# Final Review Checklist

Here are a list of algorithms, paradigms, math, and definitions that you should know from lecture, discussion, and homework from the second half of the course. This is not meant to be comprehensive. It is merely a reminder of what we have covered thus far.

## Algorithms / Problems

1. Activity Scheduling (Greedy)
2. Office party invitations - no direct supervisor (Dynamic Programming)
3. Office party invitations - no indirect supervisor (Greedy)
4. Magnetic Tape program placement (Greedy)
5. Stable Marriage
6. Stable Roommate
7. Huffman's algorithm for determining Huffman codes (Greedy)
8. String Matching
   - (a) Brute Force method
   - (b) FSM
   - (c) Approximate string matching / String alignment
   - (d) Local approximate string matching / Local alignment
   - (e) Approximate string matching / String alignment without end spaces
9. Kruskal's algorithm for finding a minimum spanning tree (Greedy)
10. Second-best minimum spanning tree
11. Shortest paths
    - (a) Dijkstra's algorithm ("single source shortest paths")
    - (b) Bellman-Ford ("single source shortest paths")
    - (c) Floyd-Warshall (Dynamic Programming) ("all pairs shortest paths")
    - (d) Solovay-Strassen ("all pairs shortest paths")
12. 3SAT
13. CLIQUE
14. Vertex-Cover
15. 3COLOR, 4COLOR

## Definitions

1. prefix code
2. graph
3. path
4. connected graph
5. component
6. cycle
7. acyclic
8. tree
9. spanning tree
10. forest
11. Binary Heap / Priority Queue
    (a) Insert($i$)
    (b) Delete($i$)
12. Union-Find ADT
    (a) Makeset($i$)
    (b) Union($s_1, s_2$)
    (c) Find($i$)
    (d) naive implementation
    (e) tree implementation
    (f) binomial trees implementation
    (g) collapsing find
    (h) union by rank
    (i) implementation using both collapsing find and union by rank
    (j) The Work Theorem
13. Binary heap for implementation of "Delete-Min"
    (a) Insert($i$)
    (b) Delete-Min()
    (c) Decrease-Key($i$)
14. A program $\Pi$ is in NP if for all instances $\Pi_i \in \Pi$ (i.e., for which the answer to the decision problem is yes), $\exists$ a machine $M$ and a "proof" or "certificate" $x$, such that $M(\Pi_i, x) = \text{True}$; and further, for all instances $\Pi_i \notin \Pi$ (i.e., for which the answer to the decision problem is yes), $\nexists$ a machine $M$ and $\nexists$ a "proof" or "certificate" $x$, such that $M(\Pi_i, x) = \text{True}$.
15. A program $\Pi$ is NP-hard if $\forall L \in \text{NP}$, $L \leq_T \Pi$.
16. A program $\Pi$ is NP-complete if it is in NP and if it is NP-hard.
17. Reduction ($\leq_T$)

## Math

1. $\log^* n$

## Concepts

1. Proving that Greedy works
2. Reducing one problem to another / Proving NP-completeness