Review For Final

Some notes:
Exam hours: 10:00 – 12:15 (Come at least 10 minutes earlier)
Classroom arrangement: VH 32 [61-0527, 98-6814] VH 29 [10-8882, 61-0525]
No blank paper
No restroom break
No access to laptops, internet, phones, books, notes etc.

1. *Defintions* Be familiar with the definitions of the following concepts.
   - Heap(Maxheap and Minheap)
   - vertex cover
   - Hamiltonian graph / Hamiltonian cycle
   - connected graph
   - connected components
   - simple cycle in a graph
   - sparse graph / dense graph
   - adjacency list/matrix
   - subgraph
   - complete graph K_n
   - tree (as a graph)
   - P vs. NP ($P \subseteq NP$)
   - NP-hard and NP-complete
   - Polynomial Reducibility

2. A heap with height h, n nodes: $2^h \leq n \leq 2^{h+1} - 1$
3. Know the BottomUpHeap algorithms, both the recursive and iterative versions.
4. Know the basic concepts for the three algorithm design strategies(Greedy strategy, divide and conquer, and dynamic programming).
5. Know the recursive solutions for Fibonacci, SubsetSum and Edit Distance. And how we come up with the dynamic programming solution from the recursive solution.
6. Be familiar with Knapsack problem.
7. Know in detail the BFS and DFS algorithms and how they are used for various tasks(spanning tree/forest, number of connected components, cycle detection, path exists between two vertices, shortest path).
8. Properties about undirected graphs:

   i. $\Sigma_v \deg(\mathbf{v}) = 2m$
   ii. $m \leq n(n - 1)/2$
   iii. **Theorem.** If G is a tree, then **m = n –1.**

9. Be able to carry out the steps of Kruskal's algorithm using the DisjointSets data structure (i.e. trees) to handle clusters.
10. Know how to do the steps of Dijkstra's shortest path algorithm with/without priority queue and how the running time is improved using priority queue.
11. Be familiar with TSP problem.
12. Be able to verify that a decision problem belongs to *NP*.
13. Be able to establish that a decision problem is NP-complete, given that some other (related) problem is already known to be NP-complete.
14. Be able to explain this point - if anyone finds a polynomial time solution to an NP-complete problem, then P = NP.
15. Know the brute force algorithm for determining the smallest size of a vertex cover for a graph. Know also the VertexCoverApprox algorithm.
16. For each algorithm listed in this document, know its running time (and how to derive that running time).