 Introduction

* Intro to the IOI
* Ad hoc problems
* The order of time of an algorithm
* Basic sorting and searching
  + Linear search
  + Bubble sort
  + Selection sort
  + Insertion sort
  + Address sort
  + Why bubble sort is less efficient that the selection/insertion sort

 Recursion

* Generating permutations and combinations
* Divide and conquer
  + Application of divide and conquer to:
    - Binary search
    - Mergesort
    - Quicksort
    - Traveling salesman problem
  + Why (simple) divide and conquer can sometimes yield inefficient solutions
* Techniques for improving the efficiency of a recursive search
  + Iterative deepening
  + Branch and bound

 Graphs

* Explicit graphs
* Implicit graphs
* DFS
  + Finding a path thorough a graph using DFS
  + Finding all sub-graphs of a graph using DFS
  + Optimizing a DFS (for weighted graphs)
* BFS
  + Finding a path through a graph using BFS
  + Finding the shortest path through an unweighted graph
* Search space and Implicit graphs
  + The concept of search space
  + How to represent search space
  + Why it is usually takes too much space to represent implicit graphs using conventional representations (ex: adj. matrices)

 Dynamic programming

* The reasoning behind DP
  + Breaking down a recursive problem
  + Graph of solutions and sub-solutions
  + Exponential number of paths but linear number of nodes
  + Independence of sub-solutions
  + Storage to avoid recomputation
* Recognition of search space
  + The fact that the search space is effectively an implicit graph
  + Mechanical method to recognize the search space
* Evaluation of search space
  + Visualization of traversal of search space
  + Recursive evaluation (e.g. a memory function)
  + Iterative evaluation (e.g. true dynamic programming

 Finding the shortest paths

* Dijkstra's algorithm
  + Theory behind Dijkstra's algorithm
  + O(N 2) implementation
  + Adaptability of Dijkstra's algorithm
* Floyd's algorithm
  + Theory behind Floyd's algorithm
  + O(N 3) implementation
  + Adaptability of Floyd's algorithm
  + Using Floyd's algorithm to find a minimum cost cycle of minimum length
* Floyd vs. Dijkstra
  + Pros and cons of each algorithm

 Complete paths

* Eulerian paths
  + Identifying an Eulerian graph
  + Finding a Eulerian path
    - Recursive method
    - Iterative method

 Network flow

* Visualization
* FF maxflow method
  + Explanation of the FF method
  + Edmonds-Karp implementation of the FF method
* Adaptability of the FF method
  + Finding the maximal matching using the FF method
  + Finding the minimal cutset using the FF method
  + Maximal disjoint paths using the FF method

 Disconnecting graphs

* Disconnecting via edges
  + Finding bridges
  + Eliminating all bridges in a graph by adding as few edges as possible (can be done in Polynomial time)
* Disconnecting via nodes
  + Finding articulation points
  + Eliminating all articulation points in a graph by adding as few edges as possible (is NP-Complete)
* Biconnectivity of graphs
  + Finding the biconnected components of graphs
  + How the collapsible set structure is used to yield an amortized time complexity of O(N lg N)
* Mincut type disconnection

 Minimum cost spanning trees

* MCST theorem
* Prim's algorithm
  + How Prim's algorithm makes use of the MCST theorem
  + Prim's innovative data structure to keep track of the nodes to be added to the MCST
  + O(N 2) implementation of Prim's algorithm
* Kruskal's algorithm
  + How Kruskal's algorithm makes use of the MCST theorem
  + The collapsible set structure used in Kruskal's algorithm
  + O(E lg E) implementation of Kruskal's algorithm
* Adaptability of Kruskal's and Prim's algorithms
  + The fact that it is more or less impossible to introduce new constraints to these algorithms without breaking them.
* Kruskal vs. Prim
  + Pros and cons of each algorithm