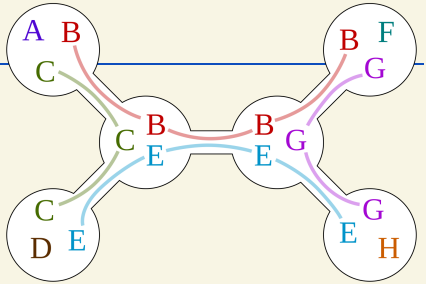
Fine-Grained Complexity

DM898: Parameterized Algorithms
Lars Rohwedder

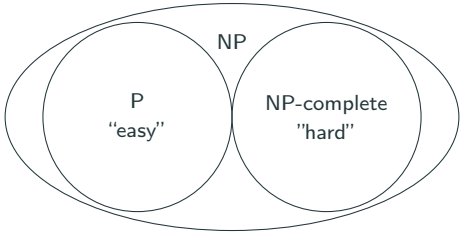


Today's lecture

- Overview Fine-Grained Complexity
- Subquadratic hardness for Orthogonal Vectors
- Subquadratic hardness for Diameter
- Pseudopolynomial time algorithms and hardness for Subset Sum
- Subquadratic hardness for String problems

Context

Classical perspective: Categorize problems into NP-complete and P



This course (thus far): Nuances in complexity of NP-complete problems (FPT, XP, mildly exponential algorithms) and hardness

Now: For polynomial time solvable problems, determine tight running time bounds necessary to solve them (linear time, quadratic time,...)

...and connections between these theories

Fine-grained complexity

- Within P it seems that some problems can be solved faster than others (e.g. $O(n)$ vs $O(n^2)$)
- Classical complexity theory does not give means to prove lower bounds on exponent of polynomial
- On the other hand, reductions that show that a certain running time for some problem implies some running time are straight-forward
- **Fine-grained complexity** is a young research field about establishing lower bounds for many problems using **a low number of** well justified hypotheses
- Research on lower bounds goes hand-in-hand with attempts to improve (non-tight) running times

Some popular hypotheses:

Strong Exponential Time Hypothesis (SETH)

For every $\epsilon > 0$ there exists $k \in \mathbb{N}$ such that k-SAT does not have an $O(2^{(1-\epsilon)n})$ time algorithm

3-SUM Hypothesis (3SUM)

The following problem (3-SUM) cannot be decided in $O(n^{2-\delta})$ time for any $\delta > 0$: Given $A, B, C \subseteq \mathbb{Z}$, $|A| = |B| = |C| = n$, are there $a \in A, b \in B, c \in C$ with $a + b + c = 0$?

All-Pair-Shortest-Path Hypothesis (APSP)

The following problem (APSP) cannot be solved in $O(n^{3-\delta})$ time for any $\delta > 0$: Given an undirected graph with non-negative edge weights find the length of the shortest path between any pair of vertices

Consequences of the SETH

Orthogonal Vectors

Orthogonal Vectors Problem

Input: $A, B \subseteq \{0,1\}^d$ with $|A| = |B| = n$
Output: YES, if there are $a \in A, b \in B$ with $a^\mathsf{T} b = \sum_{i=1}^{d} a_i \cdot b_i = 0$. NO, otherwise

Equivalent formulation: given universe U of size d and two families of sets $\mathcal{X}, \mathcal{Y}$ over U, are there $X \in \mathcal{X}, Y \in \mathcal{Y}$ with $X \cap Y = \emptyset$?

The problem is easily solvable in time $O(n^2 \cdot d)$ or $n \cdot 2^{O(d)}$. Is this optimal?

Theorem

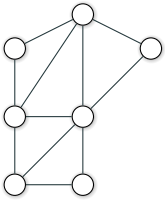
Unless the SETH is false, there is no $n^{2-\delta} d^{O(1)}$ time algorithm for Orthogonal Vectors

Proof: Blackboard

Diameter

Diameter Problem

"Compute the longest shortest path"
Input: a connected graph $G = (V, E)$ and $k \in \mathbb{N}$
Output: YES, if for every $u, v \in V$ there is a path of length at most k between u and v; NO, otherwise



Quadratic time algorithm: For each vertex compute in $O(m)$ time the shortest path to every other vertex using BFS. Then check if any of these lengths is $> k$. Total running time: $O(nm) \leq O(m^2)$

Theorem

Unless the SETH is false, there is no $O(m^{2-\delta})$ time algorithm for the Diameter Problem

Proof: Blackboard

Pseudopolynomial time algorithms for Subset Sum

Subset Sum Problem

Input: $s_1, \ldots, s_n \in \mathbb{N}$, $T \in \mathbb{N}$
Output: YES, if there exist $I \subseteq \{1, \ldots, n\}$ with $\sum_{i \in I} s_i = T$; NO otherwise

The classical dynamic programming algorithm solves the problem in $O(nT)$. Is this optimal?

Theorem (see[1])

Unless the SETH is false, there is no $n^{O(1)} \cdot T^{1-\delta}$ time algorithm for any $\delta > 0$

We do not give a proof of the theorem in this course.

On the other hand, a randomized $(n+T) \cdot \log^{O(1)}(n+T)$ algorithm has since been discovered[2]. Techniques:

- Sumset computation via fast Fourier transform (FFT): for $A, B \subseteq \{0, 1, \ldots, T\}$ we can compute in time $O(T \log T)$ their sumset $A \oplus B = \{a + b : a \in A, b \in B\}$
- Color-Coding

We only show the following weaker result for k-SUM (Subset Sum with the additional constraint $|I| = k$): there is an algorithm that solves k-SUM in time $f(k) \cdot (n+T) \cdot \log^{O(1)}(n+T)$ for some function f. See blackboard

[1] SETH-based lower bounds for subset sum and bicriteria path. Abboud, Bringmann, Hermelin, Shabtay. 2018

[2] A near-linear pseudopolynomial time algorithm for subset sum. Bringmann. 2017

Longest Common Subsequence

Longest Common Substring Problem

Input: strings $a[1]a[2]\cdots a[n]$, $b[1]b[2]\cdots b[n]$ over alphabet Σ, $k \in \mathbb{N}$
Output: YES, if there exists i, j with $a[i]a[i+1]\cdots a[i+k] = b[j]b[j+1]\cdots b[j+k]$; NO, otherwise

A naive algorithm requires $O(n^2k)$ time. Using dynamic programming this can be improved to $O(n^2)$. Using suffix trees even $O(n)$ time is possible (detail omitted).

Longest Common Subsequence Problem

Input: strings $a[1]a[2]\cdots a[n]$, $b[1]b[2]\cdots b[n]$ over alphabet Σ, $k \in \mathbb{N}$
Output: YES, if there exists $i_1 < i_2 < \cdots < i_k, j_1 < j_2 < \cdots < j_k$ with $a[i_1]a[i_2]\cdots a[i_k] = b[j_1]b[j_2]\cdots b[j_k]$; NO, otherwise

Again, dynamic programming can be used to solve the problem in $O(n^2)$ time. Can this be improved as well?

Theorem (see[1] or[2])

Unless the SETH is false, there is no $O(n^{2-\delta})$ time algorithm for Longest Common Subsequence for any $\delta > 0$

We do not give a proof of this theorem in this course.

[1] Quadratic conditional lower bounds for string problems and dynamic time warping. Bringmann, Künnemann. 2015

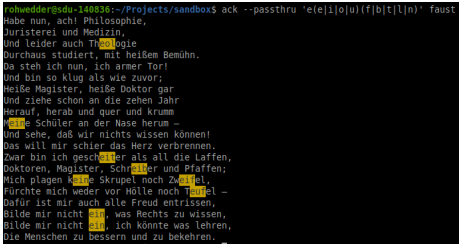
[2] Quadratic-time hardness of LCS and other sequence similarity measures. Abboud, Backurs, Vassilevska Williams

Regular expression pattern matching

Pattern Matching Problem

Input: String $a[1]a[2]\cdots a[n]$ over alphabet Σ, regular expression R

Output: YES, if there is a substring $a[i]a[i+1]\cdots a[j]$ that matches R; NO, otherwise



While general regular expressions are more expressive, we introduce here only a restricted variant for which the hardness already holds. The regular expressions we consider are recursively defined as follows:

- For each $a \in \Sigma$, (a) is a regular expression that matches $b[1]\cdots b[m]$ if and only if $m = 1$ and $b[1] = a$
- For regular expressions R_1, R_2, $(R_1|R_2)$ is a regular expression that matches $b[1]\cdots b[m]$ if and only if R_1 or R_2 matches $b[1]\cdots b[m]$
- For regular expressions R_1, R_2, (R_1R_2) is a regular expression that matches $b[1]\cdots b[m]$ if and only if there is some $1 < j < m$ such that R_1 matches $b[1]\cdots b[j]$ and R_2 matches $b[j+1]\cdots b[m]$

Theorem

Unless the SETH is false, no algorithm solves pattern matching in time $O((n+|R|)^{2-\delta})$ for any $\delta > 0$

Proof: blackboard