CS 366 Exam 3 Study Guide

Fall 2025 - Dynamic Programming & Computational Complexity

Structure

- Paper exam, closed book
- Maximum 120 minutes (but designed for 80 minutes; meaning ~1/2 done in about 40m)
- Total ~50 points + 4 bonus points
- Question types:
  - Fill in DP tables (Coin Change, Knapsack)
  - SAT notation conversion and satisfiability
  - Graph problems (Hamiltonian Cycle, TSP)
  - Short answer / explanation questions
  - Complexity classification
- In-person; proctor must be able to see your work

Topics Covered

1. Dynamic Programming

Coin Change

- Fill in C[i][j] table (minimum coins to make j cents using denominations i..n)
- Traceback to find which coins are used
- Know why greedy fails (e.g., denom=[10,6,1], amount=12 → greedy gives 3, DP gives 2)
- Runtime: $\Theta(n \times A)$

0/1 Knapsack

- Fill in `exist[i][j]` table (can we make exactly j with items 1..i?)
- Fill in `belong[i][j]` table (is item i used to make j?)
- Traceback to find selected items
- Runtime: $\Theta(n \times k)$

---

2. Boolean Satisfiability (SAT)

Notation

- List format: `[[1, -2, 3], [-1, 2]]` means $(a \vee \neg b \vee c) \wedge (\neg a \vee b)$
- Positive = variable, Negative = negated variable

Skills

- Convert list notation to CNF expression
- Determine if satisfiable (find valid assignment)

- Verify an assignment satisfies all clauses

---

3. Hamiltonian Cycle & TSP

Hamiltonian Cycle (HC)

- Decision problem: Does a cycle visiting all vertices exactly once exist?
- Can stop once ANY valid cycle is found

Traveling Salesperson Problem (TSP)

- Optimization problem: Find minimum-weight Hamiltonian cycle
- Must explore ALL tours to guarantee optimum
- Runtime: O(n!) worst case

Key Difference: HC asks "does it exist?" TSP asks "what's the best?"

---

4. Computational Complexity

Know examples for each class:

Class	Examples
P	Sorting, Shortest Path, MST, GCD
NP	SAT, HC, TSP, Clique, Vertex Cover
NP-complete	SAT, 3-SAT, HC, TSP, Clique, Vertex Cover, Knapsack
Unsolvable	Halting Problem

Key Concepts

- P ⊆ NP (every problem solvable in poly-time is also verifiable in poly-time)
- NP-complete = NP ∩ NP-hard
- To prove NP-complete: (1) show in NP, (2) reduce known NP-complete problem to it

---

5. NP-Completeness Proofs

Showing a problem is in NP:

- Write a polynomial-time verification algorithm
- Given a "certificate" (proposed solution), verify it's correct in poly-time

Reduction ($A \le_p B$):

- Transform instances of A into instances of B in polynomial time
- If $A \le_p B$ and A is NP-hard, then B is NP-hard
- Example: $HC \le_p TSP$ (set all edge weights to 1, bound B = n)