

# Chapter 26

## Lower bounds

CS 573: Algorithms, Fall 2014

December 2, 2014

### 26.1 Sorting

#### 26.1.0.1 Sorting...

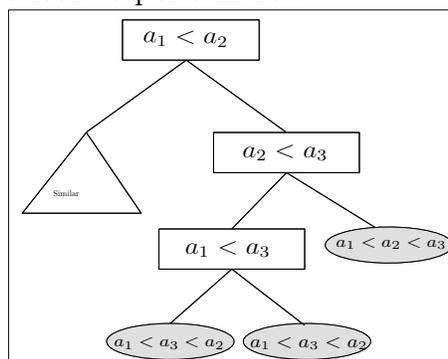
- (A)  $n$  items:  $x_1, \dots, x_n$ .
- (B) Can be sorted in  $O(n \log n)$  time.
- (C) Claim:  $\Omega(n \log n)$  time to solve this.
- (D) Rules of engagement: What can an algorithm do???

#### 26.1.0.2 Comparison model

- (A) In the *comparison model*:
  - (A) Algorithm only allowed to compare two elements.
  - (B) **compare**( $i, j$ ): Compare  $i$ th item in input to  $j$ th item in input.
- (B) **Q**: # calls to **compare** a deterministic sorting algorithm has to perform?

#### 26.1.0.3 Decision tree for sorting

- (A) sorting algorithm: a decision procedure.
- (B) Each stage: has current collection of comparisons done.
- (C) ... need to decide which comparison to perform next.



#### 26.1.0.4 Sorting algorithm...

- (A) sorting algorithm outputs a permutation.
- (B) ... order of the input elements so sorted.
- (C) Example: Input  $x_1 = 7, x_2 = 3, x_3 = 1, x_4 = 19, x_5 = 2$ .
  - (A) Output: 1, 2, 3, 7, 19.
  - (B) Output:  $x_3, x_5, x_2, x_1, x_4$ .
  - (C) Output:  $\pi = (3, 5, 2, 1, 4)$
  - (D) Output as permutation:  $\pi(1) = 3, \pi(2) = 5, \pi(3) = 2, \pi(4) = 1, \pi(5) = 4$ .
- (D)  $v$ : Node of decision tree.
  - $P(v)$ : A set of all permutations compatible with the set of comparisons from root to  $v$ .

#### 26.1.0.5 What are permutations?

- (A)  $\pi = (3, 4, 1, 2)$  is permutation in  $P(v)$ .
- (B) Formally  $\pi : \llbracket n \rrbracket \rightarrow \llbracket n \rrbracket$  is a one-to-one function.
  - $\llbracket n \rrbracket = \{1, \dots, n\}$
  - can be written as:
  - $$\pi = (3, 4, 1, 2) = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 4 & 1 & 2 \end{pmatrix}$$
- (C) Input is:  $x_1, x_2, x_3, x_4$
- (D) If arrived to  $v$  and  $\pi \in P(v)$  then
$$x_3 < x_4 < x_1 < x_2.$$
a possible ordering (as far as what seen so far).

#### 26.1.0.6 Input realizing a permutation, by example

- (A) Let  $\pi = (3, 4, 2, 1) = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 4 & 2 & 1 \end{pmatrix}$
- (B) Then the input  $\pi^{-1} = (3, 4, 1, 2) = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 3 & 1 & 2 \end{pmatrix}$
- (C) ... would generate this permutation.
- (D) Formally
$$x_1 = \pi^{-1}(1) = 4 \dots x_i = \pi^{-1}(i) \dots$$

#### 26.1.0.7 Back to sorting...

- (A)  $v$ : a node in decision tree.
- (B) If  $|P(v)| > 1$ : more than one permutation associated with it...
- (C) algorithm must continue performing comparisons
- (D) ...otherwise, not know what to output...
- (E) **Q**: What is the worst running time of algorithm?
- (F) Answer: Longest path from root in the decision tree.
  - ...because we count only comparisons!

#### 26.1.0.8 Lower bound on sorting...

**Lemma 26.1.1.** *Any deterministic sorting algorithm in the comparisons model, must perform  $\Omega(n \log n)$  comparisons.*

Proof

- (A) Algorithm in the comparison model  $\equiv$  a decision tree.
- (B) Use an adversary argument.
- (C) Adversary pick the worse possible input for the algorithm.
- (D) Input is a permutation.
- (E)  $\mathbb{T}$ : the optimal decision tree.
- (F)  $|P(r)| = n!$ , where  $r = \text{root}(\mathbb{T})$ .

### 26.1.0.9 Proof continued...

- (A)  $u, v$ : children of  $r$ .
- (B) Adversary: no commitment on which of the permutations of  $P(r)$  it is using.
- (C) Algorithm perform compares  $x_i$  to  $x_j$  in root...
- (D) Adversary computes  $P(u)$  and  $P(v)$   
[Adversary has infinite computation power!]
- (E) Adversary goes to  $u$  if  $|P(u)| \geq |P(v)|$ , and to  $v$  otherwise.
- (F) Adversary traversal: always pick child with more permutations.
- (G)  $v_1, \dots, v_k$ : path taken by adversary.
- (H) Adversary input:  
The input realizing the single permutation of  $P(v_k)$ .

### 26.1.0.10 Proof continued...

- (A) Note, that

$$1 = |P(v_k)| \geq \frac{|P(v_{k-1})|}{2} \geq \dots \geq \frac{|P(v_1)|}{2^{k-1}}.$$

- (B)  $2^{k-1} \geq |P(v_1)| = n!$ .
- (C)  $k \geq \lg(n!) + 1 = \Omega(n \log n)$ .
- (D) Depth of  $\mathbb{T}$  is  $\Omega(n \log n)$ . ■

## 26.2 Uniqueness

### 26.2.1 Uniqueness

#### 26.2.1.1 Uniqueness

Problem 26.2.1. Given an input of  $n$  real numbers  $x_1, \dots, x_n$ . Decide if all the numbers are unique.

- (A) Intuitively: easier than sorting.
- (B) Can be solved in linear time!
- (C) ...but in a strange computation model.
- (D) Surprisingly...

**Theorem 26.2.2.** *Any deterministic algorithm in the comparison model that solves Uniqueness, has  $\Omega(n \log n)$  running time in the worst case.*

- (E) Different models, different results.

### 26.2.1.2 Uniqueness lower bound

Proof similar but trickier.

$\mathbb{T}$ : decision tree (every node has three children).

**Lemma 26.2.3.**  $v$ : node in decision tree. If  $P(v)$  contains more than one permutation, then there exists two inputs which arrive to  $v$ , where one is unique and other is not.

Proof

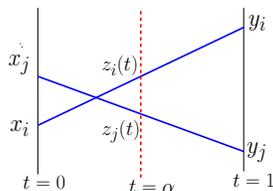
- (A)  $\sigma, \sigma'$ : any two different permutations in  $P(v)$ .
- (B)  $X = x_1, \dots, x_n$  be an input realizing  $\sigma$ .
- (C)  $Y = y_1, \dots, y_n$ : input realizing  $\sigma'$ .
- (D) Let  $Z(t) = (z_1(t), \dots, z_n(t))$  an input where  $z_i(t) = tx_i + (1-t)y_i$ , for  $t \in [0, 1]$ .

#### 26.2.1.3 Proof continued...

- (A)  $Z(t) = (z_1(t), \dots, z_n(t))$  an input where  $z_i(t) = tx_i + (1-t)y_i$ , for  $t \in [0, 1]$ .
- (B)  $Z(0) = (x_1, \dots, x_n)$  and  $Z(1) = (y_1, \dots, y_n)$ .
- (C) Claim:  $\forall t \in [0, 1]$  the input  $Z(t)$  will arrive to the node  $v$  in  $\mathbb{T}$ .

#### 26.2.1.4 Proof of claim...

- (A) Assume false.
- (B) Assume for  $t = \alpha \in [0, 1]$  the input  $Z(t)$  did not get to  $v$  in  $\mathbb{T}$ .
- (C) Assume: compared the  $i$ th to  $j$ th input element, when paths diverted in  $\mathbb{T}$ .
- (D) I.e., Different path in  $\mathbb{T}$  then the one for  $X$  and  $Y$ .
- (E) Claim:  $x_i < x_j$  and  $y_i > y_j$  or  $x_i > x_j$  and  $y_i < y_j$ .
- (F) In either case  $X$  or  $Y$  will not arrive to  $v$  in  $\mathbb{T}$ .
- (G) Consider the functions  $z_i(t)$  and  $z_j(t)$ :

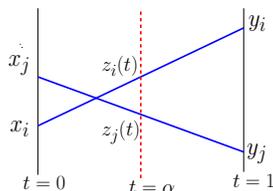


#### 26.2.1.5 Proof of claim continued...

- (A) Ordering between  $z_i(t)$  and  $z_j(t)$  is either ordering between  $x_i$  and  $x_j$  or the ordering between  $y_i$  and  $y_j$ .
- (B) Conclusion:  $\forall t$ : inputs  $Z(t)$  arrive to the same node  $v \in \mathbb{T}$ . ■

#### 26.2.1.6 Back to proof of Lemma...

- (A) Recap:
  - (A) Recall:  $X, Y$  to different permutations that their distinct input arrives to the same node  $v \in \mathbb{T}$ .
  - (B) Proved:  $\forall t \in [0, 1]$ :  $Z(t) = (z_1(t), \dots, z_n(t))$  arrives to same node  $v \in \mathbb{T}$ .
- (B) However: There must be  $\beta \in (0, 1)$  where  $Z(\beta)$  has two numbers equal:



(C)  $Z(\beta)$ : has a pair of numbers that are not unique.

### 26.2.1.7 Proof of Lemma continued...

- (A) Done: Found inputs  $Z(0)$  and  $Z(\beta)$
- (B) such that one is unique and the other is not.
- (C) ... both arrive to  $v$ . ■

Proved the following:

**Lemma 26.2.4.**  *$v$ : node in decision tree. If  $P(v)$  contains more than one permutation, then there exists two inputs which arrive to  $v$ , where one is unique and other is not.*

### 26.2.1.8 Uniqueness takes $\Omega(n \log n)$ time

- (A) Apply the same argument as before.
- (B) If in the decision tree, the adversary arrived to a node...
- (C) containing more than one permutation, it continues into the child with more permutations.
- (D) As in the sorting argument, it follows that there exists a path in  $T$  of length  $\Omega(n \log n)$ .
- (E) We conclude:

**Theorem 26.2.5.** *Solving **Uniqueness** for a set of  $n$  real numbers takes  $\Theta(n \log n)$  time in the comparison model.*

## 26.2.2 Algebraic tree model

### 26.2.2.1 Algebraic tree model

- (A) At each node, allowed to compute a polynomial, and ask for its sign at a certain point
- (B) Example: comparing  $x_i$  to  $x_j$  is equivalent to asking if the polynomial  $x_i - x_j$  is positive/negative/zero).
- (C) One can prove things in this model, but it requires considerably stronger techniques.

**Problem 26.2.6.** (Degenerate points) Given a set  $P$  of  $n$  points in  $\mathbb{R}^d$ , deciding if there are  $d + 1$  points in  $P$  which are co-linear (all lying on a common plane).

- (D) Jeff Erickson and Raimund Seidel: Solving the degenerate points problem requires  $\Omega(n^d)$  time in a “reasonable” model of computation.

## 26.3 3Sum-Hard

### 26.3.1 3Sum-Hard

#### 26.3.1.1 3Sum-Hard

- (A) Consider the following problem:

**Problem 26.3.1.** (3SUM): Given three sets of numbers  $A, B, C$  are there three numbers  $a \in A, b \in B$  and  $c \in C$ , such that  $a + b = c$ .

- (B) One can show...

**Lemma 26.3.2.** *One can solve the 3SUM problem in  $O(n^2)$  time.*

*Proof:* Exercise... ■

### 26.3.1.2 3Sum-Hard continued

- (A) Somewhat surprisingly, no better solution is known.
- (B) Open Problem: Find a subquadratic algorithm for **3SUM**.
- (C) It is widely believed that no such algorithm exists.
- (D) There is a large collection problems that are 3SUM-Hard: if you solve them in subquadratic time, then you can solve 3SUM in subquadratic time.

### 26.3.1.3 3SUM-hard problems

- (A) Those problems include:
  - (a) For  $n$  points in the plane, is there three points that lie on the same line.
  - (b) Given a set of  $n$  triangles in the plane, do they cover the unit square
  - (c) Given two polygons  $P$  and  $Q$  can one translate  $P$  such that it is contained inside  $Q$ ?
- (B) So, how does one prove that a problem is 3SUM hard?
- (C) Reductions.
- (D) Reductions must have subquadratic running time.
- (E) The details are interesting, but are omitted.