CSE520: Computational Geometry
Lecture 5
Topological Lower Bounds II

Antoine Vigneron

Ulsan National Institute of Science and Technology

June 15, 2020

1. Introduction
2. Set disjointness
3. Optimization problems
4. Diameter
5. Maximum gap
6. Discussion

Outline

- In this lecture, we give more example of lower bounds based on Ben-Or's theorem:
  - Set disjointness.
  - Diameter of a point-set.
  - Maximum gap.

References:

- Textbook by Preparata and Shamos.
- Dave Mount's lecture notes, Lecture 26.
- Ben-Or's paper.

Set Disjointness

Problem (Set Disjointness)

Given two real n-tuples $(a_1, \ldots, a_n) \in \mathbb{R}^n$ *and* $(b_1, \ldots, b_n) \in \mathbb{R}^n$*, determine whether there exists a pair* i, j *such that* $a_i = b_j$*.*

Theorem

Any ACT that solves the set disjointness problem has height $\Omega(n \log n)$*.*

- We now prove this theorem. To this end, we introduce the problem below, which is the special case where $(b_1, \ldots, b_n) = (1, \ldots, n)$.
- We denote $\mathbb{N}_n = \{1, \ldots, n\}$.

Problem (1)

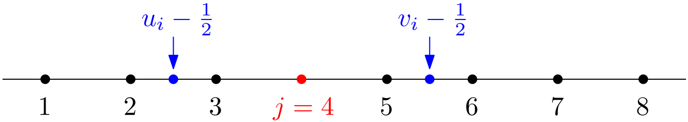
Given $(a_1, \ldots, a_n) \in \mathbb{R}^n$*, decide whether there exists* i *such that* $a_i \in \mathbb{N}_n$*.*

Set Disjointness

- We will give an $\Omega(n \log n)$ lower bound for Problem 1. As it is a special case of set disjointness, it implies the same lower bound for set disjointness. (See argument in previous lecture on line segment intersection detection.)

- Let W^- be the set of *negative* instances of Problem 1.
- So it is the set of tuples $(a_1, \ldots, a_n)$ such that $a_i \notin \mathbb{N}_n$ for all n.
- Let $u : \mathbb{N}_n \to \mathbb{N}_n$ be an arbitrary function from $\mathbb{N}_n$ into $\mathbb{N}_n$.
- Let $a(u) = (u_1 - \frac{1}{2}, u_2 - \frac{1}{2}, \ldots, u_n - \frac{1}{2})$.
- Let $v \neq u$ be another function $v : \mathbb{N}_n \to \mathbb{N}_n$.
- Then $a(u)$ and $a(v)$ are in W^-.
- We now show that they are in different connected components of W^-.

Set Disjointness



- As $u \neq v$, there exists i such that $u_i \neq v_i$.
- So there must be an integer $j \in \{1, \ldots, n-1\}$ between $u_i - \frac{1}{2}$ and $v_i - \frac{1}{2}$.
- So for any path γ from $a(u)$ to $a(v)$, there exists $t \in [0,1]$ such that $\gamma_i(t) = j$.
- Therefore, $\gamma(t) \notin W^-$, so γ is not contained in W^-.
- It follows that $a(u)$ and $a(v)$ lie in different connected components of W^-.

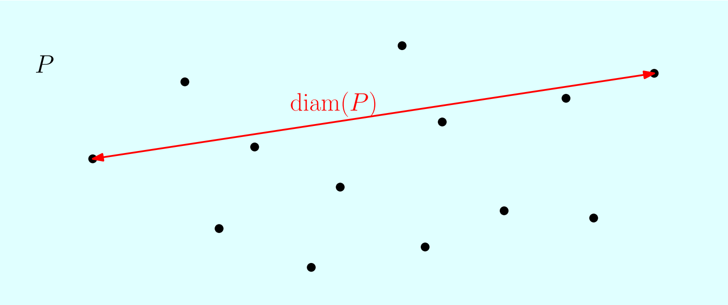
Set Disjointness

- The number of functions $u : \mathbb{N}_n \to \mathbb{N}_n$ is n^n.
- So there are n^n points $a(u)$, and thus n^n connected components in W^-.
- As $\log(n^n) = n \log n$, By Ben-Or's theorem, it follows that:

Lemma

Any algebraic computation tree deciding Problem 1 has height $\Omega(n \log n)$.

Diameter



Problem (Diameter of a Point-Set)

The diameter $\mathrm{diam}(P)$ *of a set* $P = \{p_1, \ldots, p_n\}$ *of* n *points is the maximum distance between any two points:*

$$\mathrm{diam}(P) = \max_{i,j} \mathrm{d}(p_i, p_j).$$

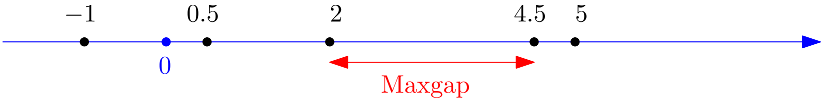
Maximum Gap

Problem (maximum gap)

Given a set of n (unsorted) numbers, the maximum gap problem is to find the largest gap between two consecutive numbers in sorted order.

Example

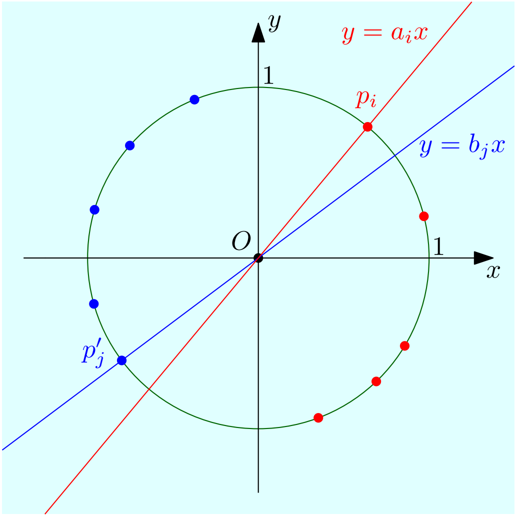
- INPUT: 0.5, 2, 5, 4.5, -1
- OUTPUT: 2.5



Optimization Algorithms

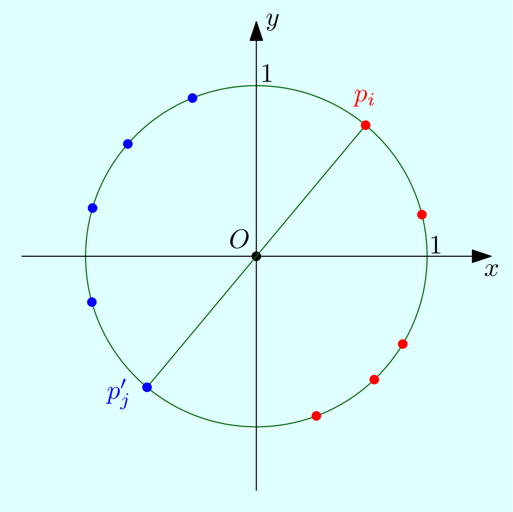
- The ACT model deals with *decision* problems: problems with a YES-NO answer.
- We will abuse notation and use ACTs for *optimization* problems:
  - The 2D-diameter problem.
  - Maximum gap.
- In fact, our lower bound argument will be for the associated decision problems: For some $\delta \in \mathbb{R}$,
  - Is the diameter at least δ?
  - Is the maximum gap at most δ?
- An optimization problem is at least as hard as the corresponding decision problem, because once you have the optimal value, you can compare it with δ in $O(1)$ time.
- So a lower bound on the decision problem implies a lower bound on the optimization problem.

Diameter



Given an instance A, B of set disjointness, we map A to the right side and B to the left side of the unit circle, respectively.

Diameter



The diameter is 2 iff two points p_i, p'_j are diametrically opposed, that is $a_i = b_j$.

Diameter

Theorem

In the ACT model, the complexity of the 2D-diameter problem is $\Omega(n \log n)$*.*

Proof.

An instance of set disjointness is mapped to the unit circle as follows:

- Each a_i is mapped to the point $p_i = (x_i, y_i)$ such that $y_i = a_i x_i$, $x_i > 0$, and $x_i^2 + y_i^2 = 1$.
- Each b_j is mapped to the point $p'_j = (x'_j, y'_j)$ such that $y_j = b_j x_j$, $x_j < 0$, and $x_j^2 + y_j^2 = 1$.

Then the diameter of $\{p_1, \dots, p_n, p'_1, \dots, p'_n\}$ is 2 iff $p_i = -p'_j$ for some i, j, which means that $a_i = b_j$ and hence $A \cap B \neq \emptyset$. $\square$

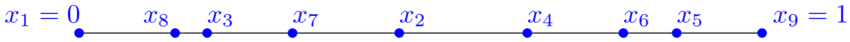
Remark: it is crucial in this argument that the coordinates of p_i and p'_i can be computed in constant time by an ACT.

Maximum Gap

Theorem

In the ACT model, the maximum gap problem has complexity $\Omega(n \log n)$.

- We now prove this theorem.
- Again, we prove it for a special case:



Problem (2)

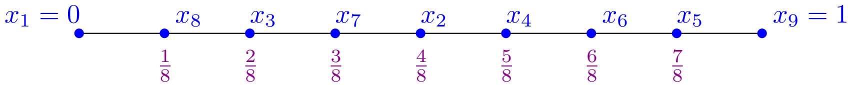
Given a n-tuple $(x_1, \ldots, x_n)$ numbers in $[0, 1]$, such that $x_1 = 0$ and $x_n = 1$, decide whether the maximum gap of X is $\frac{1}{n-1}$.

Maximum Gap

Lemma

Any ACT deciding Problem 2 has height $\Omega(n \log n)$.

- We now prove this lemma.
- What do the positive instances look like?



- In order for the maximum gap to be $1/(n-1)$, all the points must be regularly spaced, with spacing $1/(n-1)$.
- So $(x_2, \ldots, x_{n-1})$ is a permutation of $\left(\frac{1}{n-1}, \frac{2}{n-1}, \ldots, \frac{n-2}{n-1}\right)$.

Maximum Gap

- So W^- is the set of point of $(0, x_2, x_3, \dots, x_{n-1}, 1)$ where $(x_2, \dots, x_{n-1})$ is a permutation of $\left(\frac{1}{n-1}, \frac{2}{n-1}, \dots, \frac{n-2}{n-1}\right)$.
- How many connected components does it have?
- $(n-2)!$ because it is a set of $(n-2)!$ points.
- As $\log((n-2)!) = \Theta((n-2)\log(n-2)) = \Theta(n \log n)$, Problem 1 requires $\Omega(n \log n)$ time in the ACT model.

Maximum Gap

- How fast can we solve maximum gap?
- $O(n \log n)$-time algorithm:

Pseudocode

```
procedure MAXGAP(x_1,...,x_n)
    (y_1,...,y_n) ← Mergesort(x_1,...,x_n)        ▷ O(n log n) time
    M = y_2 − y_1
    for i ← 3, n do                               ▷ O(n) time.
        if y_i − y_{i−1} > M then
            M ← y_i − y_{i−1}
    return M
```

- Conclusion: the complexity of maximum gap in the ACT model is $\Theta(n \log n)$.

Maximum Gap

- However, there is a *linear-time* algorithm.
- Approach: bucketing.

- First we reduce to the case where $\min_i x_i = 1$ and $\max_i x_i = n$.
- So we first compute $m = \min_i x_i$ and $M = \max_i x_i$ in time $O(n)$:

Pseudocode

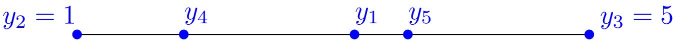
```
procedure MAX(x_1, ..., x_n)
    M ← x_1
    for i ← 2, n do
        if x_i > M then
            M ← x_i
    return M
```

Maximum Gap

- So we replace each x_i with

$$y_i = 1 + (n-1)\frac{x_i - m}{M - m},$$

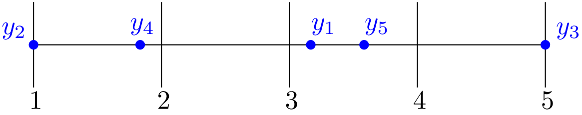
- we solve the maximum gap problem for the $y_1, \dots, y_n$,
- and we multiply the result by $\frac{M-m}{n-1}$



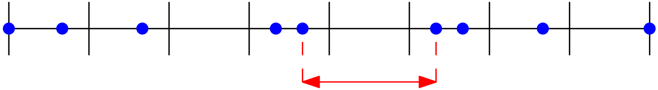
- We now show how compute the maximum gap for the y_i's.

Maximum Gap

- We first make n buckets: $[1,2)$, $[2,3)$, $\ldots$, $[n-1,n)$, $\{n\}$.



- Observation: the maximum gap is at least 1 because the sum of the gap lengths is $n-1$, and there are $n-1$ gaps.
- So the maximum gap is between two points in different buckets.
- Therefore the maximum gap is between the largest number in a bucket and the smallest in the next non-empty bucket.



Maximum Gap

Computing the maximum gap in $O(n)$ time

```
procedure MAXGAPBYBUCKETING(y_1,...,y_n)
    for i ← 1, n do
        store y_i in bucket ⌊y_i⌋
    compute the minimum m_j of each bucket j
    compute the maximum M_j of each bucket j
    result ← 0, last ← M_1
    for j ← 2, n do
        if bucket j is nonempty then
            if m_j − last > result then
                result ← m_j − last
            last ← M_j
    return result
```

Discussion

It would seem that the ACT model is not powerful enough, as the lower bound for maxgap can be broken using the floor function.

- However, it is not reasonable to allow the use of the floor function: It has been proved that the floor function together with $+, -, \times, /$ allows to solve NP-hard problems in polynomial time. (Schönhage, On the power of random access machines, ICALP 79.)

In fact, the ACT model is very powerful:

- It allows *exact* computation in constant time per arithmetic operation.
- Even with high degree polynomials, or square roots.