Limits of Computation

16 - Problems in **P**

Bernhard Reus

The complexity story so far

sequential ones

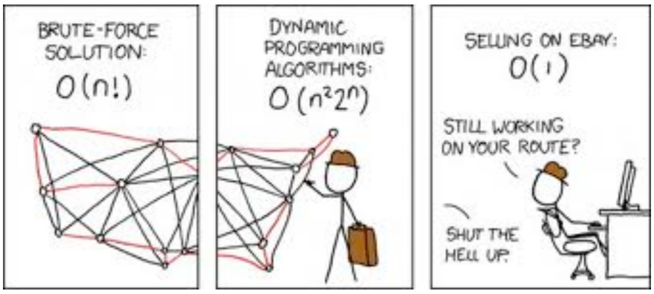
- **P** is robust under compilation between any machines/languages (**LIN** only between some of them)
- Hierarchy theorems: there exist problems that can only be solved if more running time is available.

Complexity of natural problems

THIS TIME

- we introduce some natural (and famous) problems and discuss their time complexity.
- The problems in this session are all provably in **P**.
- For others we don't know and the question remains: "are they feasible?" (see next session)

e.g. finding the best route for Travelling Salesman (next session!)



http://xkcd.com

Important sample Problems

- We introduce a couple of nice problems, all highly relevant in practice.
- Optimisation problems:
  "find the shortest/ largest ..."
  we present as **decision problems**:
  "is there a ... of size *K*?"

requires a measure function m that is computable in polynomial time

- This is simpler, fits our definition of problem, and does not simplify the complexity as we now see:

Decision Problem template

Problem P of the form "decide membership in A"

- **Instance**: some word (data) d
- **Question**: Is d in A?
- to show that P is in $\mathbf{TIME}(f(n))$ one needs to find a decision procedure for A with
  - **Input:** d
  - **Output**: 'yes' or 'no' stating whether d is in A.
  - **Runtime bound**: f

Recall that input is always encoded as word over alphabet $\{0,1\}$

Optimisation Problem template

Problem P of the form "find optimal s in G"

- **Instance**: some "scenario" G (often a graph)
- **Question**: find a solution s such that *m(s)* is minimal/maximal where *m* is some (fixed) measure of the size of s.
- to show that P is in $\mathbf{TIME}(f(n))$ one needs to find an algorithm with
- **Input**: G
- **Output**: s
- **Runtime bound**: f

Reduce Decision to Optimisation

- **Instance**: some "scenario" G (often a graph)
- **Question**: find solution *s* for G such that *m(s)* is minimal/maximal where *m* is some (fixed) measure of the size of *s*.

- **Instance**: some "scenario" G (often a graph) **and a positive number *K***
- **Question**: **is there** a **solution** *s* for G **such that *m(s)* ≦ *K*?** (or *m(s)* ≧ K, resp.)?

Why is this a reduction? Why is this sufficient for our purposes?

Why this reduction?

- computing the size of the solution (with function m) is supposed to be computable in polynomial time.
- so due to this reduction, if the optimisation problem is in $\mathbf{P}$, the decision problem must be too.

LIMITS!

- if the decision problem is not in $\mathbf{P}$, then the optimisation problem cannot be.

Problems in **P**

"tractable" by Cook-Karp

Algorithm	Time Complexity		
	Best	Average	**Worst**
Quicksort	Ω(n log(n))	Θ(n log(n))	O(n^2)
Mergesort	Ω(n log(n))	Θ(n log(n))	O(n log(n))
Timsort	Ω(n)	Θ(n log(n))	O(n log(n))
Heapsort	Ω(n log(n))	Θ(n log(n))	O(n log(n))
Bubble Sort	Ω(n)	Θ(n^2)	O(n^2)
Insertion Sort	Ω(n)	Θ(n^2)	O(n^2)
Selection Sort	Ω(n^2)	Θ(n^2)	O(n^2)
Tree Sort	Ω(n log(n))	Θ(n log(n))	O(n^2)
Shell Sort	Ω(n log(n))	Θ(n(log(n))^2)	O(n(log(n))^2)
Bucket Sort	Ω(n+k)	Θ(n+k)	O(n^2)
Radix Sort	Ω(nk)	Θ(nk)	O(nk)
Counting Sort	Ω(n+k)	Θ(n+k)	O(n+k)
Cubesort	Ω(n)	Θ(n log(n))	O(n log(n))

Array Sorting

you should know this already! :-)

- **Instance**: a number array A
- **Question**: What is A sorted?

k = number of bins/length of data/ range of data, resp. [in binary]

Membership test for context-free languages

- we know that context-free languages (generated by context free grammars) are decidable (there is a parser).
- But what is the time complexity of parsing ?
- **Instance**: a context free language L over alphabet A and a word *s* over alphabet A
- **Question**: is *s* in L?

simple minded algorithm needs to test all possible derivations but there are exponentially many.

Dynamic Programming
use solutions to subproblems you already have; in this case produce a parsing table.
Runs in $O(|s|^3)$

CockeYoungerKasami (CYK) algorithm

Is a number a prime?

- **Instance**: a natural number *n*
- **Question**: is *n* a prime number ?
- Using *binary* representation of numbers to measure (*logarithmic*) size of input.
- That there is an algorithm with polynomial time bound (in this sense) has only be shown in 2002 in a famous (award-winning) result by:
  "PRIMES is in P": Agrawal, Kayal, Saxena (AKS)

Graph (Optimisation) Problems

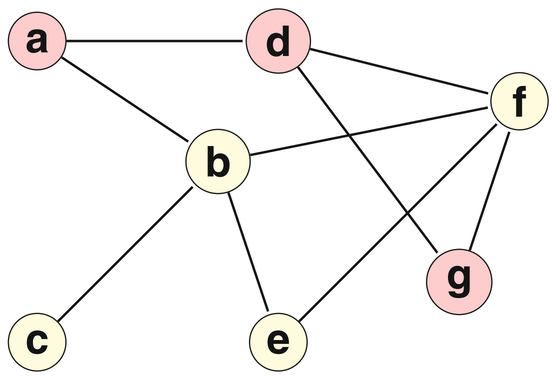
(Graph) Reachability

also called Graph Accessibility Problem

- Given a graph G=(V,E) with (un)directed edges E, two nodes *s* and *t*:
- Is there a path p from node *s* to node t in G?

Simple breath-first or depth-first graph traversal starting from s can be done in linear time, O(|E|)

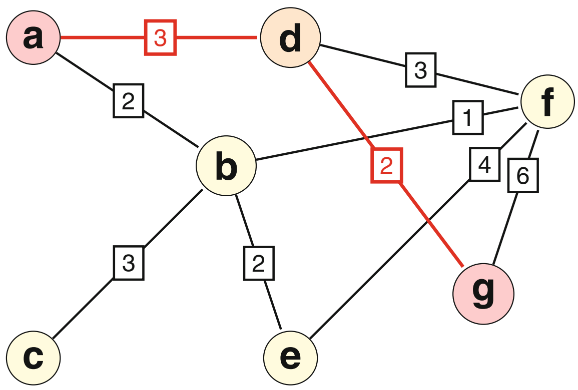
Is there a path from ***a*** to ***g***?



path: a - d - g *or* a - b - f - g

Shortest Path

- **Instance**: a weighted graph G=(V,E,w) with weighted edges E, and two vertices s and t
- **Question**: What is (the length of) the shortest path from s to t in G?



"Floyd-Warshall-algorithm"

"depth first search" F-W algorithm has runtime O(|V|³)

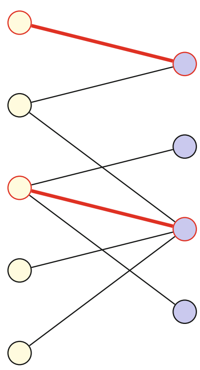
path *a* - *d* - *g* with length 3+2=5

issues with negative weights

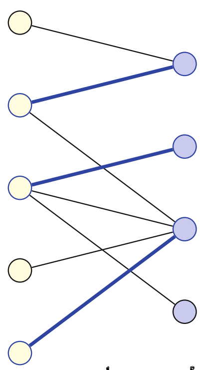
Maximal Matchings

- a *matching* in a graph is a set of edges such that no two edges in the set share a common vertex.
- **Instance**: a graph G=(V,E)
- **Question**: What is the largest matching in G?

matching of size 2



matching of size 3



largest

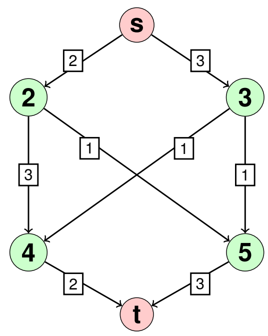
"Blossom-algorithm" by Edmonds

"depth first search" Edmonds's "Blossom" algorithm has runtime O(|E|*|V|²) but there are better ones – O(|E|*|V|⁰·⁵)

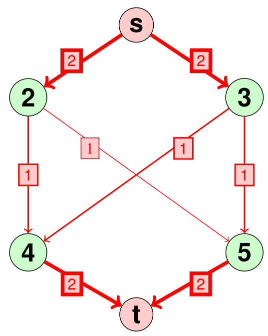
Max-Flow / Min-Cut

- **Instance**: a weighted directed graph G=(V,E,w) encoding a flow network, source node s, sink node t.
- **Question**: What is the *maximum flow (or cut with minimum capacity)* in the given network G?

flow network



max flow in red



max. total flow of 4

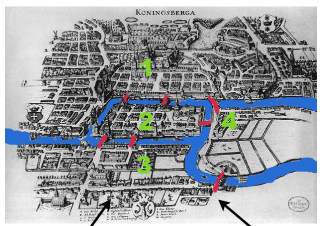
"Ford-Fulkerson-algorithm"

Ford-Fulkerson algorithm has runtime O(|E| x maxflow)

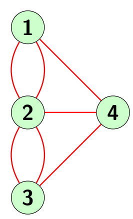
The 7 Bridges of Königsberg

- *Frederick the Great* wanted to show off the 7 bridges to visiting dignitaries

- and asked famous (Swiss-born) mathematician *Leonhard Euler (1707-1783)* for a tour that visits each bridge exactly once.
- Euler used a graph (inventing graph theory); he had to report back that this is impossible (only possible on certain condition discussed in exercises).



river "Pregel" bridges



abstract graph of bridges=edges, nodes=land

find a tour of requested kind among the 7 bridges ⇨ find a *Eulerian* circuit in the graph

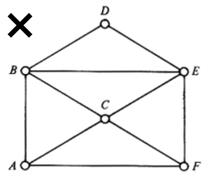
circuit = closed path with no repeating edges

The Postman Problem

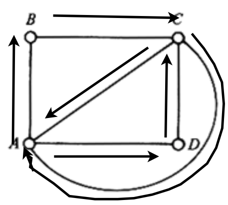
- deliver the mail in your neighbourhood where edges are streets (= undirected graph); start in A;
- you want to visit all streets and return to A without visiting a street twice.
- **Instance**: a graph G=(V,E)
- **Question**: Is there an (Eulerian) circuit in G that visits *every edge* (=street) *once* ?

also "Chinese" postman problem due to Chinese author Kwan Mei-Ko

×



√



http://web.mit.edu/urban_or_book/www/book/chapter6/6.4.2.html

like 7 bridges problem = find Eulerian circuit

Route Inspection Problem: replace "once" by at least "once"

Fleury's elegant algorithm has runtime $O(|E|^2)$ but there are faster ones: Hierholzer's $O(|E|)$

Linear Programming

very versatile & useful

- Solving linear inequalities
- **Instance**: a vector of positive (real) variables x, row vector b, matrix A such that $Ax \leqq b$ and column vector c
- **Question**: maximise $c^T x$

Simplex Algorithm

maximise $P_1 \times x_1 + P_2 \times x_2$
where
$$x_1 + x_2 \leq A$$
$$F_1 \times x_1 + F_2 \times x_2 \leq F$$
$$I_1 \times x_1 + I_2 \times x_2 \leq I$$

$$\mathbf{b} = \begin{pmatrix} A \\ F \\ I \end{pmatrix} \qquad A = \begin{pmatrix} 1 & 1 \\ F_1 & F_2 \\ I_1 & I_2 \end{pmatrix}$$

$$\mathbf{c} = (P_1 \;\; P_2)$$

Simplex algorithm does not have polynomial runtime (!) Karmarkar gave polynomial algorithm $O(n^{3.5} \times L^2 \times \ln L \times \ln \ln L)$ where n is the number of variables and L is size of input (binary).

Problems in $\mathbf{P}$

- Sorting
- Parsing
- Prime Number Test
- Graph Reachability
- Shortest Path
- Maximal Matching
- Max Flow-Min Cut
- Postman Problem
- Linear Programming

and many, many more

END

Next time:
More practically relevant problems for which **no** polynomial time algorithms are known.