

## Final Exam Announcements

---

The final examination will be held on **Friday, December 16, at 8:00-10:50am in CCB 17.**

The exam will be cumulative and closed book. You may, however, bring one 8.5" x 11" sheet of paper as a crib sheet. Preparing a crib sheet can be a useful study aid, so take time in selecting the material for it. You may use BOTH SIDES of the sheet and write as small as you like. No mechanical or electronic reproductions are allowed. You must hand in your crib sheet when you hand in your exam.

**Form:** The form of the final exam will be similar to the form of the midterm exam. The problems will require your knowledge and understanding of the material more than creativity.

**Material covered:** You are responsible for the material covered in lectures and assigned for reading till Friday, December 2 (inclusive). In particular, it includes:

1. Material covered by the Midterm Exam: asymptotic notation ( $O$ ,  $\Omega$ ,  $\Theta$ ), methods for solving recurrences (substitution, divide-conquer recurrences and the master method), insertion sort, divide-conquer algorithms (merge sort, quicksort, Strassen's matrix multiplication, integer and complex number multiplication, median and order statistic: randomized and deterministic), lower bound for comparison sorting, graphs (representations, dfs and its applications: connectivity and strong connectivity, dag, topological sorting, bfs with applications, shortest paths algorithms (bfs, Dijkstra's alg and Bellman-Ford alg.), heaps.
2. After Midterm:
  - **greedy algorithms:** Minimum Spanning Tree (Kruskal, Prim, disjoint-sets data structure), Huffman codes, Greedy approximation of the set cover problem.
  - **dynamic programming:** shortest paths in dag, longest increasing subsequence, Edit distance problem, Knapsack and 0-1 Knapsack, All-pairs shortest paths (Floyd-Warshall alg., transitive closure of a graph).
  - **Network Flow:** Ford-Fulkerson method, Edmonds-Karp alg., maximum bipartite matching, multiple sources and sinks
  - **NP-completeness:** P-class, NP, NP-hard and NP-complete, Circuit SAT, SAT and 3 SAT, 3SAT is NP-complete, Independent Set, Clique and Vertex Cover are NP-complete, Subset Sum Problem is NP-complete, Undecidability.
  - **Number theoretic algorithms:** Basic and modular arithmetic, Euclid's algorithm for greatest common divisor and its extension, Randomized Primality testing.

**Office hours:** My office hours next week will be **on Monday, 12/12 at 11-1**

Danupon will have office hours on **Tuesday, 12/13, 1-2pm and Thursday 12/15, 2-4pm.**

There will be a **review session on Wednesday, Dec. 14 at 6-8pm in CCB 53.**

**Algorithms:** For all algorithms presented in class you should know their main ideas, general techniques used (e.g. dynamic programming, greedy method, etc.), data structures and the running time. Moreover,

you should be able to perform a comparison analysis of different algorithms solving the same problem. No pseudocodes are required.

**Sample problems:** Review all homework assignments and problems from the midterm exam.

1. Consider the problems below. Justify your True or False answer in each part.
  - True or False: The merging of two sorted sequences, consisting of  $n$  elements each, takes  $\Omega(n \log n)$  comparisons.
  - True or False: Given two  $n \times n$  matrices, their product can be found in  $O(n \log n)$  time.
  - True or False: Any depth-first-search of a graph with edge weights produces a minimum spanning forest of that graph.
  - Let  $G = (V, E)$  be a connected undirected graph in which all edge weights are the same. A minimum spanning tree of  $G$  can be found in time  $O(|V| + |E|)$ .
  - True or False: Finding the shortest distance between a vertex  $s$  and a vertex  $t$  in an undirected graph ( with *no* edge weights) with  $n$  vertices takes  $\Omega(n^3)$  time.
  - True or False: There is an implementation of Dijkstra's shortest path algorithm that takes time  $O(n \log n)$  for *sparse* graphs.
  - True or False: The minimum spanning tree can be found in linear time using Kruskal's algorithm.
  - True or False: The problem of deciding if an undirected graph is connected is in **NP**.
  - True or False: All problems in **NP** take at least exponential time.
  - True or False: No problem in **NP** is solvable in polynomial-time.
  - True or False:  $\gcd(na, nb) = n \cdot \gcd(a, b)$ .
  - True or False: If an **NP-Complete** problem is solvable in polynomial-time, then every problem in **NP** is solvable in polynomial-time.
  - True or False: The heaviest edge in a connected graph cannot belong to a minimum spanning tree.
  - True or False:  $NP = co\_NP$ .
  - True or False: Knapsack problem is in  $P$ .
2. Let  $G$  be a connected, undirected graph with non-negative integer weights. Let  $G$  have  $n$  vertices and  $n$  edges. Is it possible to find a minimum spanning tree in  $G$  in linear time? Why?
3. Suppose we use the array data structure instead of a priority queue in Prim's algorithm for computing a minimum spanning tree of a undirected graph.
  - a What is the running time for Prim's algorithm using the array data structure?
  - b How does this running time compare with Kruskal's algorithm for *sparse* graphs? Why?
  - c How does this running time compare with Kruskal's algorithm for *dense* graphs? Why?

4. Let  $A$  be a decision problem to which all NP problems are reducible in polynomial time. Let  $A$  be reducible to a decision problem  $B$  in polynomial time. Is it true that if  $B$  is NP-Complete then  $A$  is also NP-Complete? Why?
5. Consider the disjoint set data structure implemented using linked lists. Suppose we use this implementation in Kruskal's algorithm for computing a minimum spanning tree of a connected undirected graph. What is the running time of this algorithm ?
6. Let  $G = (V, E, s, t, c, )$  be a flow network and  $f$  be a flow function. Give an efficient algorithm to decide if  $f$  is a maximum flow.
7. Give an efficient algorithm to compute a minimum capacity cut in a network with 0-1 capacities.
8. Give an efficient algorithm to compute a maximal independent set (MIS) in a graph (MIS is an independent set that is not contained, as a subset, in any other independent set).
9. Let  $G$  be a connected, undirected graph on  $n$  vertices with a single cycle. Show how to find a MST in  $G$  in time  $O(n)$ .
10. Let  $G$  be a directed graph with  $n$  vertices and  $m$  edges. Let  $t$  be a vertex in  $G$ . Describe an  $O(nm)$  algorithm to decide if  $G$  has a negative cycle that can reach  $t$ .
11. A full binary tree is a rooted tree in which every internal node has exactly two children. Show that a code corresponding to a non-full binary tree cannot be optimal.
12. Let  $G = (V, E)$  be a graph and  $H = (V', E')$  be a clique in  $G$ . Show that it is NP-complete to decide if  $H$  is not a maximum sized clique in  $G$ .
13. Suppose that someone gives you a black-box algorithm  $A$  that takes an undirected graph  $G = (V, E)$ , and a number  $k$ , and behaves as follows.
  - If  $G$  is not connected, then it simply returns "G is not connected"
  - If  $G$  is connected and has an independent set of size at least  $k$ , then it returns "Yes."
  - If  $G$  is connected and does not have an independent set of size at least  $k$ , then it returns "No."

Suppose that the algorithm  $A$  runs in time polynomial in the size of  $G$  and  $k$ . Show how, using calls to  $A$  you could then solve the Independent Set Problem in polynomial time: Given an arbitrary undirected graph  $G$ , and a number  $k$ , does  $G$  contain an independent set of size at least  $k$ ?

14. Give verification algorithms to show that the following problems are in  $NP$ . Formulate each problem as a decision problem first, if necessary.
  - *Longest Path*: Given an undirected graph  $G = (V, E)$  and vertices  $u$  and  $v$ , what is the longest simple path between  $u$  and  $v$ ?
  - *Graph Coloring*: Given an undirected graph  $G = (V, E)$ , what is the minimum number of colors with which one can color the vertices of the graph in such a way that no two adjacent vertices have the same color?