Prerequisites

COMP6741: Algorithms for Intractable Problems

Serge Gaspers

Abstract

This document outlines the prerequisite knowledge that students should have when studying COMP6741: Algorithms for Intractable Problems at UNSW.

1 Proof techniques

Students are expected to be familiar with short proofs using the following basic proof techniques.

- direct proof
- proof by induction
- proof by contradiction
- proof by contraposition

Examples of these proof techniques are found in Appendix A

2 Algorithm design techniques

You are expected to be familiar with the following algorithm design techniques:

- greedy algorithms,
- dynamic programming algorithms,
- recursive algorithms, including divide-and-conquer.

It is an advantage to know about branching/backtracking algorithms and randomized algorithms.

3 Analysis of algorithms

You are expected to be familiar with the analysis of worst-case time and space requirements of algorithms using asymptotic notation, such as $O, o, \Omega, \omega, \Theta, \sim$.
You should be comfortable manipulating logarithmic and exponential functions.
Stirling's approximation is useful to know:

$$n! \sim \sqrt{2\pi n} \cdot (n/e)^n.$$

We can approximate binomial coefficients using Stirling's approximation:

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} \sim \sqrt{\frac{n}{2\pi k(n-k)}} \cdot \frac{n^n}{k^k \cdot (n-k)^{n-k}}$$

The binomial theorem states that, for each natural number $n \in \mathbb{N}_0$,

$$(x+y)^n = \sum_{k=0}^{n} \binom{n}{k} x^{n-k} y^k.$$

4 Models of computation

It is an advantage to know about various models of computation, especially the Turing machine, the RAM (random access machine), and the word RAM.

5 Graphs

You are expected to be familiar with basic notions of graph theory, including

- spanning trees,
- matchings.

6 Computational problems

It is an advantage to know about

- decision problems (*where the answer is either* YES *or* NO)
  Example: Given a graph G and an integer k, does G have a vertex cover of size at most k
  Definition: A *vertex cover* is a vertex subset $S \subseteq V$ of a graph $G = (V, E)$ such that for each edge $uv \in E$, we have that $u \in S$ or $v \in S$
- optimisation problems (*where the answer is the value of the objective function of a minimisation or maximisation problem*)
  Example: Given a graph G, return the size of a smallest vertex cover of G
- function problems or search problems (where the answer is an arbitrary function of the input)
  Example: Given a graph G, return a smallest vertex cover of G.

7 Complexity

You are expected to know about the most basic complexity classes P and NP, and about NP-hardness and NP-completeness.

It is an advantage to have designed NP-completeness proofs.

Towards the start of the course, we will revisit the theory of NP-completeness and design some NP-completeness proofs.

8 Programming languages

It is an advantage to be familiar with Python (for a group assignment).

9 Mathematical writing and grades

It is expected that you are able to express your proofs, algorithms, answers to questions, and your creative problem solving in a way that a typical fellow student enrolled in this class is able to

- understand your proof/algorithm/answer/solution to the problem/question, and
- that you demonstrated that you also understood this proof/algorithm/answer/solution.

In cases where your attempted solution strategy does not work out, it is an advantage (in terms of grades) if you explicitly state that your strategy did not work out, rather than pretending that your erroneous strategy is an actual proof/solution. The realization that a "proof" has gaps is a much more valuable insight than a misrepresentation where you attempt to mislead. Grades will take into account how "close" you were to an answer to the question. While mathematical statements are either true or false, and there are no intermediate degrees of trueness and falseness, we will attempt to look for how many additional "ingredients" would have been needed for your answer to be correct when assigning grades.

A Examples of proof techniques

A.1 Direct proof

where we derive the statement by logically combining axioms, definitions, and earlier proven statements.

In a graph G, we denote the degree of a vertex v by $d_G(v)$. We omit the subscript G when it is clear from the context. For a natural number r, the graph G is *r-regular* if each vertex has degree r.

Proposition 1. *Let r be an odd natural number and let G be an r-regular graph. The graph G has an even number of vertices.*

Proof. By the Handshake lemma, $\sum_{v\in V} d(v) = 2 \cdot |E|$.[1] Denote by n the number of vertices and by m the number of edges of G. We have that

$$\begin{aligned} m &= \frac{\sum_{v\in V} d(v)}{2} && \text{(Handshake lemma)} \\ &= \frac{\sum_{v\in V} r}{2} && \text{(Every vertex has degree } r\text{)} \\ &= \frac{n \cdot r}{2} && \text{(There are } n \text{ vertices in } G\text{)} \end{aligned}$$

Since $m = nr/2$ is an integer, either n or r is divisible by 2. Since r is odd, n must be even. □

A.2 Proof by induction

where we derive the statement from a base-case, by boot-strapping the proof from the base case to all cases.

Recall that a *tree* is an acyclic connected graph.

Proposition 2. *A tree on $n \geq 1$ vertices has $n-1$ edges.*

Proof. The proof is by induction on the number of vertices, n.

As a base case, we prove the statement for each tree on 1 vertex. There is only one such tree $T_1 = (\{v\}, \emptyset)$, up to relabeling of the vertex. In this tree, the number of vertices is $n = 1$ and the number of edges is $n - 1 = 0$. This proves the statement for $n = 1$.

Our inductive hypothesis is as follows: suppose that the statement is true for all trees with at most $n-1$ vertices.

It remains to prove the inductive step: assuming that the inductive hypothesis is true, we will prove that the statement is true for trees on n vertices. Consider any tree $T = (V, E)$ on n vertices and consider a leaf u in this tree, i.e., a vertex with degree 1. We define the tree $T' = T - u$ as the tree obtained from T by deleting u (this also deletes the edge incident to u). The tree T' has $n-1$ vertices. By the inductive hypothesis, T' has $(n-1)-1 = n-2$ edges. Since T has exactly one more edge than T', we conclude that T has $n - 1$ edges. □

A.3 Proof by contradiction

where we prove p by assuming $\neg p$, arriving at a contradiction, and therefor establishing $\neg\neg p = p$.

A *vertex cover* in a graph $G = (V, E)$ is a subset of vertices $C \subseteq V$ such that for each edge $uv \in E$, we have that $u \in C$ or $v \in C$. An *independent set* in a graph $G = (V, E)$ is a subset of vertices $I \subseteq V$ such that no two vertices in I are neighbors in G.

Proposition 3. *Let $G = (V, E)$ be a graph and let $C \subseteq V$ and $I = V \setminus C$. We have that C is a vertex cover in G if and only if I is an independent set in G.*

[1] Exercise: prove the Handshake lemma.

Proof. We prove the if-and-only-if-statement ($\Leftrightarrow$) by first proving the forward direction and then the backward direction.
($\Rightarrow$): For the forward direction, we need to prove that if C is a vertex cover in G, then $I = V \setminus C$ is an independent set in G. Our proof is by contradiction. Let C is a vertex cover in G. For the sake of contradiction, assume that I is not an independent set. Therefore, there are two vertices $u, v \in I$ such that $uv \in E$. Since $I = V \setminus C$, we have that $u, v \notin C$. But then C is not a vertex cover since the edge uv has no endpoint in C – a contradiction.
($\Leftarrow$): For the backward direction, we need to prove that if $I = V \setminus C$ is an independent set in G, then C is a vertex cover in G. Our proof is by contradiction. Let I be an independent set in G. For the sake of contradiction, assume that C is not a vertex cover in G. Therefore, there are two vertices $u, v \in V \setminus C = I$ with $uv \in E$. But then, I is not an independent set because it contains two neighboring vertices – a contradiction. □

A.4 Proof by contraposition

where we prove $p \Rightarrow q$ by proving $\neg q \Rightarrow \neg p$.

An *Eulerian circuit* in G is a walk[2] in G such that the walk starts and ends at the same vertex and each edge is traversed exactly once. Here, we say that an edge uv is *traversed* by a walk if the walk contains u followed by v or v followed by u.

Proposition 4. *Let G be a connected graph. If G is Eulerian, then every vertex in G has an even degree.*

Proof. Suppose that some vertex v in G does not have even degree. Let $\mathcal{C} = (u_0, u_1, \ldots, u_m = u_0)$ be closed walk in G. We will show that $\mathcal{C}$ cannot be an Eulerian circuit. Assume that v occurs in position i in $\mathcal{C}$, so that $v = u_i$. Then, $\mathcal{C}$ traverses the edges $u_{i-1 \mod m}u_i$ and $u_iu_{i+1 \mod m}$ in this position. Since each edge is traversed exactly once, $u_{i-1 \mod m}$ and $u_{i+1 \mod m}$ do not occur anywhere else in $\mathcal{C}$ next to u_i. But since for every occurrence of v in $\mathcal{C}$, exactly two edges of G are traversed, and v has odd degree, $\mathcal{C}$ does not traverse all edges incident to v. Therefore $\mathcal{C}$ is not an Eulerian circuit. □

[2]See the glossary for a definition of walks in a graph.