CS3230 Cheatsheet
by Yiyang, AY21/22

Asymptotic Analysis

Asymptotic Notations

Big-O: $f(n) = O(g(n))$ if there exists $c > 0$ and $n_0 > 0$, s.t., for all $n \geq n_0$

$$0 \leq f(n) \leq cg(n)$$

Similar definition for **Big-Omega**, $\Omega()$.
Small-o: $f(n) = o(g(n))$ if there exists $c > 0$ and $n_0 > 0$, s.t., for all $n \geq n_0$x

$$0 \leq f(n) < cg(n)$$

Similar definition for **Small-omega**, $\omega()$.
Lastly, $f(n) = \Theta(g(n))$ iff. $f(n) = O(g(n)) \wedge f(n) = \Omega(g(n))$.

Solve Recurrence Relations

Master Theorem

For recurrence in the form of

$$T(n) = aT(n/b) + f(n)$$

there are three cases to be considered

- If $f(n) = O(n^{\log_b a - \epsilon})$ for some $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$
- If $f(n) = \Theta(n^{\log_b a} \lg^k n)$ for some $k > 0$, then $T(n) = \Theta(n^{\log_b a} \lg^{k+1} n)$
- If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some $\epsilon > 0$ and it satisfies the **regularity condition** that $af(n/b) \leq cf(n)$ for some $0 < c < 1$, then $T(n) = \Theta(f(n))$

Notes: The three cases mean whether $f(n)$ grows **polynomially** slower, around the same rate, or faster than $n^{\log_b a}$.
Notes: The regularity condition ensures the sum of sub-problems is less than $f(n)$.

Stirling's Approximation

$n! \approx \sqrt{2\pi n}(\frac{n}{e})^n$, or asymptotically, $\lg(n!) = \theta(n \lg n)$.

Harmonic Sequence

$H_n = \sum_{i=1}^{n} \frac{1}{i} = \ln n + O(1)$

Other Important Asymptotic Statements

$\lg n = O(n^\alpha)$ for any $\alpha > 0$.
$x^\alpha = O(e^x)$ for any $\alpha > 0$.

Common Recurrence Relations

(AY18/19Sem2 Midterm Qn2a) $T(n) = 2T(n/2) + n \lg \lg n = \Theta(n \lg n \lg \lg n)$
(AY19/20Sem2 Midterm Qn2d) $T(n) = 4T(n/4) + n \lg \lg \lg n = \Theta(n \lg n \lg \lg \lg n)$
(AY20/21Sem2 Midterm Qn2b) $T(n) = 27T(n/3) + n^3/\lg^2 n = \Theta(n^3)$

Hashing & Fingerprint

Amortised Analysis

There are 3 methods in Amortised Analysis.

Aggregate Method

Accounting Method ~The amortised cost provides an upper bound for the true total cost all the time. Note: The "credit saved" is associated with each of the elements operated, not overall.

Potential Method - Define potential function $\phi(i)$ potential after i-th operation. Then it follows:

- $\phi(0) = 0$ and $\phi(i) \geq 0$ for all i
- $c(i) = t(i) + \triangle\phi(i)$ relation between amortised cost, $c(i)$, and actual cost $t(i)$, of i-th operation

Reduction & Intractability

Cook-Levin Theorem

Any problem in NP poly-time reduces to 3-SAT. Hence 3-SAT is NP-Hard and NP-Complete.

Summary

Proving Techniques

Working for Accounting Method

First claim the (amortised cost) we charge each operation with. Secondly, for each operation, we compare the charge and actual cost of the operation, and claim if there is any credit left or it uses credit from other operations and how.

Prove Correctness of DP Solutions

A typical **state transition equation** is a composition equation for different cases, prove each case separately.
Use **Cut-and-Paste** arguments: If current solution S is not optimal but another S' is, then modify S' using ideas from current solution expression to create an "even better" one, thus contradicting optimality of S'.

Prove Greedy Optional Substructure

To prove that S with element i is an optimal solution

1. Claim: $S - \{i\}$ is optimal for sub-problem with i removed
2. Use **Cut-and-Paste**: If T not $S - \{i\}$ optimal for sub-problem, then $T \cap \{i\}$ better than S for original problem.

Prove NP-Hardness

To show a problem X is **NP-Hard**, choose a NP-Hard problem A, then show there is a poly-time reduction **from** A **to** X, which is

1. Any problem instance of A given can be converted to an instance of X in poly-time
2. A YES-instance to A is a YES-instance to X
3. A YES-instance to X is a YES-instance to A

Prove NP-Completeness

To show a problem X is **NP-Complete**, show 1) X is **NP**, i.e. it can be represented using a poly-time certifier and verified in poly-time, and then 2) X is **NP-Hard**.

Common NP-Complete Problems

Note: Unless otherwise stated, the descriptions below are for the decision problem version of each of the problems.

INDEPENDENT-SET

Given a graph $G = (V, E)$ and an integer k, is there a subset of k or more **vertices** such that no two are adjacent?

VERTEX-COVER

Given a graph $G = (V, E)$ and an integer k, is there a subset, S, of k or less **vertices** such that each edge is incident to at least one vertex in S?

SET-COVER

Given integers k and n, and a collection δ of subsets of $\{1, 2, ..., n\}$, are there k or less of these subsets whose union equals $\{1, 2, ..., n\}$?

3-SAT

Given a CNF (Conjunctive Normal Form) formula Φ where each clause contains exactly 3 literals (not necessarily distinct), does Φ have a satisfying truth assignment?

PARTITION

Given a collection δ of non-negative integers $x_1, ..., x_n$, is it possible to partition δ into 2 parts with equal sum?

KNAPSACK

Given non-negative integer pairs $(w_1, v_1), ..., (w_n, v_n)$ describing n items, capacity W and value threshold V, is there a subset of items with total weight at most W and total value at least V?

LONG-SIMPLE-PATH

Given an unweighted directed graph $G = (V, E)$, a positive integer l and two vertices $v, u \in V$, is there a simple path (a path with no repeated vertices) in G from u to v of length at least l?

LONG-SIMPLE-CYCLE

Given an unweighted directed graph $G = (V, E)$ and a positive integer l, is there a simple cycle (a cycle with no repeated vertices) in G on at least l vertices?

CLIQUE

Given a graph $G = (V, E)$ and a positive integer k, is there a clique ($U \subseteq V$ s.t. $forall u \neq v \in U, (u, v) \in E$) of size at least k?

HAMILTONIAN CYCLE

Given a graph $G = (V, E)$, is there a path that visits each vertex in the graph exactly once?