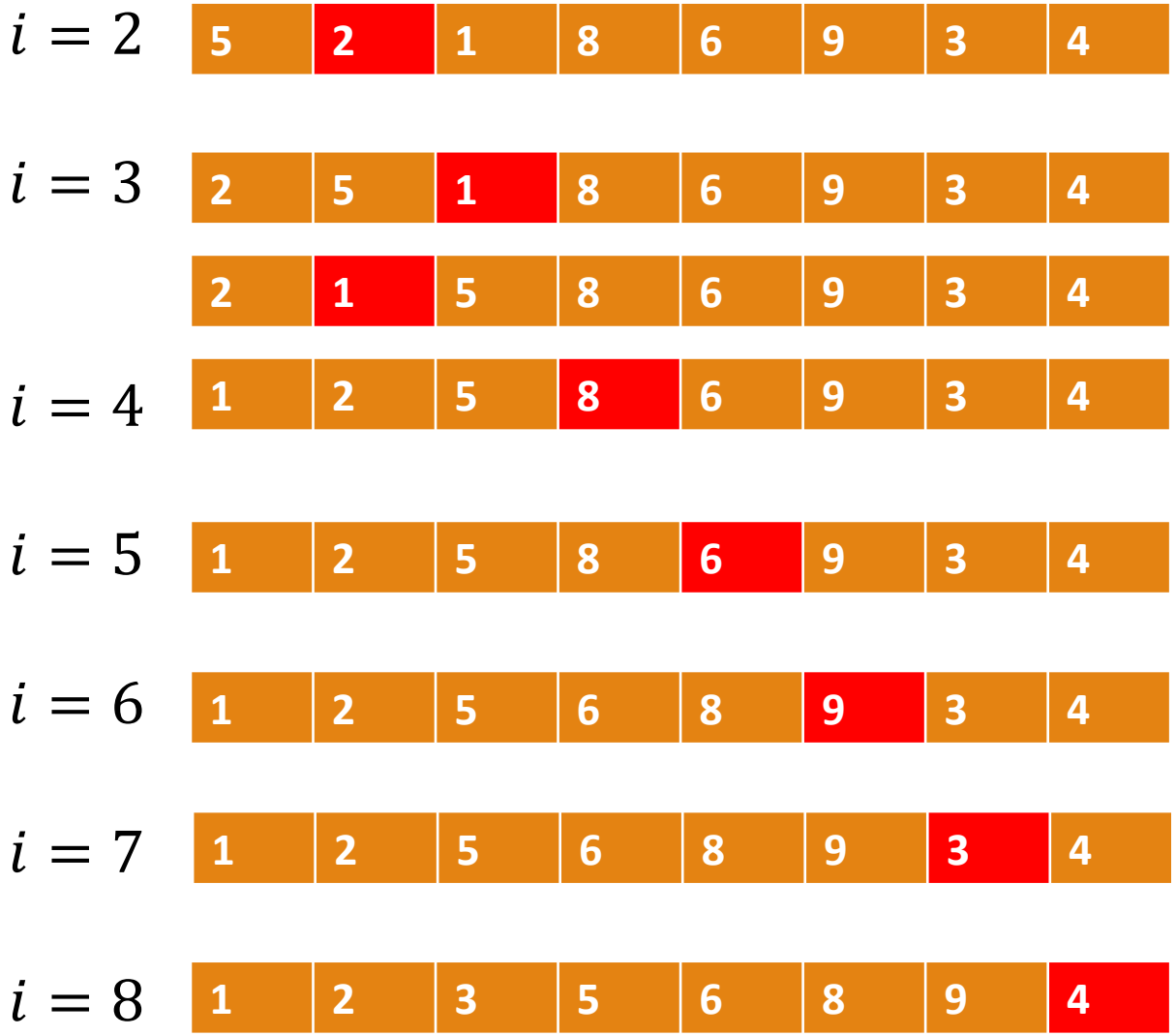
BY MICHIEL SMID

We will look at 2 sorting algorithms and use IRV to bound the running time.

```
Insertion-sort(A[1,..,n])
for i = 2 ... n:
    j = i;
    while j > i and A[j] < A[j − 1]:
        swap (A[j], A[i]);
        j = j − 1;
```

1	…	i-1	i	i+1	…	n
sorted			x	haven't seen		

Ex:



We will look at 2 sorting algorithms and use IRV to bound the running time.

Insertion-sort($A[1,..,n]$)
for $i = 2 \ldots n$:
  $j = i$;
  while $j > i$ and $A[j] < A[j-1]$:
    swap $(A[j], A[i])$;
    $j = j - 1$;

1	…	i-1	i	i+1	…	n
sorted			x	haven't seen		

What are the number of swaps?
What is the worst case?

6	5	4	3	2	1

n	n-1	…	3	2	1

We can observe that the # swaps is

$$1 + 2 + 3 + \cdots + n - 1 = \sum_{i=1}^{n-1} i = \frac{n(n-1)}{2}$$

in the worst case.

In this configuration, every pair of elements will be swapped. Then never swapped again.

Thus worst case is $\binom{n}{2}$ swaps (every pair gets swapped).

Can we use randomness to improve this?

```
Insertion-sort(A[1,..,n])
for i = 2 ... n:
   j = i;
   while j > i and A[j] < A[j − 1]:
        swap (A[j], A[i]);
        j = j − 1;
```

We will assume as input we receive a uniformly random permutation of $\{1 \dots n\}$

$$X = \#\text{ swaps}$$
$$E(X) = ?$$

1	…	i-1	i	i+1	…	n
sorted			x	haven't seen		

Consider location i.

How many swaps would we expect the number at location i to have?

In the worst case, it is swapped all the way back to location 1.

Since this is a uniformly random permutation, we might intuitively expect it to be swapped about halfway, on average.

Since $X = \binom{n}{2}$ in the worst case, we might guess that $E(X) = \frac{1}{2} \cdot \binom{n}{2}$.

Insertion-sort($A[1,..,n]$)
for $i = 2 \ldots n$:
  $j = i$;
  while $j > i$ and $A[j] < A[j-1]$:
    swap $(A[j], A[i])$;
    $j = j - 1$;

We will assume as input we receive a uniformly random permutation of $\{1 \ldots n\}$

$$X = \#\text{ swaps}$$

$$E(X) = \frac{1}{2} \cdot \binom{n}{2} ?$$

1	…	i-1	i	i+1	…	n
sorted			x	haven't seen		

We will use indicator random variables. What to indicate?

If two numbers are out of order, they will eventually be swapped with each other

(since each number moves one location at a time – they cannot "jump" over another number without being swapped).

Thus we will have a random variable for every pair of numbers to indicate if they swap or not.

Insertion-sort($A[1,..,n]$)
for $i = 2 \ldots n$:
  $j = i$;
  while $j > i$ and $A[j] < A[j-1]$:
    swap $(A[j], A[i])$;
    $j = j - 1$;

We will assume as input we receive a uniformly random permutation of $\{1 \ldots n\}$

$$X = \#\text{ swaps}$$

$$E(X) = \frac{1}{2} \cdot \binom{n}{2} ?$$

1	…	i-1	i	i+1	…	n
sorted			x	haven't seen		

For $1 \le a < b \le n$:

$$X_{ab} = \begin{cases} 1 \text{ if } a \text{ and } b \text{ are swapped} \\ 0 \text{ otherwise} \end{cases}$$

$$E(X_{ab}) = \Pr(X_{ab} = 1)$$

$$= \Pr(a \text{ and } b \text{ are swapped})$$

We look at the input array.

$X_{47} = 0$

	4	…	7		

$X_{47} = 1$

	7	…	4		

$$\Pr(a \text{ and } b \text{ are swapped}) = \Pr(b \text{ is left of } a)$$

Insertion-sort($A[1,..,n]$)
for $i = 2 \ldots n$:
  $j = i$;
  while $j > i$ and $A[j] < A[j-1]$:
    swap $(A[j], A[i])$;
    $j = j - 1$;

We will assume as input we receive a uniformly random permutation of $\{1 \ldots n\}$

$$X = \#\text{ swaps}$$

$$E(X) = \frac{1}{2} \cdot \binom{n}{2} ?$$

1	…	i-1	i	i+1	…	n
sorted			x	haven't seen		

The input is a uniformly random permutation.

What is the probability of the first case vs the probability of the second?

By symmetry, each of these has equal probability.

	4	…	7		

	7	…	4		

$$\Pr(X_{ab} = 0) = \frac{1}{2}$$

$$\Pr(X_{ab} = 1) = \frac{1}{2}$$

```
Insertion-sort(A[1, .., n])
for i = 2 ... n:
    j = i;
    while j > i and A[j] < A[j − 1]:
        swap (A[j], A[i]);
        j = j − 1;
```

We will assume as input we receive a uniformly random permutation of $\{1 \dots n\}$

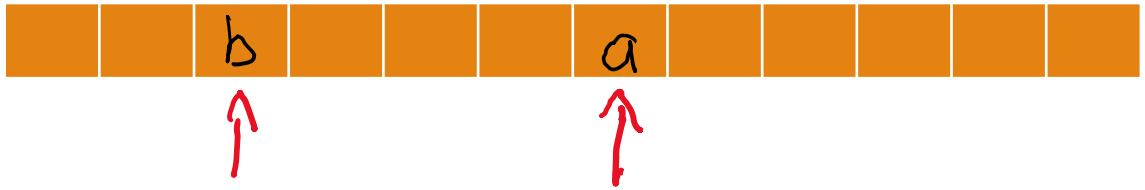
$$X = \#\text{ swaps}$$

$$E(X) = \frac{1}{2} \cdot \binom{n}{2} ?$$

1	…	i-1	i	i+1	…	n
sorted			x	haven't seen		

Or more carefully:
There are $n!$ permutations, in how many is b left of a? Use product rule.

Choose 2 spots:



$\binom{n}{2}$ ways to do this.

Place b on the left a on the right (1 way).

Permute the other $n - 2$ elements
$(n - 2)!$ ways.

Insertion-sort($A[1,..,n]$)
for $i = 2 \ldots n$:
  $j = i$;
  while $j > i$ and $A[j] < A[j-1]$:
    swap $(A[j], A[i])$;
    $j = j - 1$;

We will assume as input we receive a uniformly random permutation of $\{1 \ldots n\}$

$$X = \#\text{ swaps}$$

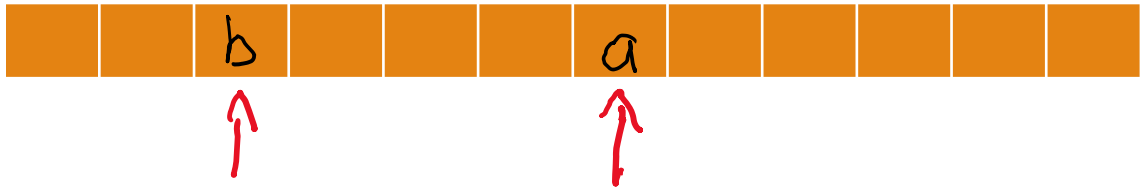
$$E(X) = \frac{1}{2} \cdot \binom{n}{2} ?$$

1	…	i-1	i	i+1	…	n
sorted			x	haven't seen		

Or more carefully:

There are $n!$ permutations.

In how many is b left of a?



#permutation with b left of a:

$$= \binom{n}{2} \cdot 1 \cdot (n-2)!$$

$$= \frac{n!}{(n-2)!2!} \cdot (n-2)!$$

$$= \frac{n!}{2}$$

Insertion-sort($A[1,..,n]$)
for $i = 2 \ldots n$:
  $j = i$;
  while $j > i$ and $A[j] < A[j-1]$:
    swap $(A[j], A[i])$;
    $j = j - 1$;

We will assume as input we receive a uniformly random permutation of $\{1 \ldots n\}$

$$X = \#\text{ swaps}$$

$$E(X) = \frac{1}{2} \cdot \binom{n}{2}?$$

$$X_{ab} = \begin{cases} 1 \; if \; a \; and \; b \; are \; swapped \\ 0 \; otherwise \end{cases}$$

$$E(X_{ab}) = \Pr(X_{ab} = 1)$$

$$= \frac{|X_{ab} = 1|}{|\#\text{ permutations}|}$$

$$= \frac{|X_{ab} = 1|}{n!}$$

$$= \frac{n!}{n! \cdot 2}$$

$$\frac{1}{2}.$$

Symmetry argument works just as well.

Insertion-sort($A[1,..,n]$)
for $i = 2 \ldots n$:
  $j = i$;
  while $j > i$ and $A[j] < A[j-1]$:
    swap $(A[j], A[i])$;
    $j = j - 1$;

We will assume as input we receive a uniformly random permutation of $\{1 \ldots n\}$

$$X = \#\text{ swaps}$$

$$E(X) = \frac{1}{2} \cdot \binom{n}{2} ?$$

$$X_{ab} = \begin{cases} 1 \; if \; a \; and \; b \; are \; swapped \\ 0 \; otherwise \end{cases}, E(X_{ab}) = \frac{1}{2}$$

$$X = \sum_{1 \le a < b \le n} X_{ab}$$

$$E(X) = E\left(\sum_{1 \le a < b \le n} X_{ab} \right)$$

$$= \sum_{1 \le a < b \le n} E(X_{ab})$$

$$= \sum_{1 \le a < b \le n} \frac{1}{2}$$

$$= \binom{n}{2} \cdot \frac{1}{2}$$

```
Insertion-sort(A[1,..,n])
for i = 2 ... n:
    j = i;
    while j > i and A[j] < A[j − 1]:
        swap (A[j], A[i]);
        j = j − 1;
```

We will assume as input we receive a uniformly random permutation of $\{1 \ldots n\}$

$$X = \#\text{ swaps}$$

$$E(X) = \frac{1}{2} \cdot \binom{n}{2} ?$$

$$X_{ab} = \begin{cases} 1 \; if \; a \; and \; b \; are \; swapped \\ 0 \; otherwise \end{cases}, E(X_{ab}) = \frac{1}{2}$$

So in the worst case, we have $\binom{n}{2}$ swaps.

Often with randomized algorithms, shuffling the input will improve the running time.

However in this case we see that randomizing the input only improves the worst case by a factor of ½ .

Asymptotically there is no improvement.

$$\binom{n}{2} \approx \frac{n^2}{2} = 0(n^2)$$

$$\frac{1}{2} \cdot \binom{n}{2} \approx \frac{n^2}{4} = 0(n^2)$$

QuickSort

Given a sequence S of integers in some permutation (not random), return them in sorted order.

$$S = \{1,2, \dots, n\}$$

Left	pivot	Right
<p	p	>p

QuickSort QuickSort

After step 3, p is in its correct location.
Elements in Left stay in Left
Elements in Right stay in Right
Base case is length 0 or 1

QuickSort Algorithm:

1. Select a *pivot* p uniformly at random.

2. Take all elements $< p$ and put them to the Left of p in the list.

3. Take all elements $> p$ and put them to the Right of p in the list.

4. Run QuickSort on elements on the Left

5. Run QuickSort on elements on the Right

QuickSort

Input:

Permutation of $\{1,2,\dots,n\}$ (not random)

$X = \#$ of comparisons

$E(X) = ?$

Left	pivot	Right
<p	p	>p

QuickSort QuickSort

We will use indicator random variables to count the number of comparisons:

$$1 \le a < b \le n:$$

$$X_{ab} = \begin{cases} 1 \text{ if } a \text{ and } b \text{ are swapped} \\ 0 \text{ otherwise} \end{cases}$$

$$E(X) = E\left(\sum_{1\le a<b\le n} X_{ab}\right)$$

$$= \sum_{1\le a<b\le n} E(X_{ab})$$

$$E(X_{ab}) = \Pr(X_{ab} = 1)$$

$$= \Pr(a \text{ and } b \text{ are compared})$$

$$E(X) = \sum_{1\le a<b\le n} \Pr(a \text{ and } b \text{ are compared})$$

QuickSort

Input:

Permutation of $\{1,2,\dots,n\}$ (not random)

$X = \#$ of comparisons

$E(X) = ?$

Left	pivot	Right
<p	p	>p

QuickSort QuickSort

We will use indicator random variables to count the number of comparisons:

$$1 \le a < b \le n:$$

$$X_{ab} = \begin{cases} 1 \text{ if } a \text{ and } b \text{ are swapped} \\ 0 \text{ otherwise} \end{cases}$$

$$E(X) = \sum_{1 \le a < b \le n} \Pr(a \text{ and } b \text{ are compared})$$

Observe all comparisons are made with the pivot p. For 2 integers a and b:

$$S_{ab} = \{a, a+1, a+2, \dots, b\}$$

For any given (recursive) call to QuickSort:

1. $p < a$
2. $p > b$
3. $a < p < b$
4. $a = p$ or $b = p$

QuickSort

Input:

Permutation of $\{1,2,\dots,n\}$ (not random)

$X = \#$ of comparisons

$E(X) = ?$

Left	pivot	Right
<p	p	>p

QuickSort QuickSort

$$1 \le a < b \le n:$$

$$X_{ab} = \begin{cases} 1 \text{ if } a \text{ and } b \text{ are swapped} \\ 0 \text{ otherwise} \end{cases}$$

$$E(X_{ab}) = \Pr(X_{ab} = 1)$$
$$= \Pr(a \text{ and } b \text{ are compared})$$

$$S_{ab} = \{a, a+1, a+2, \dots, b\}$$

1. $p < a$: after rearranging:

Left	pivot	Right
<p	p	>p, S_{ab}

All elements in S_{ab} are compared to p, but a has not been compared to b. But a and b are still together, so they may be compared in the future.

No comparison between a and b yet.

QuickSort

Input:

Permutation of $\{1,2,\dots,n\}$ (not random)

$X = \#$ of comparisons

$E(X) = ?$

Left	pivot	Right
<p	p	>p

QuickSort QuickSort

$1 \le a < b \le n$:

$$X_{ab} = \begin{cases} 1 \text{ if } a \text{ and } b \text{ are swapped} \\ 0 \text{ otherwise} \end{cases}$$

$$E(X_{ab}) = \Pr(X_{ab} = 1)$$
$$= \Pr(a \text{ and } b \text{ are compared})$$

$$S_{ab} = \{a, a+1, a+2, \dots, b\}$$

2. $p > b$: after rearranging:

Left	pivot	Right
<p, S_{ab}	p	>p

All elements in S_{ab} are compared to p, but a has not been compared to b. But a and b are still together, so they may be compared in the future.

No comparison between a and b yet.

QuickSort

Input:

Permutation of $\{1,2,\dots,n\}$ (not random)

$X = \#$ of comparisons

$E(X) = ?$

Left	pivot	Right
<p	p	>p

QuickSort QuickSort

$1 \le a < b \le n$:

$$X_{ab} = \begin{cases} 1 \text{ if } a \text{ and } b \text{ are swapped} \\ 0 \text{ otherwise} \end{cases}$$

$$E(X_{ab}) = \Pr(X_{ab} = 1)$$
$$= \Pr(a \text{ and } b \text{ are compared})$$

$$S_{ab} = \{a, a+1, a+2, \dots, b\}$$

3. a $< p <$ b: after rearranging:

Left	pivot	Right
<p, a	p	>p, b

Element a and b are now permanently separated and will never be compared.

QuickSort

Input:

Permutation of $\{1,2,\dots,n\}$ (not random)

$X = \#$ of comparisons

$E(X) = ?$

Left	pivot	Right
<p	p	>p

QuickSort QuickSort

$1 \le a < b \le n$:

$$X_{ab} = \begin{cases} 1 \text{ if } a \text{ and } b \text{ are swapped} \\ 0 \text{ otherwise} \end{cases}$$

$$E(X_{ab}) = \Pr(X_{ab} = 1)$$
$$= \Pr(a \text{ and } b \text{ are compared})$$

$$S_{ab} = \{a, a+1, a+2, \dots, b\}$$

4. $p = a$ or $p = b$: after rearranging:

Left	pivot	Right
<p	p	>p

If $p = a$, every element is compared with a including b.

If $p = b$, every element is compared with b including a.

QuickSort

Input:

Permutation of $\{1,2,\dots,n\}$ (not random)

$X = \#$ of comparisons

$E(X) = ?$

Left	pivot	Right
<p	p	>p

QuickSort QuickSort

$$1 \le a < b \le n:$$

$$X_{ab} = \begin{cases} 1 \text{ if } a \text{ and } b \text{ are swapped} \\ 0 \text{ otherwise} \end{cases}$$

$$E(X_{ab}) = \Pr(X_{ab} = 1)$$
$$= \Pr(a \text{ and } b \text{ are compared})$$

$$S_{ab} = \{a, a+1, a+2, \dots, b\}$$

1. $p < a$: no comparison yet.
2. $b < p$: no comparison yet

3. $a < p < b$: no comparison ever

$$X_{ab} = 0$$

Observe that for 3, $p \in S_{ab}$

4. $p = a$ or $p = b$: a and b are compared

$$X_{ab} = 1$$

Observe that for 4, $p \in S_{ab}$

That means that once $p \in S_{ab}$, the value of X_{ab} is fixed.

QuickSort

Input:

Permutation of $\{1,2,\dots,n\}$ (not random)

$X = \#$ of comparisons

$E(X) = ?$

Left	pivot	Right
<p, S_{ab}	p	>p

QuickSort QuickSort

$1 \le a < b \le n$:

$$X_{ab} = \begin{cases} 1 \text{ if } a \text{ and } b \text{ are swapped} \\ 0 \text{ otherwise} \end{cases}$$

$$E(X_{ab}) = \Pr(X_{ab} = 1)$$
$$= \Pr(a \text{ and } b \text{ are compared})$$

$$S_{ab} = \{a, a+1, a+2, \dots, b\}$$

$$|S_{ab}| \ge 2$$

Any choice of p that is not in S_{ab} results in S_{ab} being on the Right or Left.

Wlog assume Left. Recursively call QuickSort on Left.

We return without selecting a pivot when $|S| \le 1$. Since $S_{ab} \subseteq$ Left, $|\text{Left}| \ge 2$.

Thus we recursively call QuickSort on Left, which contains S_{ab}

This repeats until $p \in S_{ab}$.

QuickSort

Input:

Permutation of $\{1,2,\dots,n\}$ (not random)

$X = \#$ of comparisons

$E(X) = ?$

Left	pivot	Right
<p	p	>p

QuickSort QuickSort

$1 \le a < b \le n$:

$$X_{ab} = \begin{cases} 1 \text{ if } a \text{ and } b \text{ are swapped} \\ 0 \text{ otherwise} \end{cases}$$

$$E(X_{ab}) = \Pr(X_{ab} = 1)$$
$$= \Pr(a \text{ and } b \text{ are compared})$$

$$S_{ab} = \{a, a+1, a+2, \dots, b\}$$

1. $p < a$: no comparison yet.
2. $b < p$: no comparison yet
3. $a < p < b$: no comparison ever
$$X_{ab} = 0$$
4. $p = a$ or $p = b$: a and b are compared
$$X_{ab} = 1$$

3 and 4 correspond to $p \in S_{ab}$

Because $|S_{ab}| \ge 2$, one of these events (3 or 4) must happen.

There could be many recursive calls, but we will consider the first recursive call when $p \in S_{ab}$ (at which point, X_{ab} is decided).

QuickSort

Input:

Permutation of $\{1,2,\dots,n\}$ (not random)

$X = \#$ of comparisons

$E(X) = ?$

Left	pivot	Right
<p	p	>p

QuickSort QuickSort

$$1 \le a < b \le n:$$

$$X_{ab} = \begin{cases} 1 \text{ if } a \text{ and } b \text{ are swapped} \\ 0 \text{ otherwise} \end{cases}$$

$$E(X_{ab}) = \Pr(X_{ab} = 1)$$
$$= \Pr(a \text{ and } b \text{ are compared})$$

$$S_{ab} = \{a, a+1, a+2, \dots, b\}$$

At some point in the recursion, we choose a pivot uniformly at random from the set S_{ab}.

1. $a < p < b$: $X_{ab} = 0$

2. $p = a$ or $p = b$: $X_{ab} = 1$

a	a+1	a+2	…	b-2	b-1	b

$$|S_{ab}| = b - a + 1$$

The pivot is chosen uniformly at random, thus

$$\Pr(X_{ab} = 1)$$
$$= \Pr(p = a \text{ or } p = b) = \frac{2}{b-a+1}.$$

QuickSort

Input:

Permutation of $\{1,2,\dots,n\}$ (not random)

$X = \#$ of comparisons

$E(X) = ?$

Left	pivot	Right
<p	p	>p

QuickSort QuickSort

$1 \le a < b \le n$:

$$X_{ab} = \begin{cases} 1 \text{ if } a \text{ and } b \text{ are swapped} \\ 0 \text{ otherwise} \end{cases}$$

$$E(X_{ab}) = \Pr(X_{ab} = 1)$$
$$= \Pr(a \text{ and } b \text{ are compared})$$

$$\mathrm{E}(\mathrm{X_{ab}}) = \Pr(X_{ab} = 1) = \frac{2}{b-a+1}$$

$$E(X) = E\left(\sum_{a=1}^{n-1}\sum_{b=a+1}^{n} X_{ab}\right)$$

$$= \sum_{a=1}^{n-1}\sum_{b=a+1}^{n} E(X_{ab})$$

$$= \sum_{a=1}^{n-1}\sum_{b=a+1}^{n} \frac{2}{b-a+1}$$

$$= 2 \cdot \sum_{a=1}^{n-1}\sum_{b=a+1}^{n} \frac{1}{b-a+1}$$

QuickSort

Input:

Permutation of $\{1,2,\dots,n\}$ (not random)

$X =$ # of comparisons

$E(X) = ?$

Left	pivot	Right
<p	p	>p

QuickSort QuickSort

$1 \le a < b \le n$:

$$X_{ab} = \begin{cases} 1 \text{ if } a \text{ and } b \text{ are swapped} \\ 0 \text{ otherwise} \end{cases}$$

$$E(X_{ab}) = \Pr(X_{ab} = 1)$$
$$= \Pr(a \text{ and } b \text{ are compared})$$

$$E(X) = 2 \cdot \sum_{a=1}^{n-1} \sum_{b=a+1}^{n} \frac{1}{b-a+1}$$

$$= 2 \cdot \sum_{a=1}^{n-1} \left[\frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \cdots + \frac{1}{n-a+1}\right]$$

$$\le 2 \cdot \sum_{a=1}^{n-1} \left[\frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \cdots + \frac{1}{n}\right]$$

$$\le 2 \cdot \sum_{a=1}^{n-1} H_n - 1 \le 2 \cdot \sum_{a=1}^{n-1} \ln n + 1 - 1$$

$$\le 2 \cdot (n-1) \cdot \ln n \le 2 \cdot \mathrm{n} \ln n$$

QuickSort

Input:

Permutation of $\{1,2,\dots,n\}$ (not random)

$X = \#$ of comparisons

$E(X) = ?$

Left	pivot	Right
<p	p	>p

QuickSort QuickSort

$$1 \le a < b \le n:$$

$$X_{ab} = \begin{cases} 1 \text{ if } a \text{ and } b \text{ are swapped} \\ 0 \text{ otherwise} \end{cases}$$

$$E(X_{ab}) = \Pr(X_{ab} = 1)$$
$$= \Pr(a \text{ and } b \text{ are compared})$$

$$E(X) \le 2 \cdot n \ln n$$

You will learn in COMP2402 that, using comparison based sorting, you cannot do better than $O(n \cdot log\ n)$.

This is for all sorting algorithms invented, and also all sorting algorithms not yet invented.