CS514: Design and Analysis of Algorithms

Recursion: Sorting

Arijit Mondal

Dept of CSE

arijit@iitp.ac.in

https://www.iitp.ac.in/~arijit/

Sorting-1

- sortS(S, n)

1. if $n{=}1$ then return S_1
2. $x = \max(S)$; $S' = \{S\} - x$;
3. $S'' = \text{sortS}(S', n-1)$
4. return $\{S''\} * \{x\}$

Sorting-2

- sortI(S, n)

1. if n=1 then return s_1
2. Split S as follows: $S' = \{S\} - s_n$;
3. $S'' =$ sortI$(S', n-1)$
4. Insert s_n in S'' at appropriate location to obtain S'''
5. return S'''

Sorting-3

- sortM(S, n)

1. if n=1 then return S_1
2. Split S into two disjoint sets S_1, S_2
3. $S_1' =$ sortM(S_1, n_1)
4. $S_2' =$ sortM(S_2, n_2)
5. return merge(S_1, S_2)

Sorting-3

- sortM(S, n)

1. if n=1 then return S_1
2. Split S into two disjoint sets S_1, S_2
3. S'_1 = sortM(S_1, n_1)
4. S'_2 = sortM(S_2, n_2)
5. return merge(S_1, S_2)

- merge$(S_1[1 \ldots n_1], S_2[1 \ldots n_2])$

1. if $n_1 = 0$ return S_2
2. if $n_2 = 0$ return S_1
3. if $S_1[1] \leq S_2[1]$
4. return merge$(S_1[2 \ldots n_1], S_2[1 \ldots n_2])$
5. else
6. return merge$(S_1[1 \ldots n_1], S_2[2 \ldots n_2])$

Sorting-4

- sortQ(S, n)

1. if n=1 then return S_1
2. Choose a pivot $p \in S$
3. Split S in such a way that $S_1 = \{y | y \leq p\}$ and $S_2 = \{y | y > p\}$
4. $S'_1 = \text{sortQ}(S_1, l)$
5. $S'_2 = \text{sortQ}(S_2, n - l - 1)$
6. return $\{S'_1\} * \{p\} * \{S'_2\}$

Sorting-4

- sortQ$(S, n)`

1. if n=1 then return S_1
2. Choose a pivot $p \in S$
3. Split S in such a way that $S_1 = \{y|y \leq p\}$ and $S_2 = \{y|y > p\}$
4. $S_1' = \text{sortQ}(S_1, l)$
5. $S_2' = \text{sortQ}(S_2, n - l - 1)$
6. return $\{S_1'\} * \{p\} * \{S_2'\}$

- Partition(S, p)

1. swap $s_p \leftrightarrow s_n$
2. $l = 0$
3. for i in 1 to $(n - 1)$
4. if $s_i < s_n$ then $\{\ l = l + 1;$ swap $s_l \leftrightarrow s_i\ \}$
3. swap $s_n \leftrightarrow s_{l+1}$
6. return $l + 1$

Average case analysis-1

$$T(n) = cn + \frac{1}{n}\sum_{i=0}^{n-1}(T(i) + T(n-i-1)) = cn + \frac{2}{n}\sum_{i=0}^{n-1} T(i)$$

$$\Rightarrow nT(n) = cn^2 + 2\sum_{i=0}^{n-1} T(i)$$

$$\Rightarrow (n-1)T(n-1) = c(n-1)^2 + 2\sum_{i=0}^{n-2} T(i)$$

$$nT(n) - (n-1)T(n-1) = c(2n-1) + 2T(n-1)$$

$$nT(n) = (n+1)T(n-1) + c(2n-1)$$

$$\frac{T(n)}{n+1} = \frac{T(n-1)}{n} + \frac{2cn}{n(n+1)} - \frac{c}{n(n+1)}$$

Average case analysis-2

$$\frac{T(n)}{n+1} = \frac{T(n-1)}{n} + \frac{2cn}{n(n+1)} - \frac{c}{n(n+1)}$$
$$\frac{T(n-1)}{n} = \frac{T(n-2)}{n-1} + \frac{2c(n-1)}{(n-1)n} - \frac{c}{(n-1)n}$$
$$\frac{T(n-2)}{n-1} = \frac{T(n-3)}{n-2} + \frac{2c(n-2)}{(n-2)(n-1)} - \frac{c}{(n-2)(n-1)}$$
$$\vdots = \vdots$$
$$\frac{T(2)}{3} = \frac{T(1)}{2} + \frac{2c \times 2}{2 \times 3} - \frac{c}{2 \times 3}$$

$$\frac{T(n)}{n+1} = \frac{T(1)}{2} + 2c\left[\frac{1}{n+1} + \frac{1}{n} + \frac{1}{n-1} + \ldots + \frac{1}{3}\right] - c\left[\left(\frac{1}{n} - \frac{1}{n+1}\right) + \left(\frac{1}{n-1} - \frac{1}{n}\right) + \ldots + \left(\frac{1}{2} - \frac{1}{3}\right)\right]$$
$$\frac{T(n)}{n+1} = \frac{T(1)}{2} + 2cH_n - 2c \times \frac{3}{2} + \frac{2c}{n+1} - \frac{c(n-1)}{2(n+1)}$$
$$T(n) = 2c(n+1)\log n - 4cn - \frac{1}{2} = O(n \log n)$$

Lower bound for sorting

- Comparison based sorting has lower bound $\Omega(n \lg n)$

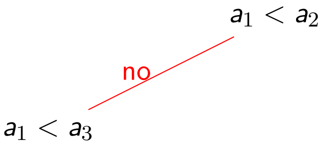
Lower bound for sorting

- Comparison based sorting has lower bound $\Omega(n \lg n)$

$$a_1 < a_2$$

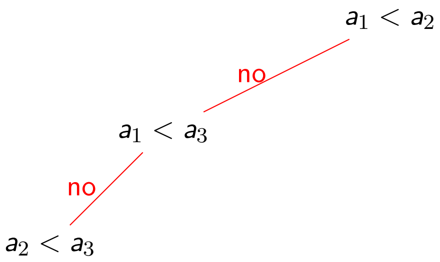
Lower bound for sorting

- Comparison based sorting has lower bound $\Omega(n \lg n)$



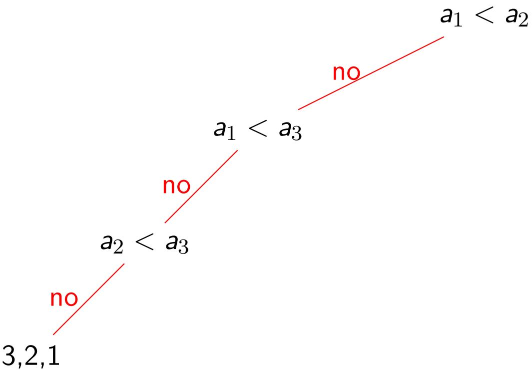
Lower bound for sorting

- Comparison based sorting has lower bound $\Omega(n \lg n)$



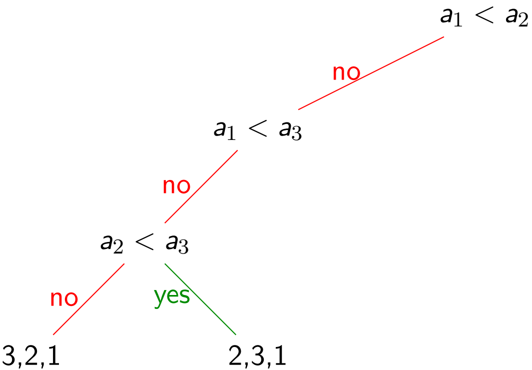
Lower bound for sorting

- Comparison based sorting has lower bound $\Omega(n \lg n)$



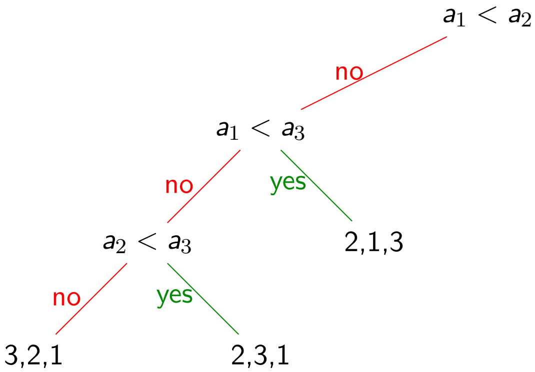
Lower bound for sorting

- Comparison based sorting has lower bound $\Omega(n \lg n)$



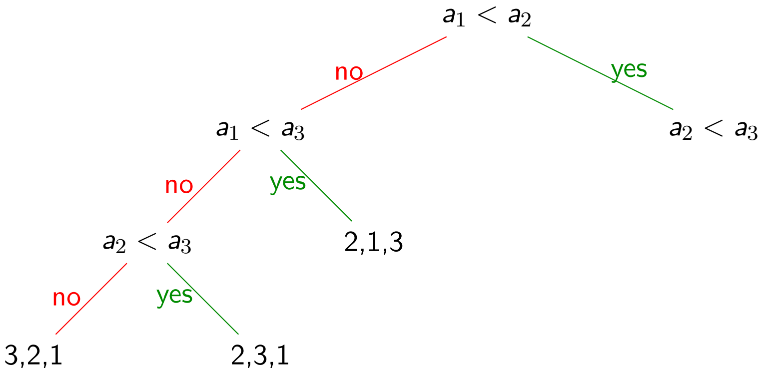
Lower bound for sorting

- Comparison based sorting has lower bound $\Omega(n \lg n)$



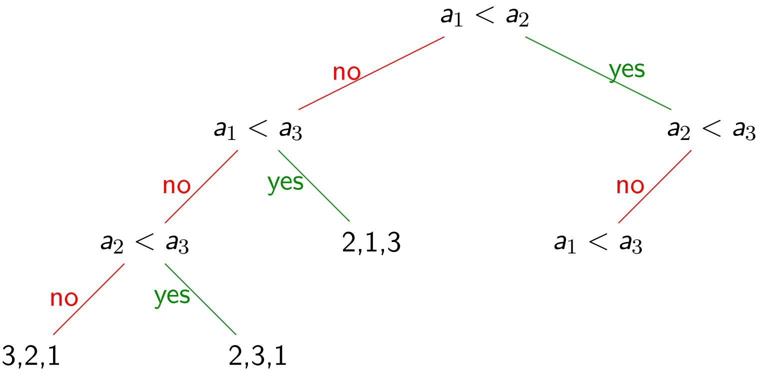
Lower bound for sorting

- Comparison based sorting has lower bound $\Omega(n \lg n)$



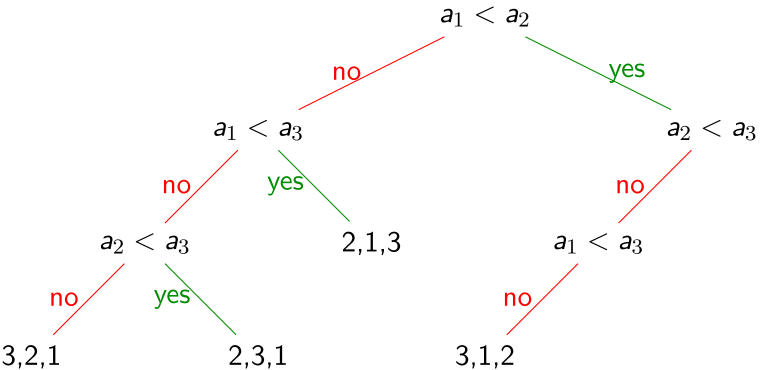
Lower bound for sorting

- Comparison based sorting has lower bound $\Omega(n \lg n)$



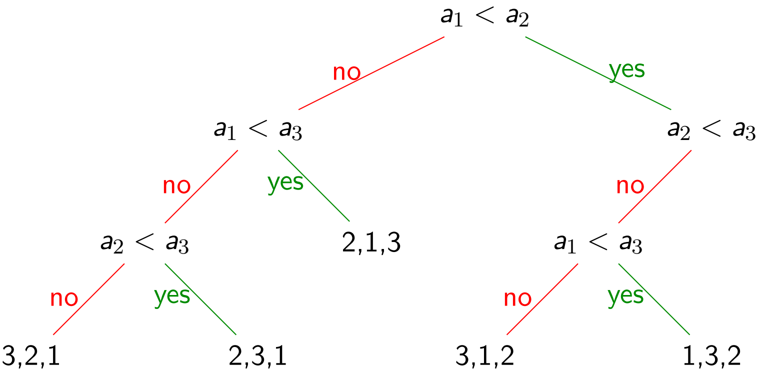
Lower bound for sorting

- Comparison based sorting has lower bound $\Omega(n \lg n)$



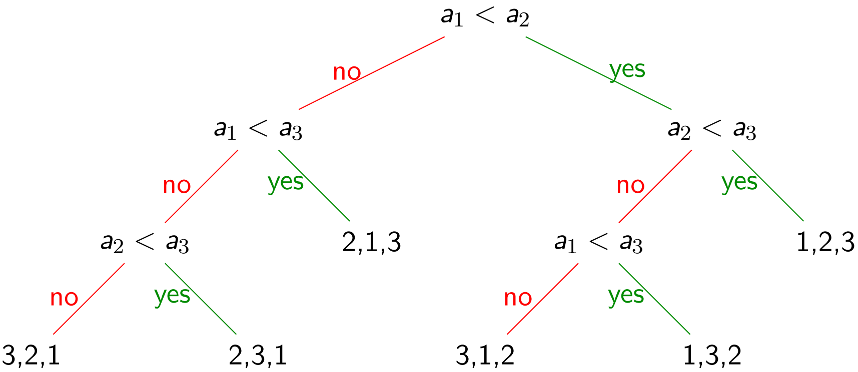
Lower bound for sorting

- Comparison based sorting has lower bound $\Omega(n \lg n)$



Lower bound for sorting

- Comparison based sorting has lower bound $\Omega(n \lg n)$



Exercise-1

- **Input**: An array A of distinct integers. **Output**: The number of inversions of A — the number of pairs (i, j) of array indices with $i < j$ and $A[i] > A[j]$
- Example: Consider the permutation — $5, 9, 1, 8, 2, 6, 4, 7, 3$. The number for inversions for 1-9 are as follows: $2, 3, 6, 4, 0, 2, 2, 1, 0$ (aka. inversion table). Hence total number of inversion is 20.
- Given an inversion table, is it possible to find original permutation?
- Can *inversion* count be used for collaborative filtering?

Exercise-2

- Can **four** elements be sorted using 5 comparison?
- Can **five** elements be sorted using 7 comparison?

Thank you!