Topics:

1. Greedy algorithms
   - 4.5 The Minimum Spanning Tree Problem
   - 4.6 Implementing Kruskal's Algorithm: The Union-Find Data Structure
   - 4.8 Huffman Codes and Data Compression
2. Divide and conquer
   - 5.1 A First Recurrence: The Mergesort Algorithm
   - 5.2 Further Recurrence Relations
   - 5.3 Counting inversions
   - 5.4 Finding the Closest Pair of Points
3. Dynamic programming
   - 6.1 Weighted Interval Scheduling: A Recursive Procedure
   - 6.2 Principles of Dynamic Programming: Memorization or Iteration over Subproblems
   - 6.3 Segmented Least Squares: Multi-Way Choices
   - 6.4 Subset Sums and Knapsacks: Adding a Variable
   - 6.5 RNA Secondary Structure: Dynamic Programming over Intervals
   - 6.6 Sequence Alignment
   - 6.8 Shortest Paths in a Graph
4. Network flow
   - 7.1 The Maximum-Flow Problem and the Ford-Fulkerson Algorithm
   - 7.2 Maximum Flows and Minimum Cuts in a Network
5. NP
   - 8.1 Polynomial-Time Reductions
   - 8.4 NP-Complete Problems
   - 8.10 A Partial Taxonomy of Hard Problems

Vertex Cover

On a graph $G = (V, E)$, a vertex cover $C \subseteq V$ is a subset of the vertices such that for all $e = (u, v) \in E$, $u \in C$ or $v \in C$. That is, it is a set of vertices such that every edge has at least one of its endpoints in the vertex cover. The cost of a vertex cover is the size of set C.
Suppose you are given a bipartite graph $G = ((L \cup R), E)$ and wish to find a vertex cover of smallest cost.

1. Set up a min-cut in $G \cup \{s, t\}$ that does not cut any edges of E.
2. Show how to use this min-cut to get a vertex cover of minimum size for G. That is, show how to select a subset $l \subseteq L$ and a subset $r \subseteq R$, based on the cut, such that every edge is incident to some element of $l \cup r$.

page 505, Ex.1

For each of the two questions below, decide whether the answer is (i) "Yes," (ii) "No," or (iii) "Unknown, because it would resolve the question of whether $\mathcal{P} = \mathcal{NP}$." Give a brief explanation of your answer.

- Let's define the decision version of the Interval Scheduling Problem from Chapter 4 as follows: Given a collection of intervals on a time-line, and a bound k, does the collection contain a subset of non-overlapping intervals of size at least k?

  Question: Is it the case that Interval Scheduling $\leq p$ Vertex Cover?

- Question: Is it the case that Independent Set $\leq p$ Interval Scheduling?

page 505, Ex.2

A store trying to analyze the behavior of its customers will often maintain a two-dimensional array A, where the rows correspond to its customers and the columns correspond to the products it sels. The entry $A[i, j]$ specifies the quantity of product j that has been purchased by customer i.

Example:

	liquid detergent	beer	diapers	cat litter
Raj	0	6	0	3
Alanis	2	3	0	0
Chelsea	0	0	0	7

One thing that a store might want to do with this data is the following. Let us say that a subset S of the customers is *diverse* is no two of the customers in S have ever bought the same product (i.e., for each product, at most one of the customers in S has ever bought it). A diverse set of customers can be useful, for example, as a target pool for market research.

We can now define the Diverse Subset Problem as follows: Given an $m \times n$ array A as defined above, and a number $k \leq m$, is there a subset of at least k of customers that is *diverse*?

Show that Diverse Subset is NP-complete.

page 505, Ex.3

Suppose you're helping to organize a summer sports camp, and the following problem comes up. The camp is supposed to have at least one counselor who's skilled at each of the n sports covered by the camp (baseball, volleyball, and so on). They have received job applications from m potential counselors. For each of the n sports, there is some subset of the m applicants qualified in that sport. The question is: For a given number $k < m$, is it possible to hire at most k of the counselors and have at least one counselor qualified in each of the n sports? We'll call this the *Efficient Recruiting Problem*.

Show that Efficient Recruiting is NP-complete.

Proof The Efficient Recruiting problem is exactly the Set Covering problem, but formulated in different terms. Formally, we can prove that Efficient Recruiting is $\leq_p$ that Set Covering by providing an algorithm for Set Covering, which uses Efficient Recruiting as a subroutine.

Let set U of n elements represent some distinct virtual sport types
Let a collection $S_1, \ldots, S_m$ represent applicants, whereby each subset S represents proficiency of an applicant in each of virtual sports
return Run Efficient_Recruiting(n sports, m applicants, k positions)

N-ary Huffman Codes

While binary coding dominates, there are cases where more than two coding symbols are preferable. In the context of electromechanical devices, one often has a channel that allows for more than two voltage values: It would be a waste not to use this possibility to our advantage!

Closest pair of points problem

- 1-D version
- 2-D version

Projects

We're running a lab with P people working for us. We have a set of possible projects $1, 2, ..., n$ each of which requires some number of people $p_1, p_2, ..., p_n$. Our goal is to determine the maximum number of projects we can work on simultaneously; in other words, the size of the largest set $S \subseteq \{1, 2, ..., n\}$ such that $\Sigma_{x \in S} p_x \leq P$.

1. Show that a simple greedy algorithm solves this problem in $O(n \log n)$ time.
2. Now suppose that we additionally have C computers for our lab. Now each project i requires p_i people *and* c_i computers. Give an example to show that the greedy algorithm no longer works.
3. Design a dynamic programming algorithm to find the maximum number of projects we can work on simultaneously; in other words the size of the largest set $S \subseteq \{1, 2, ..., n\}$ such that $\Sigma_{x \in S} p_x \leq P$ and $\Sigma_{x \in S} c_x \leq C$. Your algorithm should run in time $O(nPC)$.

Distance-vector algorithm

Find shortest paths in a graph with negative edges, but without negative cycles.

Vertex cover

On a graph $G = (V, E)$, a vertex cover $C \subseteq V$ is a subset of the vertices such that for all $e = (u, v) \in E$, $u \in C$ or $v \in C$. That is, it is a set of vertices such that every edge has at least one of its endpoints in the vertex cover. The cost of a vertex cover is the size of set C.
Suppose you are given a bipartite graph $G = ((L \cup R), E)$ and wish to find a vertex cover of smallest cost.

1. Set up a min-cut in $G \cup \{s, t\}$ that does not cut any edges of E.
2. Show how to use this min-cut to get a vertex cover of minimum size for G. That is, show how to select a subset $l \subseteq L$ and a subset $r \subseteq R$, based on the cut, such that every edge is incident to some element of $l \cup r$.