Practice Questions and Reading Material

The final exam is cumulative and will include problems relating to all the lectures. Please go over all three study guides. The current study guide includes material corresponding only to the third module of the class (MST, NP-Complete problems).

The chapters and exercises from the books are provided as supportive references for the material covered during the lectures.

Minimum Spanning Trees

Reading material: Chapter 5.1 from DPV, Chapters 23.1, 23.2 from CLRS2
Practice questions:

- What kind of graph-based optimization problems require as a solution the computation of a Minimum Spanning Tree? Why the solution to such problems does not contain cycles?
- Provide a generic description of an algorithm for the computation of minimum spanning trees.
- What is a cut of a graph? When does an edge cross a cut? When does a cut respect a set of edges? What is the definition of a light edge?
- Assume a connected, undirected graph $G(V,E)$ with weights w and a set of edges $A \subset E$ that is a subset of minimum spanning tree T. Consider then any cut $(S, V-S)$ that respects the set A. Prove that if (u,v) is a light-edge crossing $(S, V-S)$, then (u,v) can be safely be added to A as an edge that is part of a minimum spanning tree of G.
- Given the above statement, prove the following: Let $C = (V_C, E_C)$ be a connected component (tree) in the forest $G_A(V,A)$. If (u,v) is a light-edge connecting C to another component in G_A, then (u,v) is safe for A.

Kruskal's and Prim's algorithms

Reading material: Chapter 5.1 from DPV, Chapters 23.2, 21.2, 21.2, 21.3 from CLRS2
Practice questions:

- Given the generic algorithm for the computation of a minimum spanning tree, how is the safe edge computed in Kruskal's algorithm and how is it computed in Prim's algorithm?
- You may be provided a graph and asked to trace the steps of Kruskal's and Prim's algorithms for the computation of the minimum spanning tree of the graph.
- Provide Kruskal's algorithm.
- Consider an implementation of the disjoint sets data structure with the union-by-rank heuristic as part of the implementation of Kruskal's algorithm. What is the running time of Kruskal's algorithm in this case and why? Consider an implementation of the disjoint sets data structure with the union-by-rank and the path compression heuristic. Furthermore, consider that the edge weights are upper bounded by the value $|E|$. What is the running time of Kruskal's algorithm in this case and why?
- Describe the makeset, find_set and union functions of the disjoint sets data structure given the union-by rank heuristic. What is the running time of these operations? How does the find_set function change when you also use the path compression heuristic? What is the new running time of the find_set and union functions in this case?

- Provide Prim's algorithm.
- What is the running time of Prim's algorithm given an array implementation of a priority queue? What is the running time of the algorithm given a binary heap implementation of a priority queue?

The class of NP problems - Examples of NP complete problems

Reading material: Chapter 8.1 from DPV, Chapters 34.2, 34.5 from CLRS2
Practice questions:

- What is the satisfiability problem? What is the worst-case running time of the best known algorithm for this problem? What is the running time of checking whether a candidate solution is truly solving the problem or not?
- Are there versions of the general satisfiability problem that can be solved in polynomial time? Describe two of them.
- What is the traveling salesman problem? What is a dynamic programming approach for the traveling salesman approach? What is the running time of this approach?
- Show that any optimization problem can be reduced to a search problem. Show that any search problem can be reduced to an optimization problem. Why do we typically prefer to work with search problems when studying the computational complexity of algorithms?
- What is an Eulerian tour? When does a graph have an Eulerian tour?
- What is the Rudrata cycle/path problem? Is there a polynomial time algorithm for this problem?
- What is the independent set problem? Describe a dynamic programming solution for computing an independent set on trees. What is the running time of this solution? Is there a polynomial time algorithm for this problem on general graphs?
- What is the vertex cover problem? What is the clique problem? Are there polynomial time algorithms for these problems?

Additional examples of NP complete problems - Reductions

Reading material: Chapters 8.1, 8.2, 8.3 from DPV, Chapters 34.5, 34.3 from CLRS2
Practice questions:

- What is the minimum cut problem? Is there a polynomial time algorithm for this problem?
- Consider the following approach for computing a minimum cut: run Kruskal's algorithm on an unweighted graph and remove the last edge of the resulting minimum spanning tree to define a cut (randomize the selection of the edges among multiple equivalent choices). Show that the probability of this cut being the minimum cut is at least $\frac{1}{n^2}$. What is the running time of the randomized approach that computes the minimum cut with high probability through Kruskal's algorithm?
- What is the balanced cut problem? Is there a polynomial time algorithm for this problem?
- What is the knapsack problem? Is there a polynomial time algorithm for this problem? What is the subset sum problem and how does it relate to the knapsack problem?
- What is the class of NP problems? What is the class of P problems? What is the relation between these two classes of problems? For instance, is P = NP?

- What does it mean that you can reduce a search problem A to a search problem B? What do you need to do in order to provide such a reduction? (you can provide a drawing to explain your answer)
- What is the class of NP-compete problems? What is the class of NP-hard problems?
- When is a problem in the class of co-NP problems? Provide an example of a co-NP problem.

Examples of reductions between NP complete problems

Reading material: Chapter 8.3 from DPV, Chapter 34.3, 34.4 from CLRS2
Practice questions:

- Show that the general satisfiability (SAT) problem reduces to the 3-SAT problem.
- Show that the 3-SAT problem reduces to the Independent Set problem.
- Show that the Independent Set problem reduces to the Vertex cover problem.
- Show that the Independent Set problem reduces to the Clique problem.
- Show that the Circuit SAT problem reduces to the SAT problem. What is the importance of this reduction?

Intelligent Exhaustive Search, Intro to Approximation Algorithms

Reading material: Chapters 9.1, 9.2 from DPV, Chapter 35.Intro from CLRS2
Practice questions:

- What is the idea behind backtracking search in order to solve NP-complete problems?
- Describe a general framework for backtracking search.
- Describe the application of backtracking search to the satisfiability problem, i.e., which subproblems are expanded at each iteration and which branching variable is considered?
- Describe the general branch-and-bound approach for optimization problems.
- Describe an application of branch-and-bound to the traveling salesman problem.
- How is the approximation ratio of an approximation algorithm computed?
- Provide an approximation algorithm for the vertex cover problem. What approximation ratio does it achieve?

Approximation Algorithms, Local Search Heuristics

Reading material: Chapters 9.2, 9.3 from DPV, Chapter 35.1-3 from CLRS2
Practice questions:

- When does a distance function satisfy metric properties?
- What is the k-clustering NP-complete problem? Provide a simple approximation scheme for the k-clustering problem. What is the approximation ratio that it achieves. Prove it.
- Provide an approximation algorithm for the traveling salesman problem given that the underlying graph has edge weights that satisfy metric properties. What is the approximation ratio for this algorithm? Prove it.
- Describe the general local search framework.
- Provide a local search approach for the traveling salesman problem.

- Provide a local search strategy for the graph partitioning problem.
- How can you deal with local optimal in the context of local search?

Related exercises from DPV (i.e., the book “Algorithms” by Dasgupta, Papadimitriou, Vazirani):
Chapter 5: 5.1, 5.2, 5.3, 5.4, 5.6, 5.7, 5.8, 5.9, 5.10, 5.11, 5.12, 5.20, 5.21, 5.23, 5.24
Chapter 8: 8.1, 8.2, 8.3, 8.4a, 8.5, 8.6, 8.7, 8.9, 8.12, 8.14, 8.15
Chapter 9: 9.1, 9.2, 9.3, 9.4, 9.5, 9.6, 9.7, 9.8, 9.9

Related exercises from CLRS - 2nd Edition (i.e., the book “Introduction to Algorithms” by Cormen, Leiserson, Rivest, Stein):
Chapter 21: 21.1-2, 21.1-3, 21.3-3
Chapter 23: 23.1-1, 23.1-2, 23.1-3, 23.1-4, 23.1-5, 23.1-6, 23.1-7, 23.1-8, 23.1-8, 23.1-9, 23.1-10, 23.2.3, 23.2.4, 23.2.5