Advanced Algorithms: Top 10 Ideas

CS 181 Advanced Algorithms — Spring 2026

---

1. **The Max-Flow Min-Cut Theorem**

   Coming in at number 1 is the beautiful theorem relating the maximum flow and minimum cut in any graph. We saw several algorithms and many applications for computing both of these quantities, including the Ford-Fulkerson algorithm and the Push-relabel paradigm. We saw applications to computer vision, social network analysis, matrix-rounding and fault-tolerant network design.

2. **Hall's Theorem**

   We have seen the surprising prevalence of matchings in decision-making tasks. When does a bipartite graph have a perfect matching? The simple condition that all neighborhoods must expand (or at least not shrink) is both necessary and sufficient, and has many algorithmic consequences including the Hungarian Algorithm for minimum-cost perfect matching in a bipartite graph.

3. **Linear Programming is in P**

   This fact alone may be the most useful takeaway from this class if you are a practitioner of applied algorithms. Many problems are directly modeled as Linear Programs. Not only this, but LP provides a unified language for all optimization problems and is commonly used as a subroutine in the design of algorithms in general.

4. **Linear Programming Duality**

   All Linear Programs come in pairs, where the feasible solutions of each provide bounds on the optimal solution of the other (by construction). The surprising thing is that their optimal values always coincide.

5. **Integer Programming is NP-Hard**

   Almost every combinatorial optimization problem can be modeled as an Integer Linear Program. While solving them is NP-hard in general, plugging your problem into a highly optimized ILP solver will give you a solution faster than most brute-force algorithms. Not only this, but relaxing the integrality constraints defines a whole new class of fractional solutions to combinatorial problems we know and love.

6. **A Perfect Formulation for Bipartite Matching**

   This is the prototypical example of how linear programs can be utilized to tackle complicated combinatorial problems. The set of fractional solutions to the Perfect Matching LP defines a geometric polyhedron in m-dimensional space, whose corners will always have integer coordinates. Thus, we get an optimal integral solution by simply solving the LP.

7. **Beating NP-Hardness**

   The main takeaway here is that NP-hardness is not the end of the road! We saw a variety of techniques to get around this fundamental limit on tractability, including restricting to cases with more structure, parameterization and approximation.

8. **Approximation Algorithms: Problems, Solutions, and Techniques**

   The Traveling Salesperson Problem, vertex cover, max-cut and load balancing. We saw how fundamental problems can be tackled through the lens of approximation algorithms. More importantly, we worked with a variety of techniques which are generally useful in the design of algorithms including greedy algorithms, local search, and LP-based rounding.

9. **Online Algorithms and Competitive Analysis.**

   While computing a solution fast given full information is very useful, in most decision-making problems we encounter in real life, the bottleneck may not be computational time but lack of information. The brilliant thing is that we can design algorithms which perform well on the fly despite not knowing what the future holds. We saw applications to ski-rental, optimal-stopping (secretary), Online matchings and memory management. Policies for these problems are competitive against a strong benchmark: the clairvoyant optimum.

10. **Randomness is Provably Useful**

    In the world with an uncertain and adversarial future, it turns out that "hedging" can be a good strategy for making decisions! For some problems, there is a provable separation between the performance of a randomized algorithm and any deterministic algorithm. Whether this phenomenon holds in the realm of computational time-complexity remains a major open question in the field.
