Algorithm Theory

Georg Moser Mircea Dan Hernest

Institute of Computer Science @ UIBK

Summer 2007

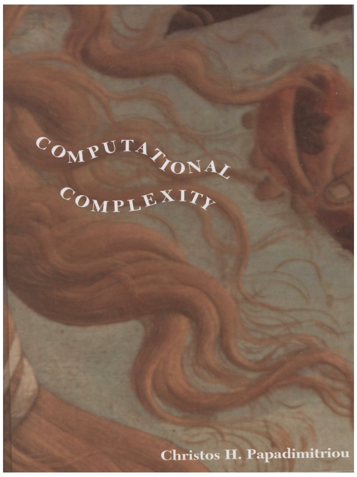
Schedule

week 1	March 5	week 9	May 14
week 2	March 12	week 10	May 21
week 3	March 19	week 11	June 4
week 4	March 26	week 12	June 11
week 5	April 16	week 13	June 18
week 6	April 23	week 14	June 25
week 7	April 30	first exam	July 2
week 8	May 7		

Literature & Online Material

Literature

Papadimitriou, Christo, Computational Complexity (Addison-Wesley, 1994)



Online Material

Transparencies and homework are available from IP starting with 138.232; solution to selected exercises will be available online after they have been discussed.

Exams and Exercises

- lecture is a VU, i.e., "Vorlesung" and "Übung" are combined
- we offer 3 exercise groups
- mid-term test (45 min) on May 4 (covers the material of first 7 weeks)
- let E denote the exam result, T the test result; the final grade is computed as

$$\max\{E, \lceil \frac{2}{3} \cdot E + \frac{1}{3} \cdot T \rceil\}$$

Exercise Groups

UE	Group 1	Friday 12.15-13.00, SR 12	Georg Moser
	Group 2	Friday 12.15-13.00, HS 10	Dan Hernest
	Group 3	Friday 13.15-14.00, HS 10	Dan Hernest

Content

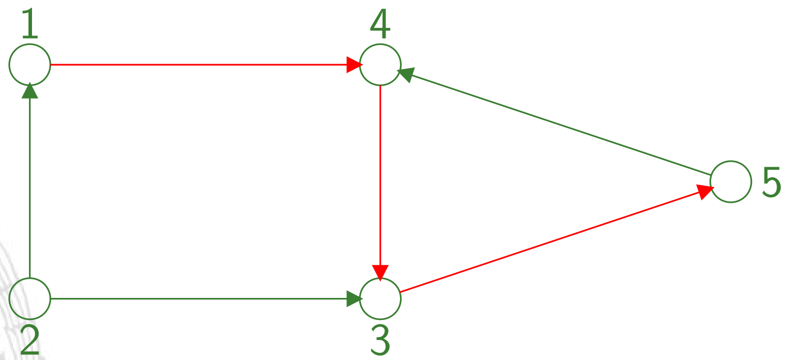
Problem: REACHABILITY

A (directed) graph $G = (V, E)$ is a finite set V of nodes and a set E of edges, which are pairs of nodes.

Problem

Given a graph G and nodes $1, n \in V$, is there a path between 1 and n? This problem is called REACHABILITY.

Example



Algorithm

- mark 1, set $S := \{1\}$
- Choose $i \in S$, remove i from S
  - For all $(i, j) \in E$ and j unmarked, mark j, add j to S
- Iterate till S is empty.
- Answer "yes" if n is marked, otherwise "no"

Fact: The (time) complexity of the search algorithm is $\mathcal{O}(n^2)$; search can be depth-first or breadth-first.

Definition

f, g functions from $\mathbb{N}$ to $\mathbb{N}$.

- $f(n) = \mathcal{O}(g(n))$, $\exists c, n_0 \geq 1\ \forall n \geq n_0\ \ (f(n) \leq c \cdot g(n))$
- $f(n) = \Omega(g(n))$, if $g(n) = \mathcal{O}(f(n))$
- $f(n) = \Theta(g(n))$, if $f(n) = \mathcal{O}(g(n))$ and $f(n) = \Omega(g(n))$

Worst-Case Analysis

We deal with growth rates only and regard polynomial growth rates as acceptable, while exponential growth rates are intractable.

Only worst-case analysis; average case analysis would be better, but

- what is the input distribution of a problem?
- what happens if we are interested in the worst-case?

Motto

Adopting polynomial worst-case performance as our criterion of efficiency results in an elegant and useful theory that says something meaningful about practical computation, and would be impossible without this simplification.

Networks

Fact: The space requirements of the Dijkstra algorithm is $\mathcal{O}(n)$. Can be improved to $\mathcal{O}((\log n)^2)$.

- a network $N = (V, E, s, t, c)$ is a graph with source s and sink t
- if $(i, j) \in E$, then $c(i, j) > 0$ is the capacity
- a flow f assigns non-negative integers to edges, s.t. $f(i, j) \leq c(i, j)$
- for each node j (except s, t)

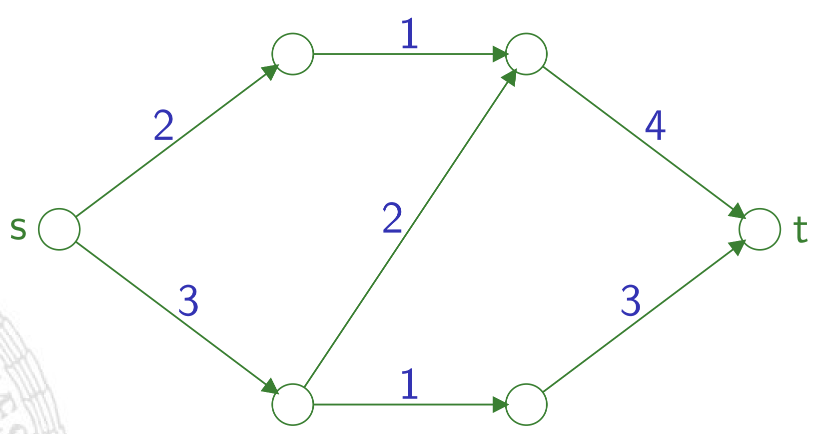
$$\sum_{(i,j)\in E} f(i,j) = \sum_{(j,k)\in E} f(j,k) \ .$$

- the value of the network is $\sum_{(s,s')\in E} f(s, s')$

Problem: MAX FLOW

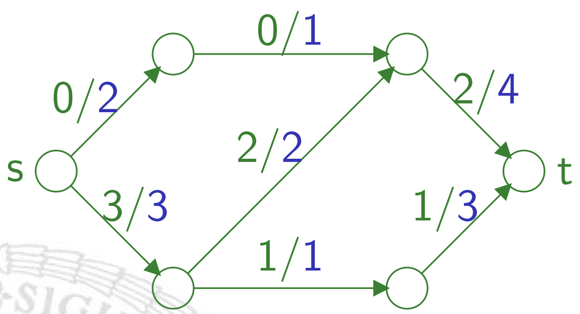
- MAX FLOW is the problem to find the flow with the largest value
- MAX FLOW(D) is the related decision problem
- MAX FLOW and MAX FLOW(D) are polynomial equivalent

Example

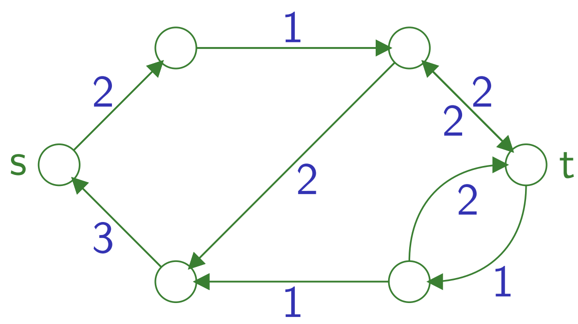


we define the derived network $N(f) = (V, E', s, t, c')$

$$\begin{aligned} E' &:= E - \{(i,j) \mid f(i,j) = c(i,j)\} \cup \\ &\quad \cup \{(j,i) \mid (i,j) \in E \text{ and } f(i,j) > 0\} \\ c'(i,j) &:= c(i,j) - f(i,j) \quad \text{for old edges} \\ c'(i,j) &:= f(j,i) \quad \text{for new edges} \end{aligned}$$



network N



derived network $N(f)$

- ➡ assume there exists f' with f' greater than f: $\Delta f = f' - f$ is positive flow in $N(f)$

Algorithm

1. start with zero-flow in N.
2. construct $N(f)$
   - ➡ if there is a path from s to t:
     find the smallest capacity c' on the path and add it to the flow
   - ➡ employ algorithm for REACHABILITY
   - ➡ Repeat, until no path can be found

Complexity

- ➡ at most $n \cdot C$ iterations, where C is the maximal capacity
- ➡ implies time complexity: $\mathcal{O}(n^2 \cdot nC) = \mathcal{O}(n^3 C)$
- ➡ DANGER: C depends exponential on any succinct representation of the input
- ➡ more thought (i.e. using the shortest path) yields $\mathcal{O}(n^5)$
- ➡ reducible to $\mathcal{O}(n^3)$

TSP

- given n cities, with positive distance d_{ij}, s.t. $d_{ij} = d_{ji}$
- what is the fastest tour of the cities, i.e. minimise

$$\sum_{i=1}^{n} d_{\pi(i),\pi(i+1)} \qquad \text{for permutation } \pi \text{ with } \pi(n+1) = \pi(1)$$

- the problem is called TSP
- the (polynomially) related decision problem: TSP(D)

Naive Algorithm

- enumerate all possible solutions; compute the costs; pick the best

Fact: Time bound: $\mathcal{O}(n!)$, Space bound: $\mathcal{O}(n)$
This bound can be improved slightly, but remains exponential

P vs NP

Definition (informal)

1. the complexity class **P** contains all feasible problems
2. **NP** contains all problems that are feasible on a machine that can guess

we will see that TSP can be solved in polynomial time if we allow a non-deterministic algorithm

- no clever way of removing non-determinism is known
- in fact if you find a polynomial-time algorithm you can win $1 million:

  The Board of Directors of CMI [Clay Mathematics Institute] designated a $1 million prize fund for the solution to this problem.
- latest conjecture: **P** $\neq$ **NP** proven in 2050 (Natarajan Shankar)