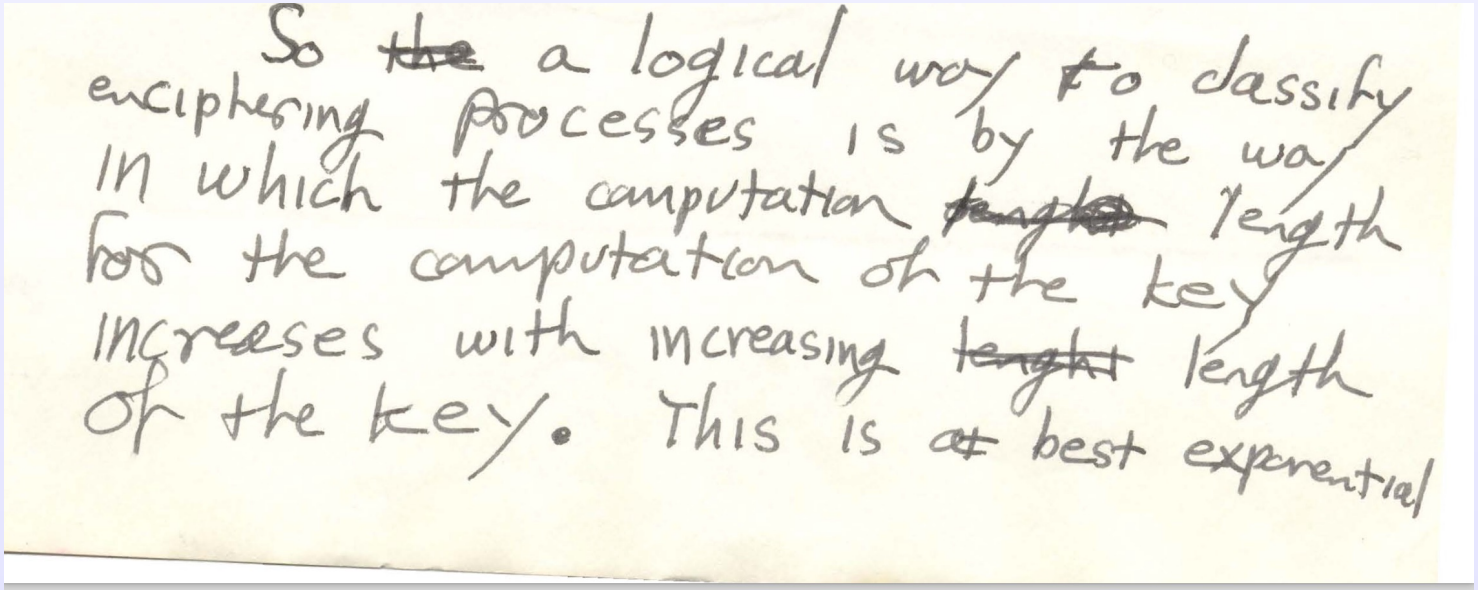
Computational Complexity; slides 1, HT 2019
Introduction, motivation

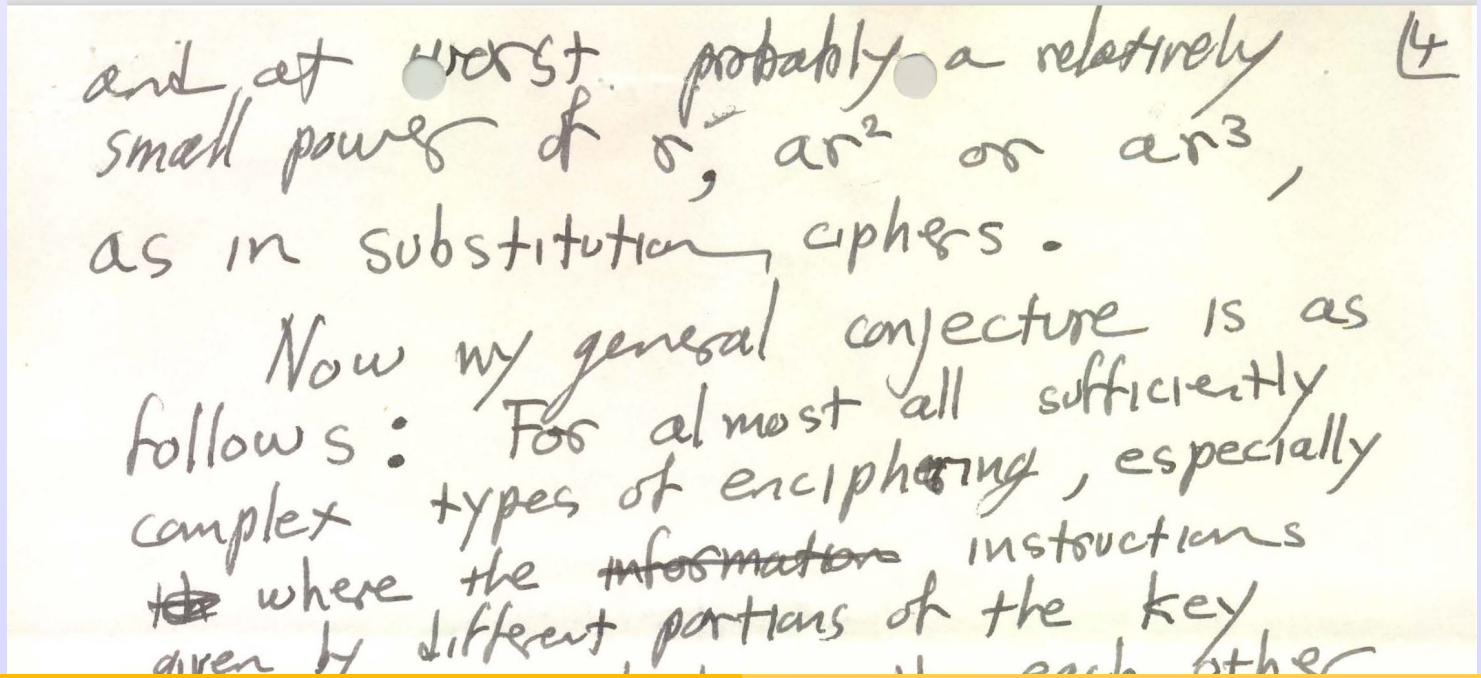
Prof. Paul W. Goldberg (Dept. of Computer Science,
University of Oxford)

HT 2019

Computational complexity, in the beginning



So ~~the~~ a logical way to classify enciphering processes is by the way in which the computation ~~length~~ length for the computation of the key increases with increasing ~~length~~ length of the key. This is at best exponential



and at worst probably a relatively small power of r, ar^2 or ar^3, as in substitution ciphers. (4

Now my general conjecture is as follows: For almost all sufficiently complex types of enciphering, especially ~~the~~ where the ~~information~~ instructions ~~given~~ by different portions of the key

John F. Nash (1928–2015)

National Security Agency (est. 1952)

Correspondence between Nash and NSA, 1955

Organisation

www.cs.ox.ac.uk/people/paul.goldberg/CC/index.html

- Slides, exercise sheets, often updated

www.cs.ox.ac.uk/teaching/courses/2018-2019/complexity/

- General info

Problem sheets: 6 sheets
Available on web page by Monday in weeks 2–7
hand in solutions by Friday 12pm

Class tutors will mark the sheets.
Completed exercise sheets should be left in relevant tutor's pigeon hole.

Lecture Notes: slides for current lecture available on the web page.

Some proofs on whiteboard.

Some terminology

Problem (e.g. SHORTEST PATH, TSP, 3SAT, FACTORING[1])
Instance of a problem; ("yes-instance", "no-instance"),
Complexity (watch out, this has many meanings),
Algorithm
Complexity class (e.g. P, NP, PSPACE),
reduction

Certificate, oracle,...

[1] *Factor Man*, Matt Ginsberg

Aims

(from the web page)

- Introduce the most important complexity classes
- Give you tools to classify problems into appropriate complexity classes
- Enable you to reduce one problem to another

Above terminology to be made precise

We will see there are major gaps in our understanding of computation!

main emphasis on time/space requirements; but cf "communication complexity", "query complexity"

note usage of word "complexity"

Connections with other courses

- Models of Computation:
  - Introduce Turing machines as a universal computing device
  - Classification of problems into decidable/undecidable
  - Refine classification of undecidable problems (degrees of undecidability)
- Introduction to Formal Proof:
  - Logic and proof
- Algorithms (part A)
  - address "intractability" studied here
- Design and Analysis of Algorithms (prelims)
  - design of efficient algorithms.
  - asymptotic complexity analysis of runtime.
- MSc Foundations of Computer Science:
  - Introduces Turing machines and basic complexity

Prerequisites

- Essentially, no particular prerequisites except general mathematical and theoretical computer science skills.
- Basic understanding of analysis of algorithms.
  Big-O notation, (Turing machines)

  See e.g. Chapter 7.1 in Sipser's book (or any other algorithms book)

Not necessary:

- Programming skills
- More advanced methods from algorithm design

Course Structure (roughly)

1. **[2 lectures]** introduction, Turing machines, (un)decidability, reductions
2. **[1 lecture] Deterministic Complexity Classes**. DTIME[t]. Linear Speed-up Theorem. PTime. Polynomial reducibility.
3. **[3 lectures] NP and NP-completeness**. Non-deterministic Turing machines. NTIME[t]. NP. Polynomial time verification. NP-completeness. Cook-Levin Theorem.
4. **[3 lectures] Space complexity and hierarchy theorems**. DSPACE[s]. Linear Space Compression Theorem. PSPACE, NPSPACE. PSPACE = NPSPACE. PSPACE-completeness. Quantified Boolean Formula problem is PSPACE-complete. L, NL and NL-completeness. NL = coNL. Hierarchy theorems.

Course Structure

5. **[1 or 2 lectures] Optimization and approximation**. Combinatorial optimisation problems. approximation schemes.
6. **[2 lectures] Randomized Complexity**. The classes BPP, RP, ZPP. Interactive proof systems: IP = PSPACE.
7. **Advanced topics.** Randomised complexity, Circuit complexity, PCP theorem

Reading List

Primary:

- S. Arora and B. Barak, *Computational Complexity: A Modern Approach*, Cambridge University Press
- M. Sipser, *Introduction to the Theory of Computation*, 2005

Further:

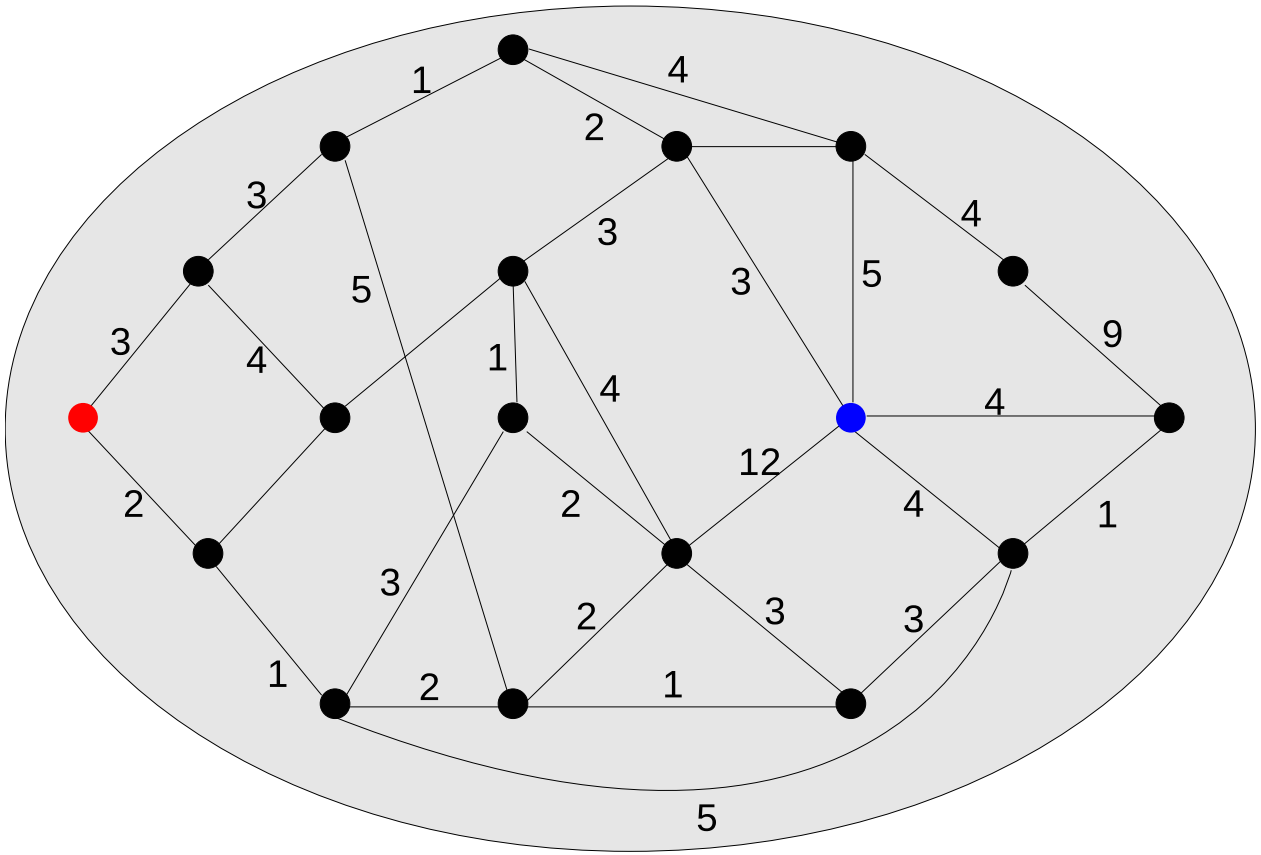
- C.H. Papadimitriou, *Computational Complexity*, 1994.
- I. Wegener, *Complexity Theory*, Springer, 2005.
- O. Goldreich, *Complexity Theory*, CUP, 2008.
- M.R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, 1979.
- T.H. Cormen, S. Clifford, C.E. Leiserson and R. L. Rivest, *Introduction to Algorithms*, 2001.

Problem: Finding Paths – Shortest Path

Problem. (Shortest Path)

Given a weighted graph and two vertices s, t, find a shortest path between s and t.

Can be solved efficiently (for instance with Dijkstra's algorithm)

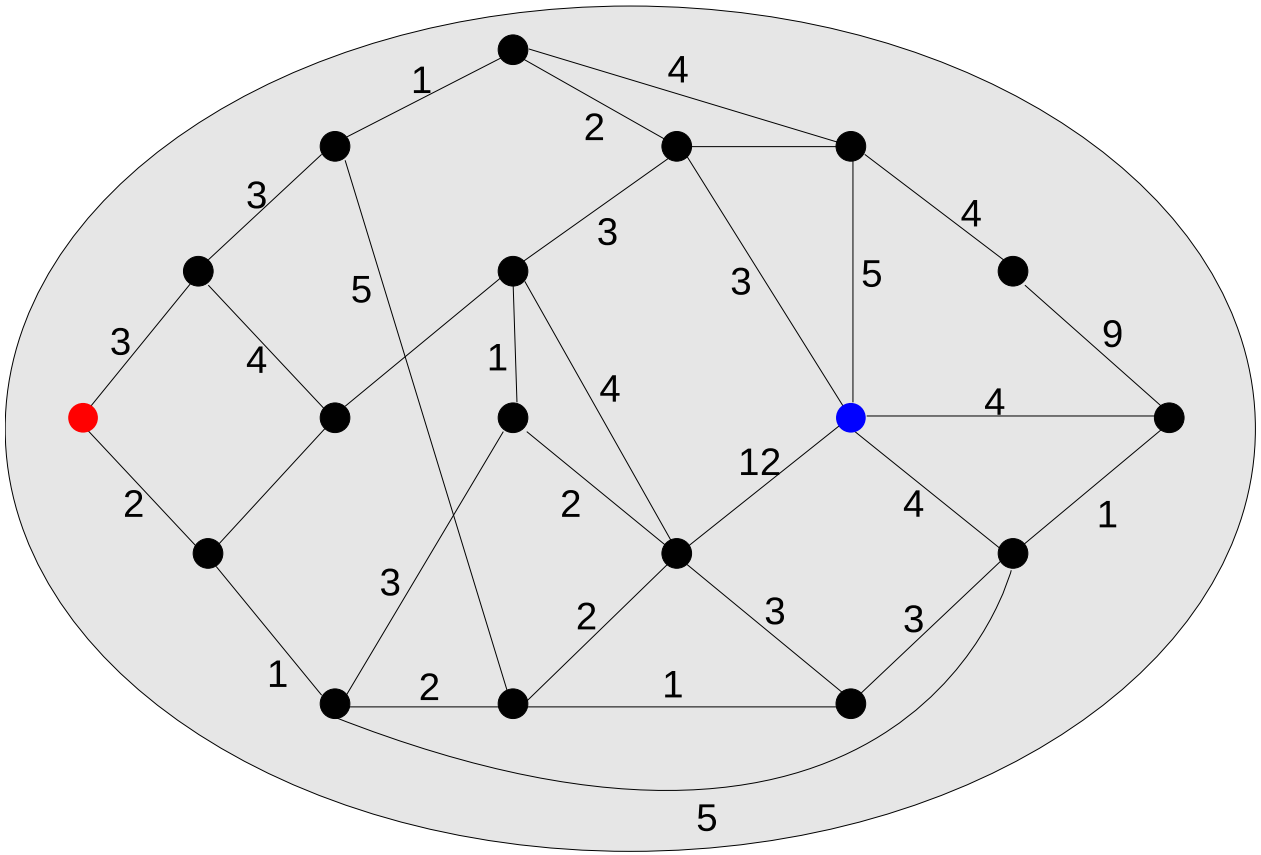


Problem: Finding Paths – Longest Path

Problem. (Longest Path)

Given a weighted graph and two vertices s, t, find a longest simple (cycle-free) path between s and t.

No efficient solution known (and conjectured not to exist)

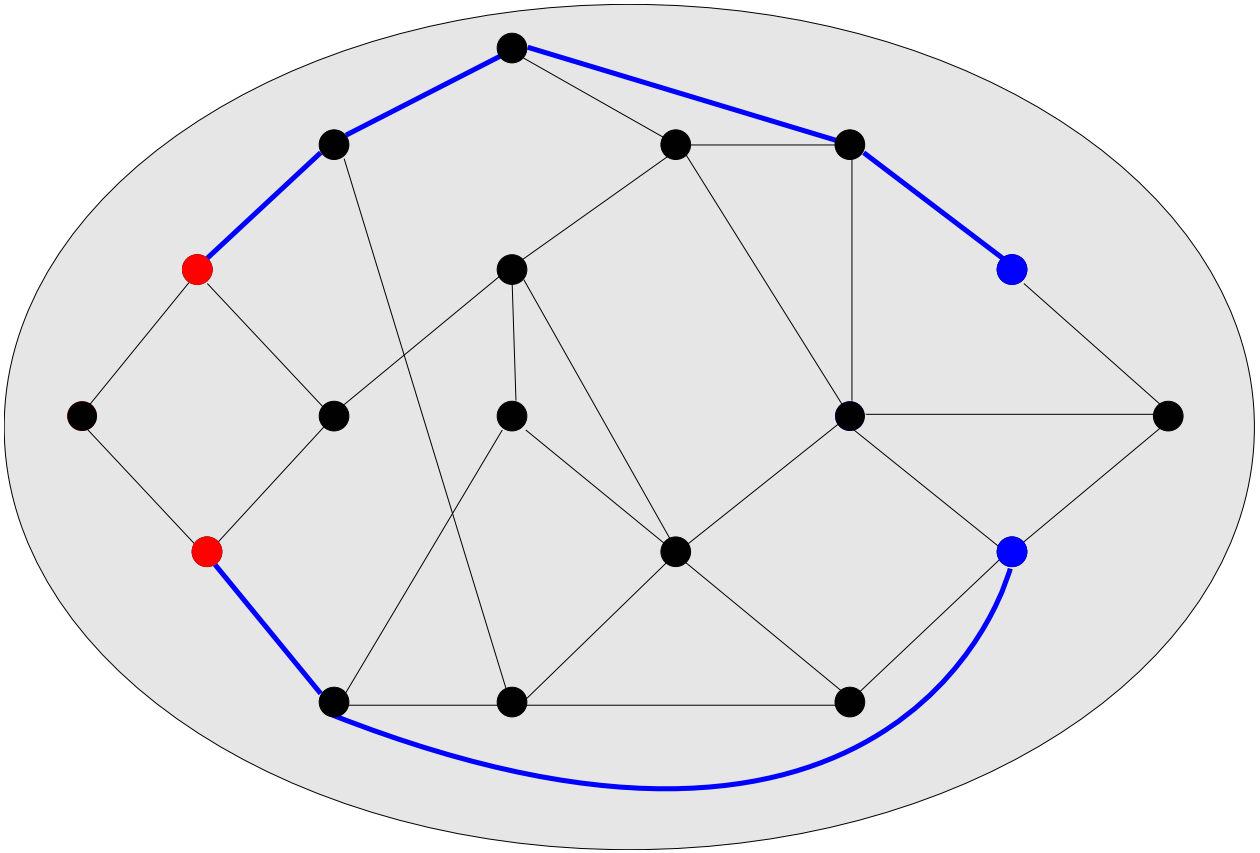


Problem: $X-Y$-Disjoint Paths

Problem. ($X-Y$-disjoint paths)

Given a graph, two sets X, Y of vertices and $k \in \mathbb{N}$, find k disjoint paths between vertices in X and Y.

Can be solved efficiently using network flow techniques

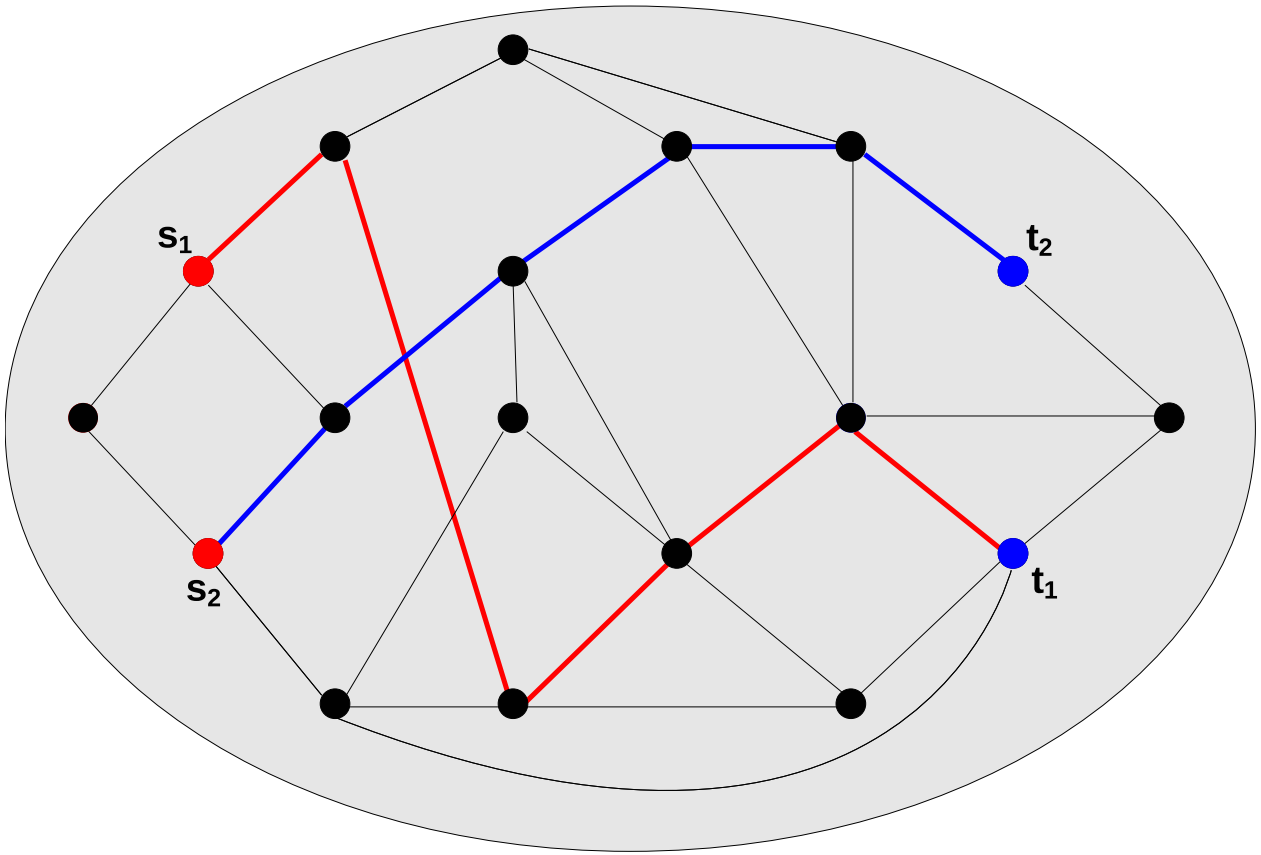


Problem: Disjoint Paths

Problem. (disjoint paths)

Given a graph, two tuples $X := (s_1, \ldots, s_k), Y := (t_1, \ldots, t_k)$ of vertices, find disjoint paths linking s_i, t_i, for all i.

No efficient solution known (and conjectured not to exist)

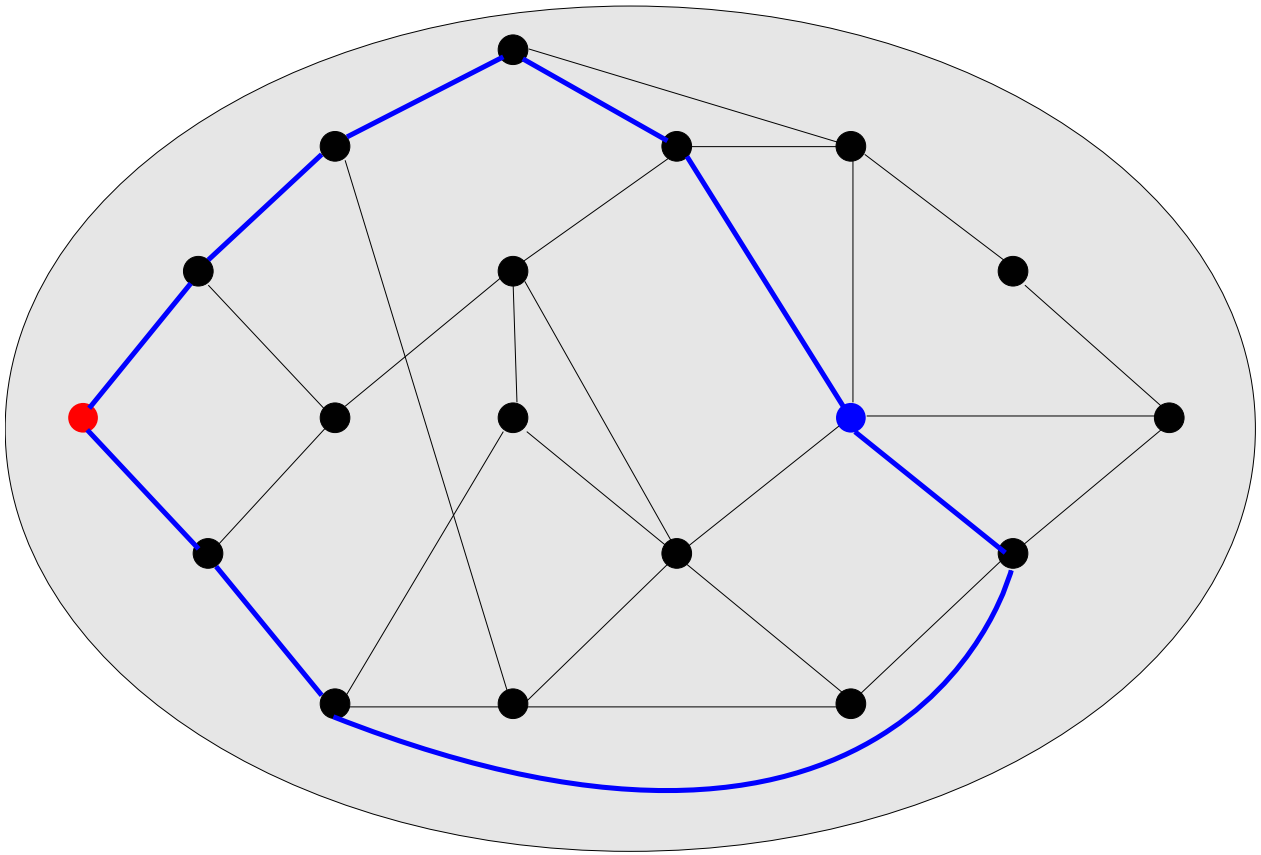


Problem: Length Constrained Disjoint Paths

Problem. (Length constrained disjoint paths)

Given a graph, two vertices s, t and $c, k \in \mathbb{N}$, find k disjoint paths between s and t of length $\leq c$.

No efficient solution known (and conjectured not to exist)



Problem: Solving Linear Equations

Problem. (solving equations)

Given a system of linear equations, check whether it has an integer solution.

$$\begin{aligned} x + y &= 2 \\ y - 3z &= 5 \end{aligned}$$

No efficient solution known (and conjectured not to exist)

Problem: Solving Equations

Problem. (solving equations)

Given a system of arbitrary equations with a polynomial on the left-hand-side, check whether it has an integer solution.

$$\begin{aligned} xyz - y^3 + z^2 &= 2 \\ y - 3z &= 5 \end{aligned}$$

No algorithmic solution exists at all !!

Motivation

Questions.

- Why are some problems so much harder to solve than other – seemingly very similar – problems?
- Are they really harder to solve?

  Or have we just not found the right method to do so?

Complexity Theory:

The aim of computational complexity theory is to classify problems according to the amount of resources needed for their computation.

- Classify problems into classes of problems which are of the same "difficulty".
- Provide methods to establish the complexity of a problem.

Graph Problems: Clique

Graphs:

- Graphs in this lecture will be finite, undirected, and simple.
- The order of a graph is the number of its vertices.

Clique: A clique is a complete graph.

A clique in a graph G is a complete subgraph of G.

Opt-Clique

Input: Graph G

Problem: Find clique $C \subseteq G$ of maximal order.

Optimisation problem: Find a maximum clique in a graph.

Applications: In chemistry, psychology, ...

Algorithms

Simple algorithm for clique:

Given: Graph G of order $n := |G|$
Problem: Find a clique $C \subseteq G$ of maximal order.

for $k := n$ **to** 1 **do**
  for all $X \subseteq V(G), |X| = k$ **do**
    if X induces a complete subgraph **then** output X.

Running time:

Analyse the asymptotic worst-case complexity of the algorithm

$$\mathcal{O}(\sum_{k=1}^{n} n^k \cdot k^2) = \mathcal{O}(n^{n+3}) = 2^{\mathcal{O}(n \cdot \log n)}$$