

# CS 381 Final Exam Cheatsheet

## 1 Master Theorem

(Easy Form) For  $a, b > 1$ , and  $T(n)$  be defined on nonnegative integers in the form of  $T(n) = aT(\frac{n}{b}) + f(n)$ , and let  $f(n) \in \Theta(n^c)$ , the following asymptotic bounds apply:

- Case 1:  $\frac{a}{b^c} < 1$ ,  $T(n) \in \Theta(n^c)$
- Case 2:  $\frac{a}{b^c} = 1$ ,  $T(n) \in \Theta(n^c \log n)$
- Case 3:  $\frac{a}{b^c} > 1$ ,  $T(n) \in \Theta(n^{\log_b a})$

(General Form)  $T(n) = aT(\frac{n}{b}) + f(n)$  for some positive function  $f(n) \geq 1$ . (A) If  $f(n) \in \Omega(n^c)$  for some constant  $c > \log_b a$  then  $T(n) \in \Theta(f(n))$  as long as the regularity condition holds (i.e.,  $af(n/b) \leq (1 - \epsilon)f(n)$  for some constant  $0 < \epsilon < 1$ ). (B) If  $f(n) \in \Theta(n^{\log_b a} \log^k n)$  for some constant  $k \geq 0$  then  $T(n) \in \Theta(n^{\log_b a} \log^{k+1} n)$ . (C) If  $f(n) \in O(n^c)$  for some constant  $c < \log_b a$  then  $T(n) \in \Theta(n^{\log_b a})$ .

## 2 Definitions

**Big-O:**  $f(n) \in \mathcal{O}(g(n))$  if there exists  $c, n_0 > 0$  such that for all  $n > n_0$ ,  $f(n) \leq c \cdot g(n)$ .

**little-o:**  $f(n) \in o(g(n))$  if for all  $c > 0$ , there exists  $n_0 > 0$  such that for all  $n \geq n_0$ ,  $f(n) \leq c \cdot g(n)$ .

**Big-Ω:**  $f(n) \in \Omega(g(n))$  if there exists  $c, n_0 > 0$  such that for all  $n > n_0$ ,  $c \cdot g(n) \leq f(n)$

**little-ω:**  $f(n) \in \omega(g(n))$  if for all  $c > 0$ , there exists  $n_0 > 0$  such that for all  $n \geq n_0$ ,  $c \cdot g(n) \leq f(n)$ .

**Big-Θ:**  $f(n) \in \Theta(g(n))$  if there exists  $c_1, c_2, n_0 > 0$  such that for all  $n > n_0$ ,  $c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$ .

## 3 Summations/Bounds

$$\begin{aligned} \sum_{i=0}^n ar^i &= \frac{a(1-r^{n+1})}{1-r} & \sum_{i=0}^{\infty} ar^i &= \frac{a}{1-r} \quad (r < 1) \\ \sum_{i=0}^n i &= \frac{n(n+1)}{2} & \sum_{i=0}^n i^2 &= \frac{n(n+1)(2n+1)}{6} \\ \sum_{i=0}^n i^3 &= \frac{n^2(n+1)^2}{4} \\ \sum_{i=1}^n \frac{1}{i} &= \Theta(\log n) & \log(n!) &= \Theta(n \log n) \end{aligned}$$

## 4 Probability

### Inclusion-Exclusion Principle

$$P(E \cup F) = P(E) + P(F) - P(E \cap F)$$

### Independence

If  $E$  and  $F$  are independent, then

$$\begin{aligned} P(E \cap F) &= P(E)P(F) \\ P(E|F) &= P(E) \end{aligned}$$

### Conditional Independence

If  $E$  and  $F$  are conditionally independent on  $X$ , then

$$\begin{aligned} P(E \cap F|X) &= P(E|X)P(F|X) \\ P(E|F \cap X) &= P(E|X) \end{aligned}$$

### Union Bound

$$P\left(\bigcup_{i=1}^{\infty} X_i\right) \leq \sum_{i=1}^{\infty} P(X_i)$$

### Markov's Inequality

$$P(X \geq a) \leq \frac{E[X]}{a}$$

## 5 Other Useful Formulae

### Binomial Coefficient: Limit Rules

- If  $\lim_{x \rightarrow c} f(x) = \lim_{x \rightarrow c} g(x) = 0$  or  $\pm\infty$ , and  $g'(x) \neq 0$ ,  $\lim_{x \rightarrow c} \frac{f(x)}{g(x)} = \lim_{x \rightarrow c} \frac{f'(x)}{g'(x)}$
- $\lim_{x \rightarrow \infty} \frac{f(x)}{g(x)} = c \Rightarrow f \in \Theta(g)$ , for a constant  $c$
- $\lim_{x \rightarrow \infty} \frac{f(x)}{g(x)} = 0 \Rightarrow f \in o(g)$
- $\lim_{x \rightarrow \infty} \frac{f(x)}{g(x)} = \infty \Rightarrow f \in \omega(g)$

## 6 Dynamic Programming Approach

**Step 1:** Clearly define your subproblems (e.g., for Knapsack problem let  $OPT[i, w]$  be the maximum profit we can obtain if we have capacity  $w$  and only consider items 1 to  $i$ )

**Step 2:** Solve the problem for the base cases. Example (Knapsack): we have  $OPT[1, w] = 0$  whenever

$w < w_1$  (item 1 cannot fit) and  $OPT[1, w] = v_1$  whenever  $w \geq w_1$  (item 1 can fit).

**Step 3:** Develop a recurrence expressing the solution to larger subproblems in terms of the solution to smaller subproblems. It typically helps to do a case analysis. Example (Knapsack): If  $w_i > w$  then item  $i$  is too big to fit so we have  $OPT[i, w] = OPT[i-1, w]$ . Otherwise, if  $w_i \leq w$  there are two cases to consider. Case 1: The optimal solution to our subproblem  $OPT[i, w]$  uses item  $i$ , or Case 2: the optimal solution does not include item  $i$ . In case 1 we obtain value  $v_i$  and can take the best solution to the subproblem with items  $1, \dots, i-1$  and remaining capacity  $w - w_i$  — we will obtain total value  $v_i + OPT[w - w_i]$ . In case 2 we will use the optimal solution to the subproblem with the first  $i-1$  items and capacity  $w$  and get value  $OPT[i-1, w]$ . Thus, if  $w_i \leq w$  we have

$$OPT[i, w] = \max\{OPT[i-1, w], v_i + OPT[i-1, w - w_i]\}.$$

Hint: If you are having trouble finding a recurrence you may need to revisit step 1 and define more subproblems or different subproblems.

**Step 4:** Fill in the DP-table and find the solution to the original problem. Example: For Knapsack the solution to the original problem with  $n$  items the optimal solution has value  $OPT[n, W]$ . We can run  $Backtrack(n, W)$  to find the original solution where  $Backtrack(i, w)$  outputs  $\{i\} \cup Backtrack(i-1, w - w_i)$  if  $w_i \leq w$  and  $OPT[i, w] > OPT[i-1, w]$ . Otherwise  $Backtrack(i, w)$  simply outputs  $Backtrack(i-1, w)$ .

## 7 Graphs

A undirected graph  $G = (V, E)$  consists of a set of nodes  $V$  and edges  $E \subseteq \{\{u, v\} : u, v \in V \wedge u \neq v\}$  connecting those nodes. A directed graph  $G = (V, E)$  consists of a set of nodes  $V$  and directed edges  $E \subseteq \{(u, v) : u, v \in V \wedge u \neq v\}$ . Note: For the purpose of this exam we do *not* consider graphs with self-loops or multiple parallel edges. We generally use  $n = |V|$  to denote the number of nodes and  $m = |E|$  to denote the number of edges.

A *path* (resp. directed path) in an undirected (resp. directed) graph  $G = (V, E)$  is a sequence  $P = v_1, \dots, v_k$  of nodes with the property that  $\{v_i, v_{i+1}\} \in E$  (resp.  $(v_i, v_{i+1}) \in E$ ) for all  $i < k$ . The path is simple if all nodes are distinct.

An undirected graph is connected if for every pair of nodes  $u, v \in V$  there is a path beginning at node  $u$  and ending at node  $v$ .

A (directed) *cycle* is a (directed) path  $v_1, \dots, v_k$  in which  $v_1 = v_k$ ,  $k > 2$  and the first  $k-1$  nodes are all distinct.

An undirected graph  $G = (V, E)$  is a *tree* if it is connected and does not contain a cycle. Any two of the following statements imply the third (1)  $G$  is connected, (2)  $G$  does not contain a cycle, (3)  $G$  has  $n-1$  edges.

An undirected graph  $G = (V, E)$  is *bipartite* if the nodes  $V$  can be partitioned into two sets  $B$  (blue) and  $R$  (red) such that every edge in  $E$  has one blue endpoint in  $B$  and one red endpoint in  $R$ .

**Thm:** A graph is bipartite if and only if it does not contain an odd length cycle

A *Directed Acyclic Graph (DAG)*  $G = (V, E)$  is a directed graph that contains no directed cycles.

A *topological order* of a directed graph  $G = (V, E)$  is an ordering of its nodes  $V$  as  $v_1, \dots, v_n$  so that for every edge  $(v_i, v_j) \in E$  we have  $i < j$ .

**Thm:** A directed graph  $G = (V, E)$  has a topological order if and only if  $G$  is a DAG.

## 8 Minimum Weight Spanning Tree (MST)

Given a connected graph  $G = (V, E)$  with real-valued edge weights  $c_e$  the MST is a subset of edges  $T \subseteq E$  such that  $T$  forms a spanning tree whose sum of edge weights ( $\sum_{e \in T} c_e$ ) is minimized.

Given  $S \subseteq V$  we say that an edge  $e = \{u, v\} \in E$  is cut by  $S$  if and only if exactly one endpoint is in  $S$  i.e.,  $|S \cap \{u, v\}| = 1$ .

**Cut Property:** Let  $S \subseteq V$  be any subset of nodes and let  $e$  be the minimum cost edge with exactly one endpoint in  $S$ . Then the MST must include  $e$ . (Assumption: all edges have distinct edge costs).

**Cycle Property:** Let  $C$  be any cycle and let  $e$  be the maximum cost edge belonging to  $C$  then the MST *does not* contain  $f$ .

**Cycle/Cut Intersection:** Let  $S \subseteq V$  be any subset  $S$  and let  $C$  be any cycle. Let  $x$  denote the number of edge in the cycle  $C$  that have exactly one endpoint in  $S$ . Then  $x$  is even.

## 9 Network Flow

Suppose we are given a flow network  $G = (V, E)$  with a source node  $s$  and a sink node  $t$  and capacities  $c(e) \geq 0$  on each directed edge.

**st-cut:** An st-cut is a given by a set  $A \subseteq V$  with  $s \in A$  and  $t \notin A$  i.e.,  $A$  contains the source but not the sink. We usually let  $B = V \setminus A$  be the set of all remaining nodes that are not in  $A$ . The capacity of

the cut  $(A, B)$  is given by

$$\text{cap}(A, B) = \sum_{e \in E \cap A \times B} c(e)$$

In particular, it is the sum of the capacities of all edges from some node in  $A$  to some node in  $B$ .

**Weak Duality:** If  $f$  is a valid flow and  $(A, B)$  is an st-cut then  $v(f) \leq \text{cap}(A, B)$ . If equality holds  $v(f) = \text{cap}(A, B)$  then  $f$  is a maximum flow and  $(A, B)$  is the minimum capacity cut.

**Max-Flow Min-Cut:** Let  $f^*$  be a maximum flow and let  $(A, B)$  be a minimum cut then  $v(f^*) = \text{cap}(A, B)$ .

**Integrality:** Suppose that every directed edge  $e$  in our flow network has integer capacity  $c(e)$ . Then there is an integral maximum flow  $f^*$  such that the flow on every edge  $f^*(e)$  is an integer.

## 10 P, NP

**Polynomial time:** A deterministic algorithm  $\mathcal{A}$  runs in polynomial time in the length of its input if there exist positive constants  $c, N_0 > 0$  such that for every input  $x$  of length  $|x| \geq N_0$  the algorithm  $\mathcal{A}(x)$  terminates after at most  $|x|^c$  steps.

A decision problem  $X$  is in **P** if there is a deterministic algorithm  $\mathcal{A}$  such that (1)  $\mathcal{A}$  decides  $X$  i.e., for  $\mathcal{A}(x) = 1$  if  $x \in X$  and  $\mathcal{A}(x) = 0$  if  $x$  is not in  $X$ , and (2)  $\mathcal{A}$  runs in polynomial time.

A decision problem  $X$  is in **NP** if there is a polynomial time certification algorithm  $C(\cdot, \cdot)$  and a positive constant  $c_1$  such that (1) for every input  $x \in X$  there exists a witness  $w$  of length  $|w| \leq |x|^{c_1}$  such that  $C(x, w) = 1$ , and (2) for every input  $x \notin X$  we have  $C(x, w) = 0$  for every *alleged* witness  $w$  that  $x \in X$ .

A decision problem  $X$  is in **coNP** if there is a polynomial time certification algorithm  $C(\cdot, \cdot)$  and a positive constant  $c_1$  such that (1) for every input  $x \notin X$  there exists a witness  $w$  of length  $|w| \leq |x|^{c_1}$  such that  $C(x, w) = 1$ , and (2) for every input  $x \in X$  we have  $C(x, w) = 0$  for every *alleged* witness  $w$  that  $x \notin X$ .

**Fact:** If  $P \subseteq NP \cap \text{coNP}$

**Fact:** A decision problem  $X$  is in **NP** if and only if its complement  $\bar{X} = \{x : x \notin X\}$  is in **coNP** (EXAMPLE: SAT denotes the set of boolean formulas that are satisfiable. UNSAT is the set of boolean formulas that are *not* satisfiable. SAT is in **NP** while UNSAT is in **coNP**.)

The notation  $X \leq_p Y$  indicates that there is a polynomial time reduction from decision problem  $X$  to decision problem  $Y$ . If  $X \leq_p Y$  and  $Y \leq_p Z$  then  $X \leq_p Z$ .

## 11 NP-Complete Problems

A decision problem  $X$  is **NP-Hard** if for any problem  $Y$  in **NP** we have  $Y \leq_p X$  i.e., there is a polynomial time reduction from  $Y$  to  $X$ .

A decision problem  $X$  is **NP-Complete** if (1)  $X$  is in **NP**, and (2)  $X$  is **NP-Hard** i.e., for any problem  $Y$  in **NP** we have  $Y \leq_p X$  i.e., there is a polynomial time reduction from  $Y$  to  $X$ .

**Fact:** If  $X$  is **NP-Hard** and  $X \leq_p Y$  then  $Y$  is also **NP-Hard**.

Here is a list of **NP-Complete** decision problems. On the final exam you may assume, without proof, that these problems are **NP-Complete**.

**3COLOR:** Given a simple undirected graph  $G = (V, E)$  can we color each node  $v \in V$  with one of three colors (e.g.,  $c(v) \in \{\text{red}, \text{blue}, \text{green}\}$ ) any adjacent pair of nodes has different colors i.e., for all edges  $\{u, v\} \in E$  we have  $c(u) \neq c(v)$ .

**CLIQUE:** Given a simple undirected graph  $G = (V, E)$  and an integer  $k$  does  $G$  contain a clique of size  $k$ ? More formally, does there exist a subset  $S \subseteq V$  of  $|S| \geq k$  nodes such that for all pairs  $\forall u, v \in S$  with  $u \neq v$  we have  $\{u, v\} \in E$ .

**VERTEX COVER (VC):** Given a simple undirected graph  $G = (V, E)$  and an integer  $k$  does  $G$  contain a vertex cover of size *at most*  $k$ ? More formally, does there exist a subset  $S \subseteq V$  of  $|S| \leq k$  nodes such that for every edge  $\{u, v\} \in E$  we have  $\{u, v\} \cap S \geq 1$  i.e.,  $S$  contains at least one of the endpoints  $u$  and  $v$ .

**INDEPENDENT SET:** Given a simple undirected graph  $G = (V, E)$  and an integer  $k$  does  $G$  contain an independent of size *at least*  $k$ ? More formally, does there exist a subset  $S \subseteq V$  of  $|S| \geq k$  nodes such that for all pairs  $\forall u, v \in S$  with  $u \neq v$  we have  $\{u, v\} \notin E$  i.e., there are no edges between any pair of nodes in

$S$ .

**CIRCUIT SAT:** Given a circuit  $C$  with  $n$  boolean inputs and one boolean output does there exist inputs  $(x_1, \dots, x_n) \in \{0, 1\}^n$  such that  $C(x_1, \dots, x_n) = 1$ ?

**SAT:** Given a boolean formula  $\Phi$  over  $n$  variables does there exist an assignment  $(x_1, \dots, x_n) \in \{0, 1\}^n$  to our  $n$  variables such that  $\Phi(x_1, \dots, x_n)$  evaluates to true?

**3SAT:** Given a 3CNF<sup>1</sup> boolean formula  $\Phi$  over  $n$  variables does there exist an assignment  $(x_1, \dots, x_n) \in \{0, 1\}^n$  to our  $n$  variables such that  $\Phi(x_1, \dots, x_n)$  evaluates to true?

**Subset Sum:** Given  $n$  positive integers  $x_1, \dots, x_n$  and a target integer  $T$  does there exist a subset  $S \subseteq \{1, \dots, n\}$  such that  $\sum_{i \in S} x_i = T$ ?

**PARTITION:** Given  $n$  positive integers  $x_1, \dots, x_n$  does there exist a subset  $S \subseteq \{1, \dots, n\}$  such that  $\sum_{i \in S} x_i = \frac{1}{2} \sum_{i=1}^n x_i$ ?

**(DIR) HAM PATH:** Given a simple undirected (resp. directed) graph  $G = (V, E)$  with  $n = |V|$  nodes does there exist a simple undirected (resp. directed) path of length  $n - 1$  i.e., a path which visits every node exactly once?

**(DIR) HAM CYCLE:** Given a simple undirected (resp. directed) graph  $G = (V, E)$  with  $n = |V|$  nodes does there exist a simple undirected (resp. directed) cycle containing all  $n$  nodes?

---

<sup>1</sup>a boolean formula in conjunctive normal form with at most 3 literals in each clause