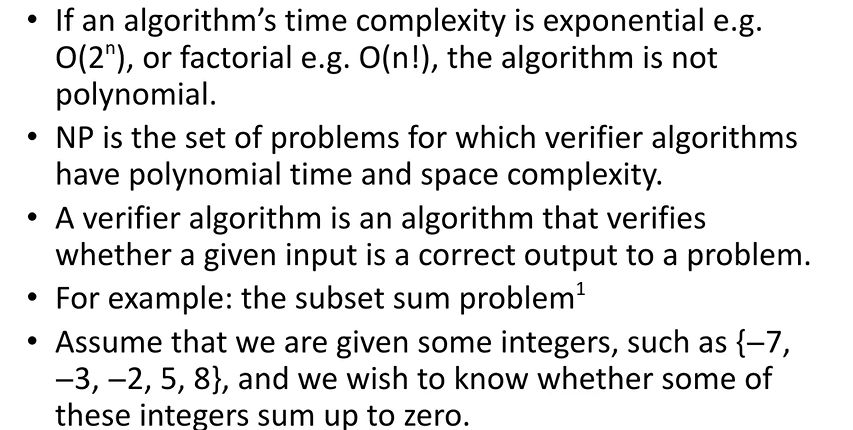
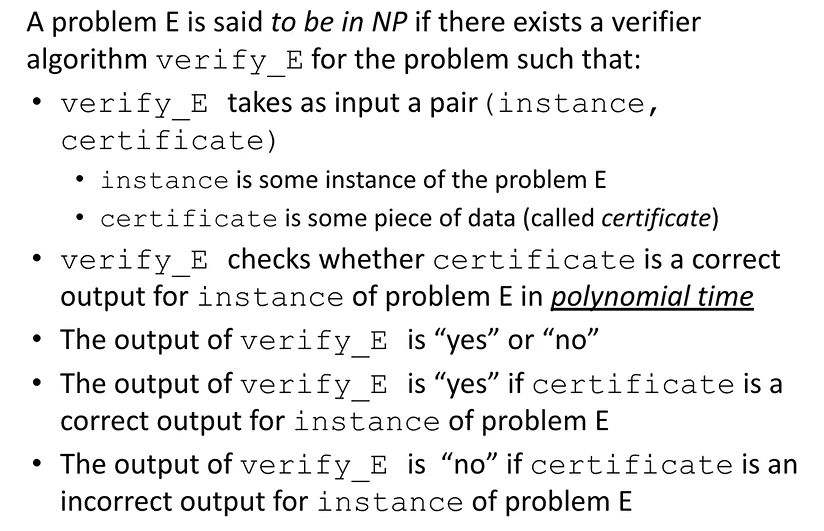
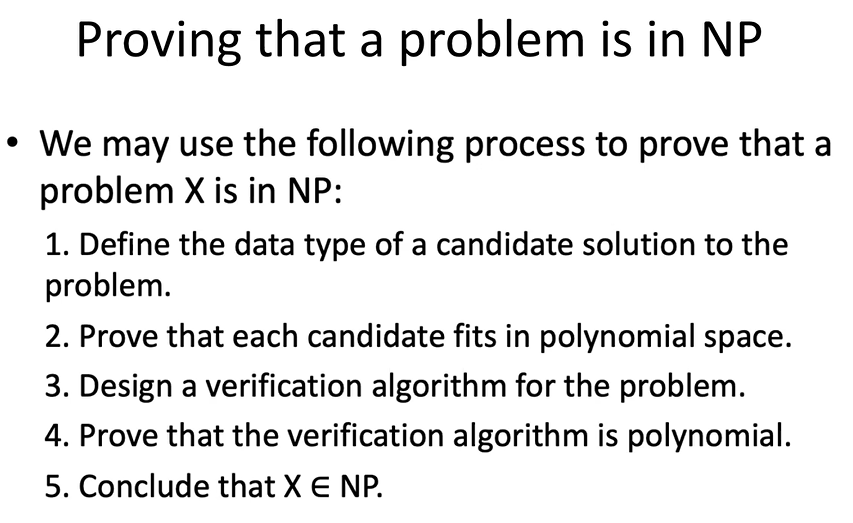
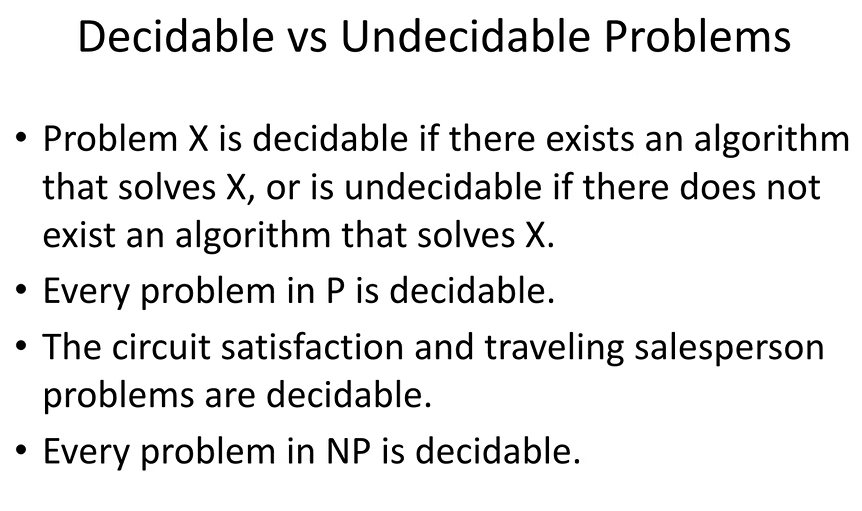
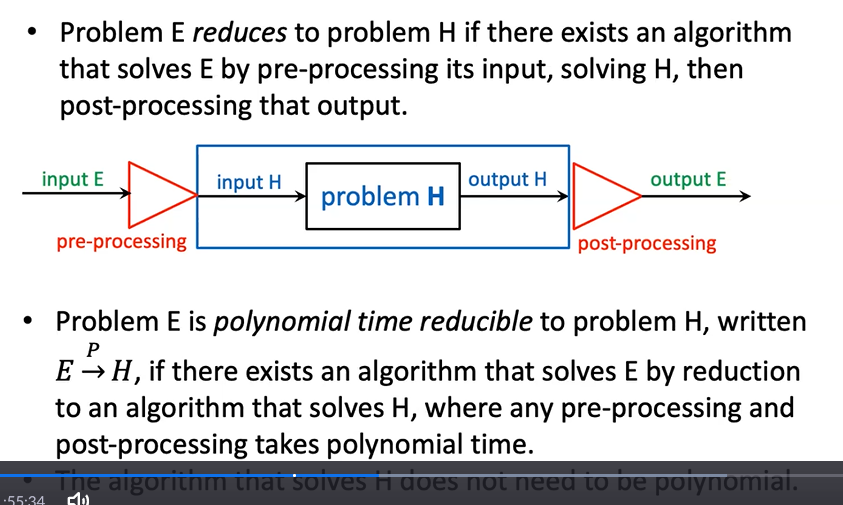
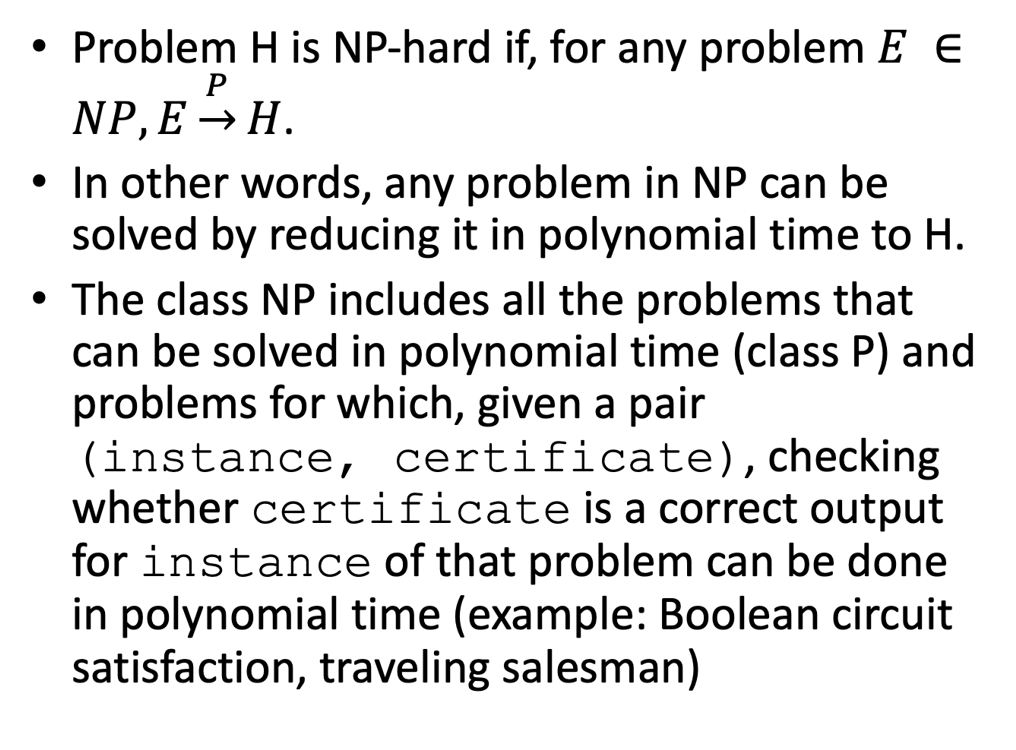
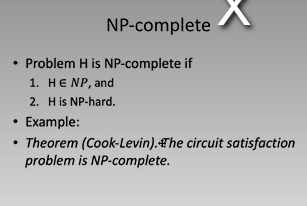
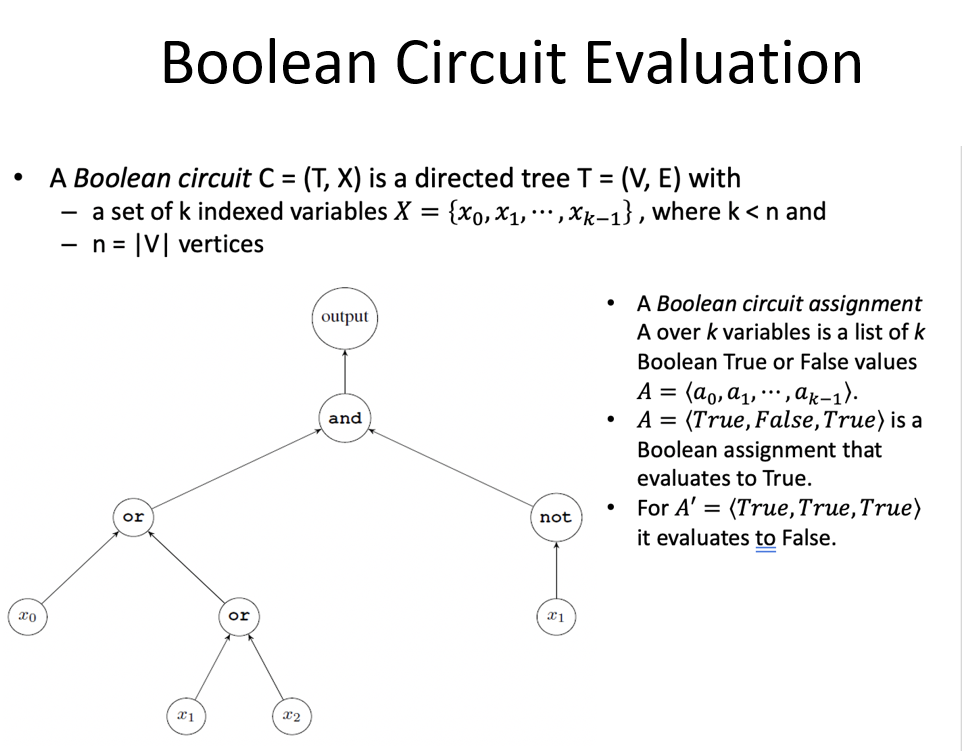
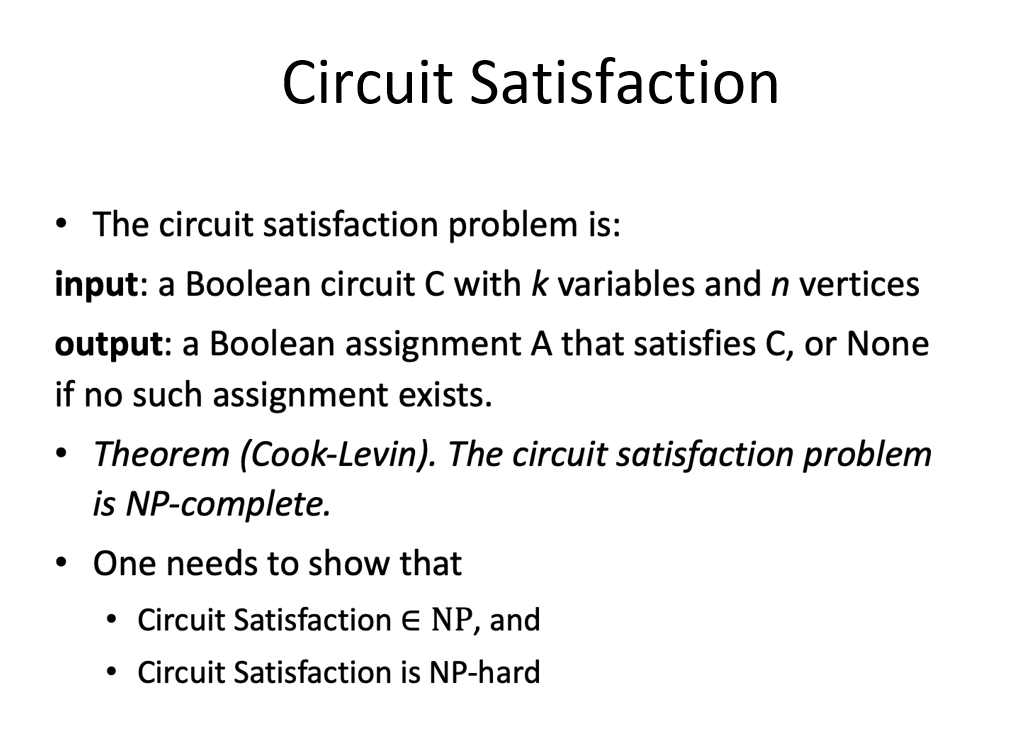
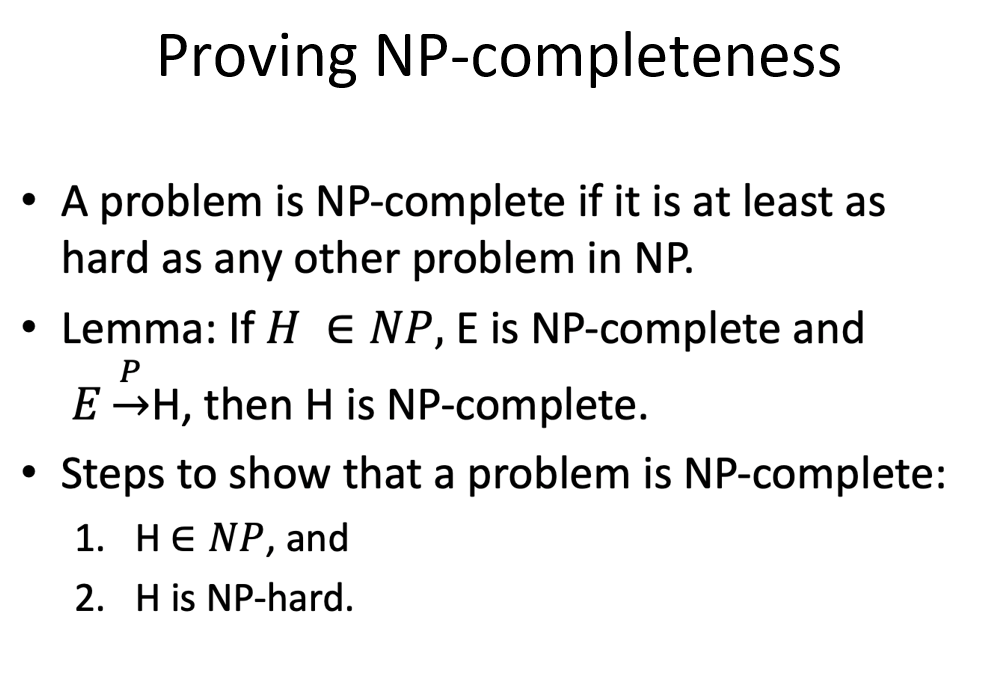
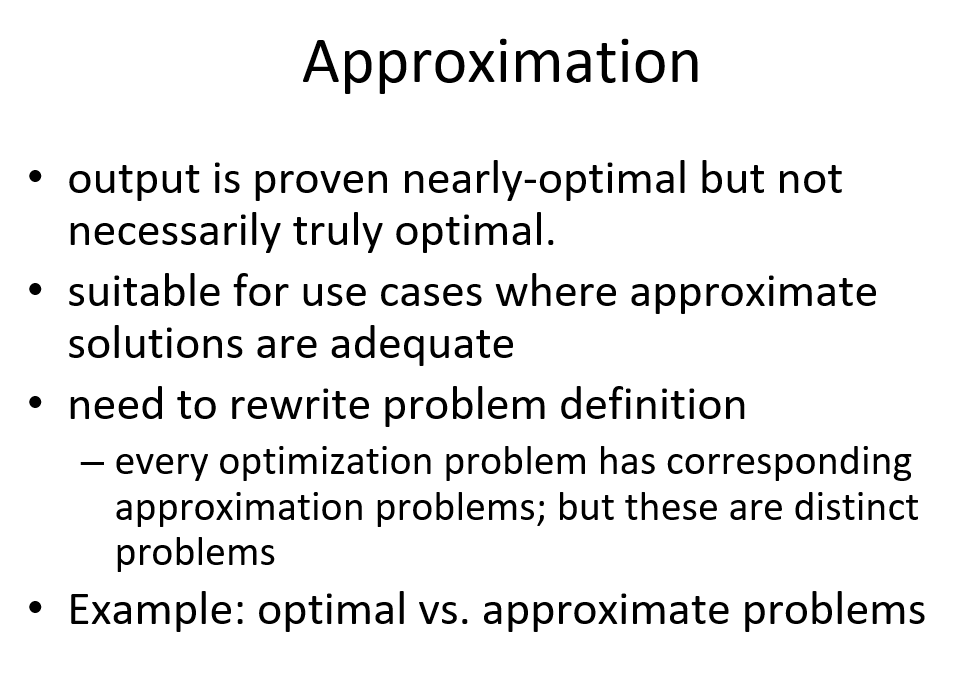
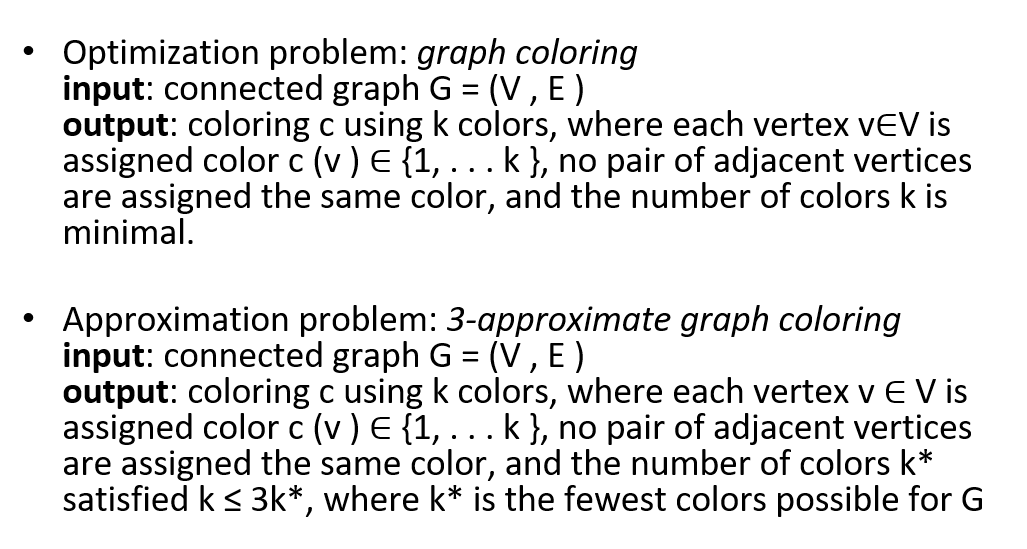
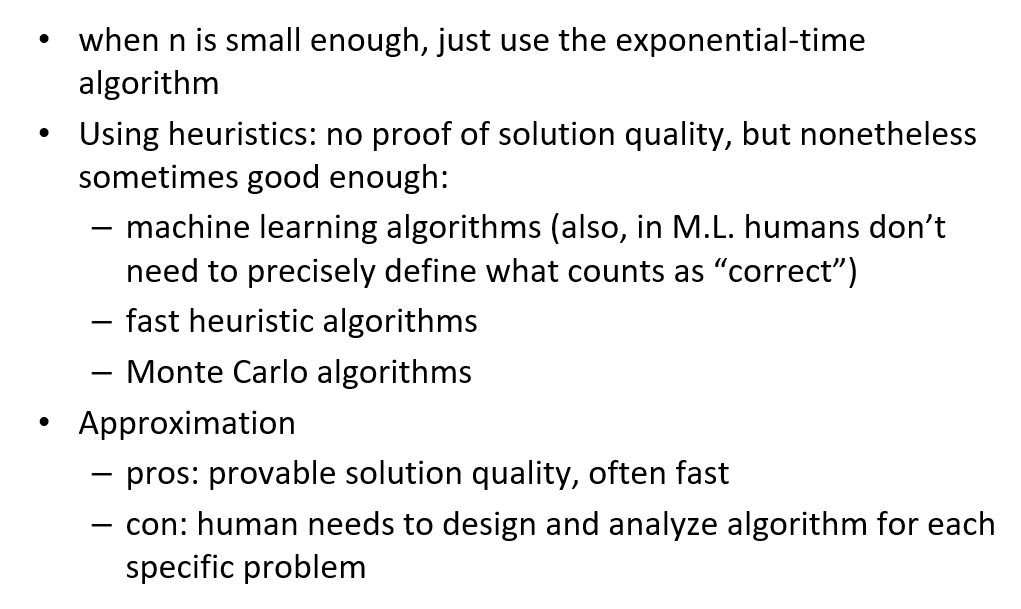
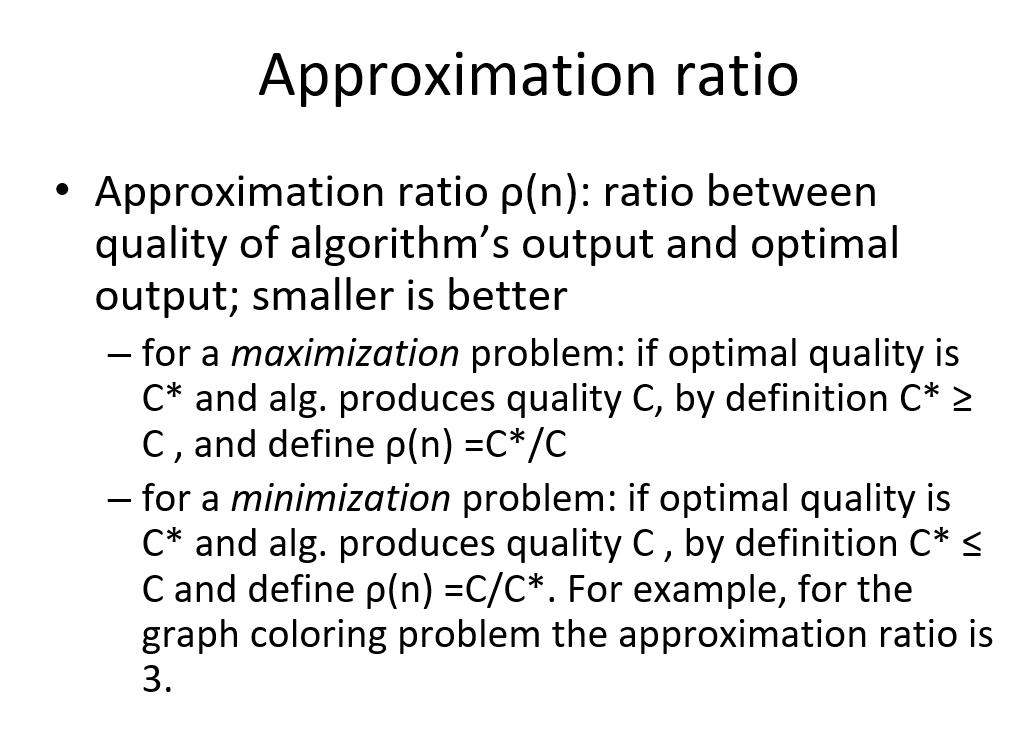
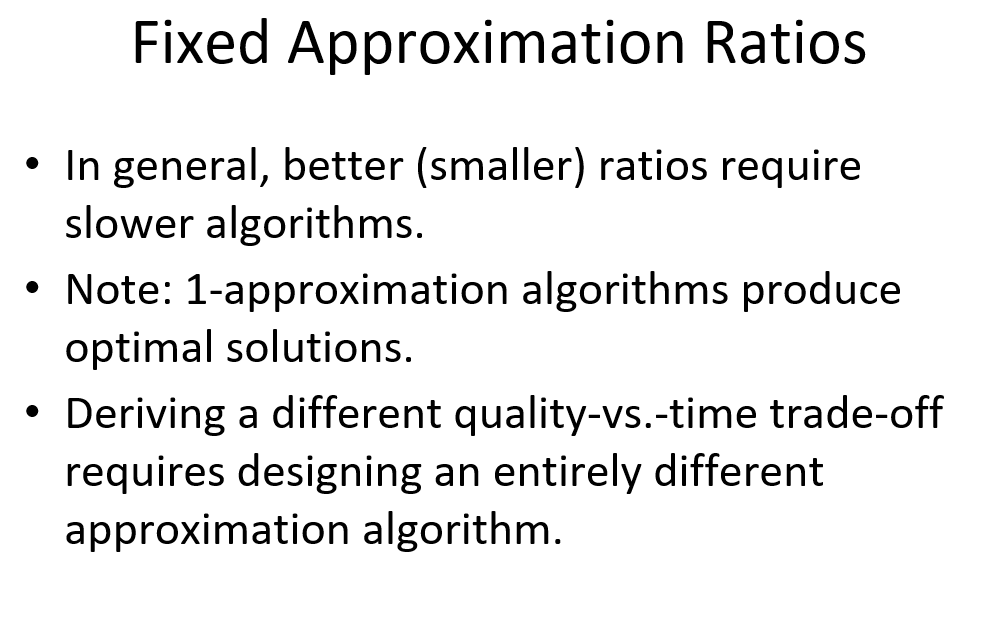
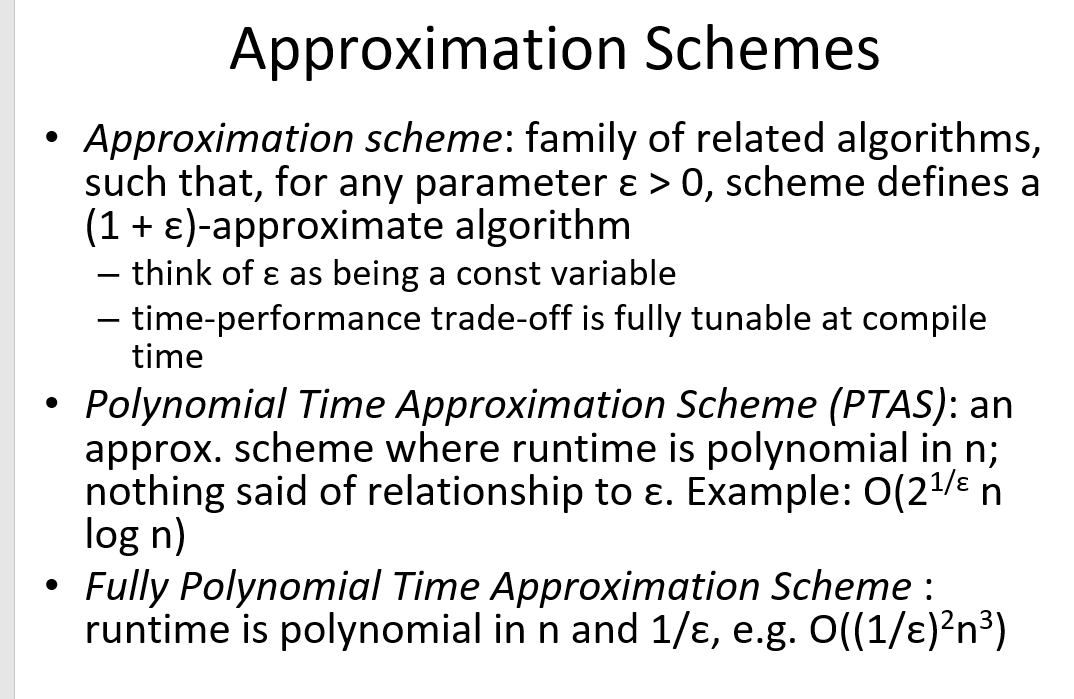
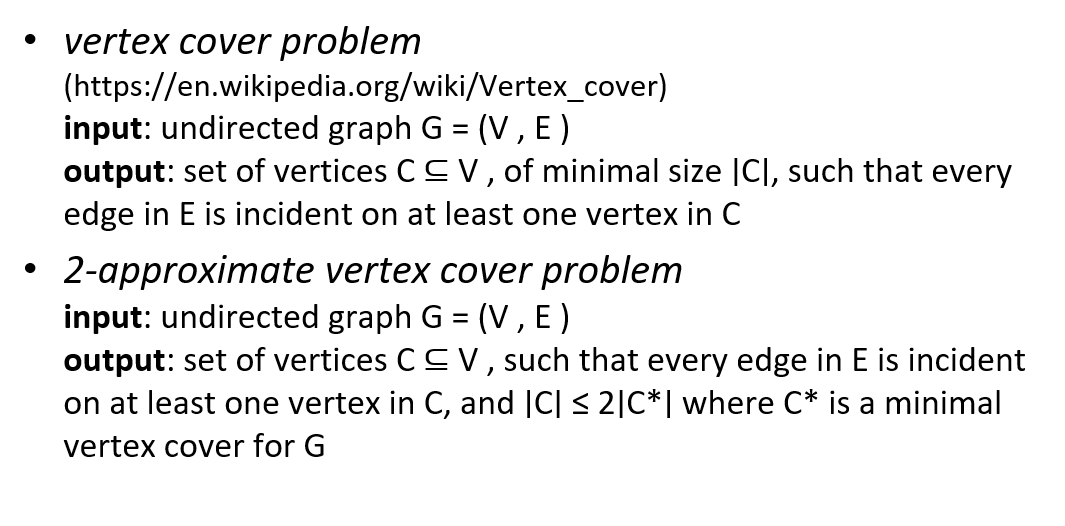
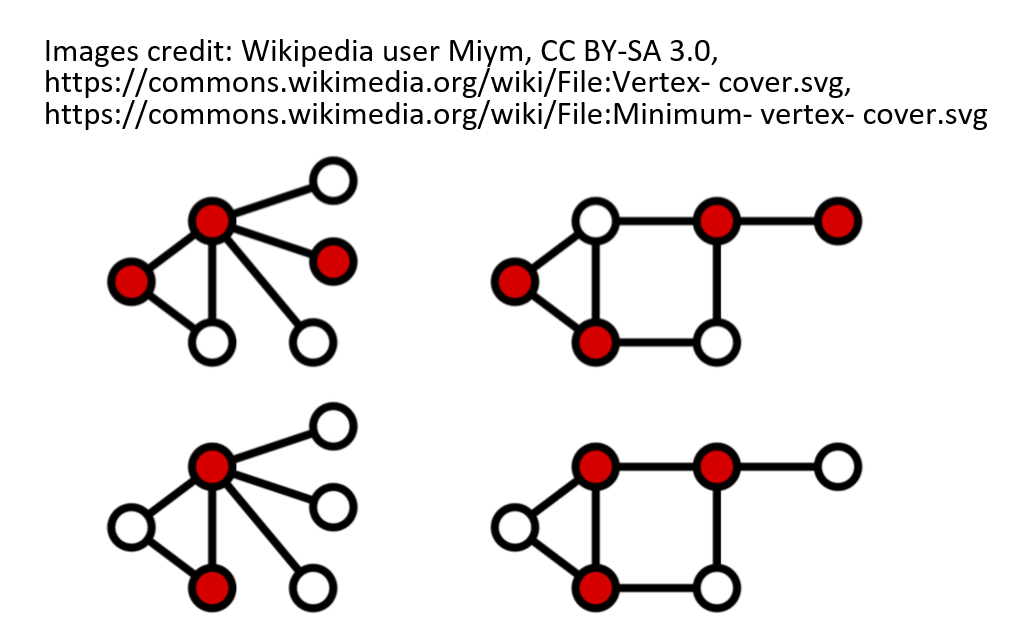
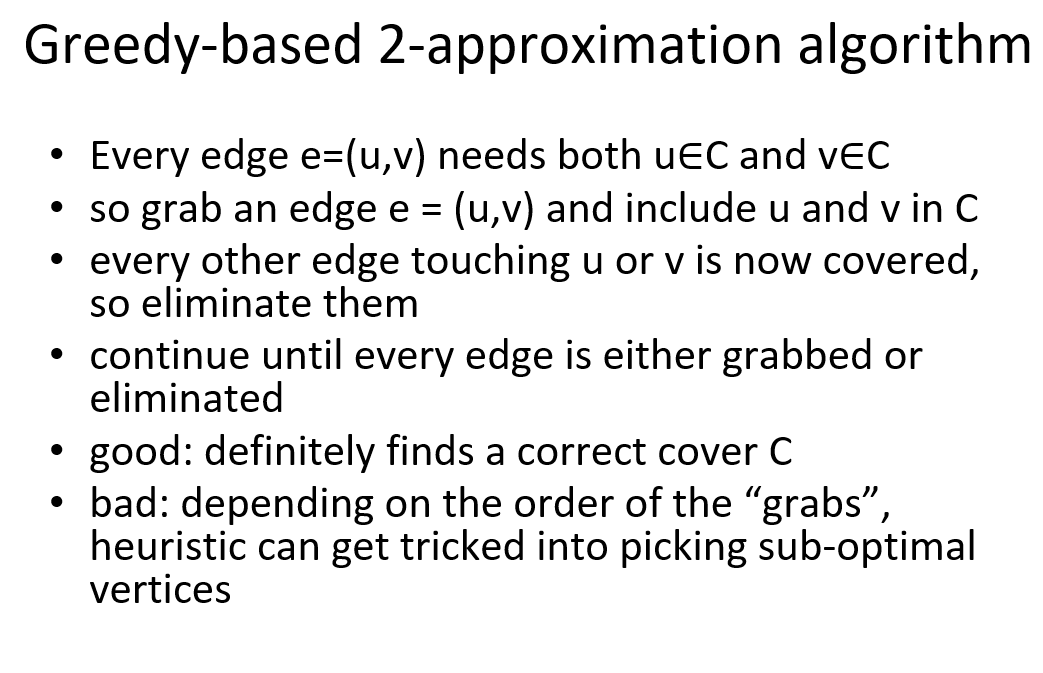
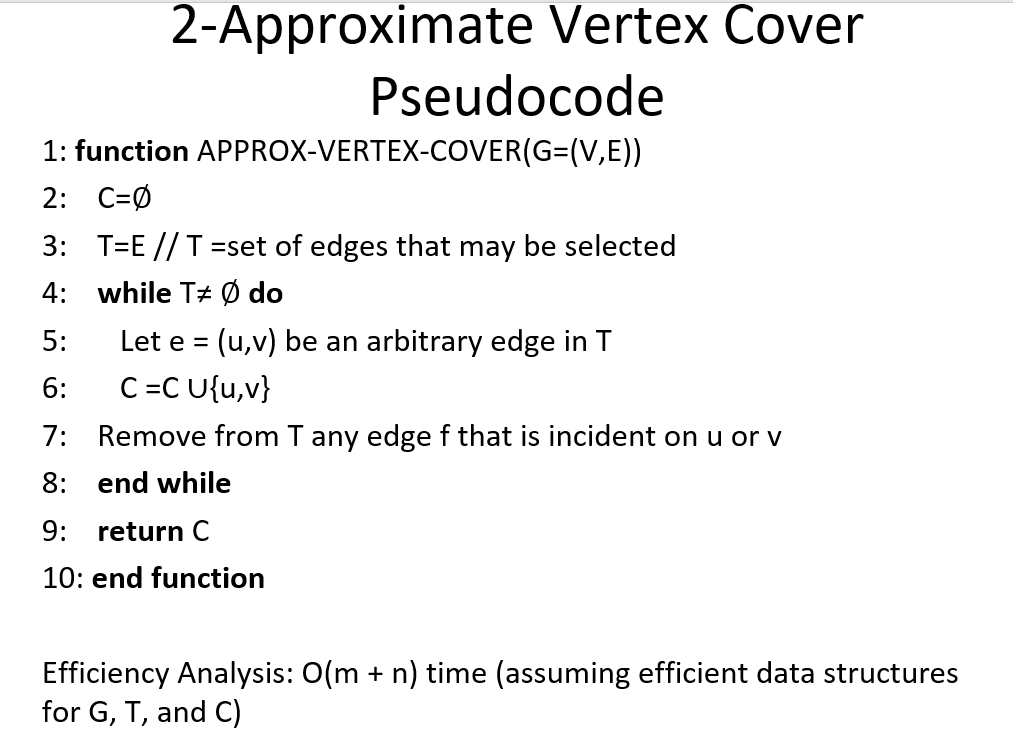
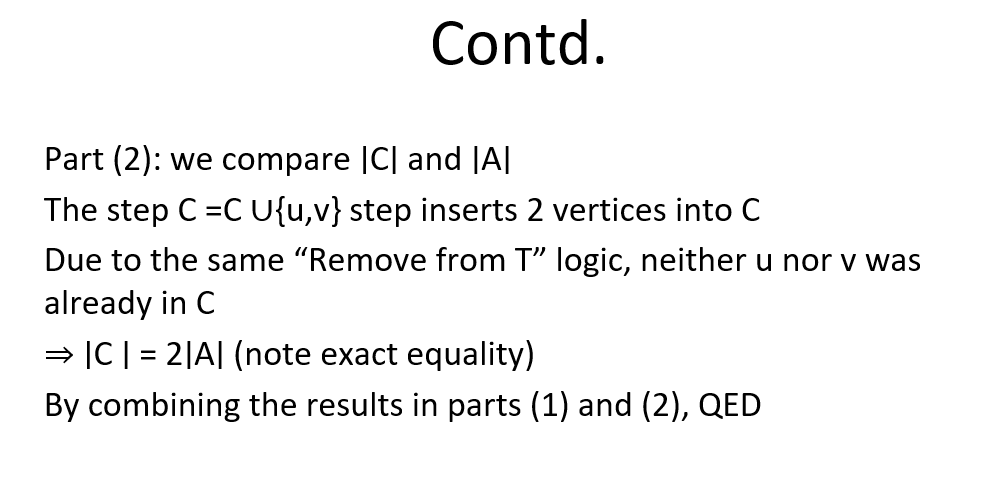
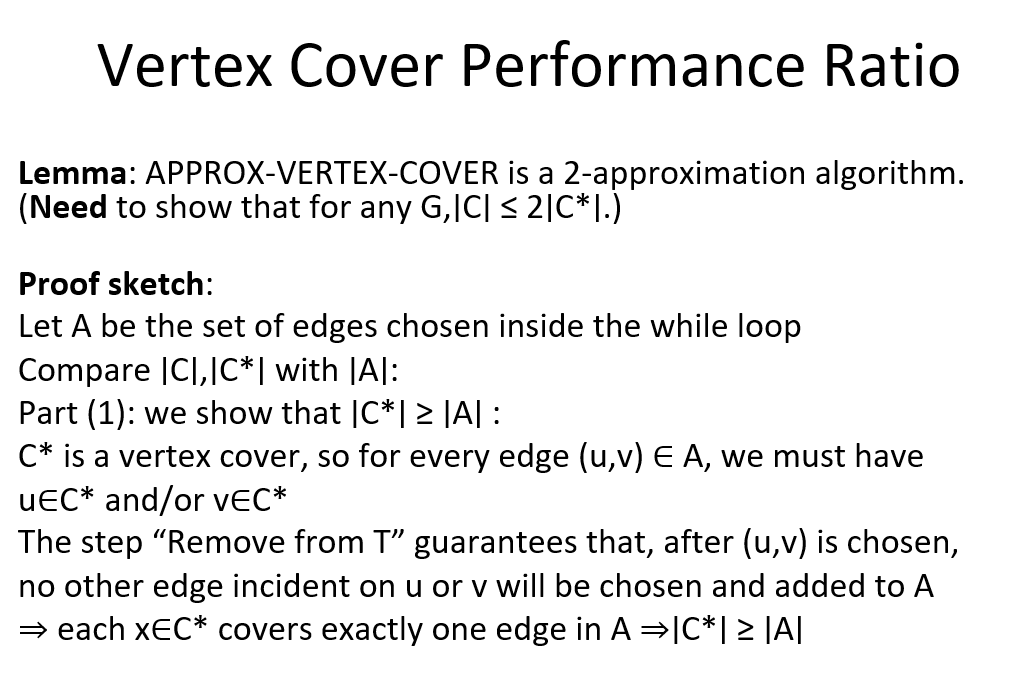
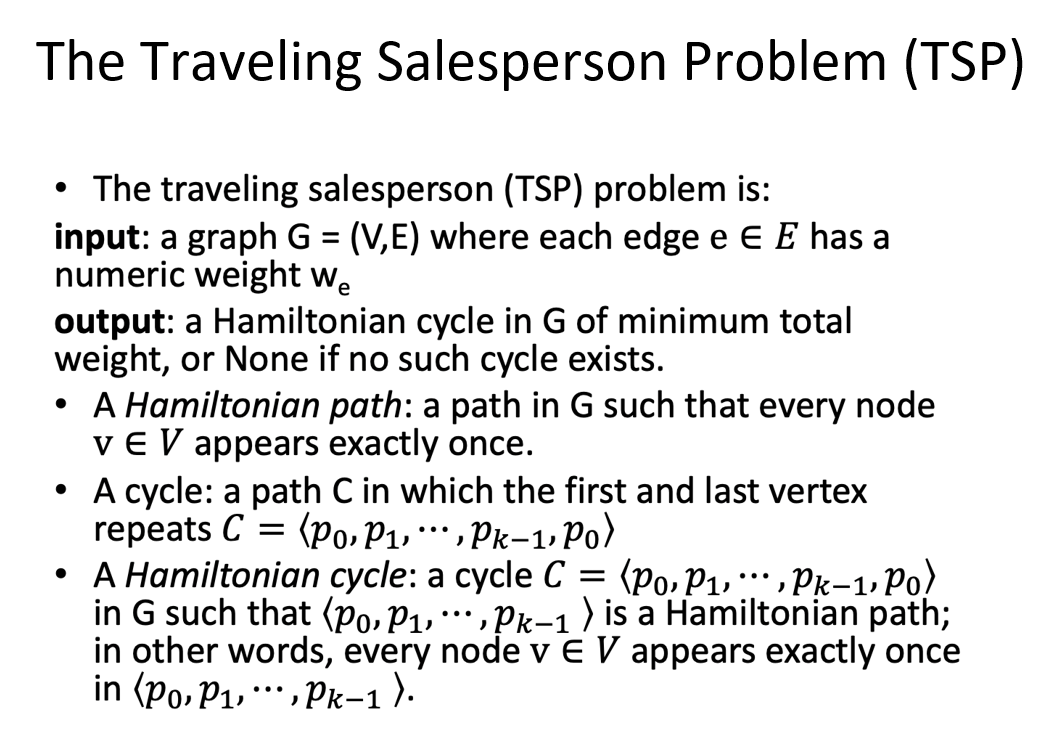
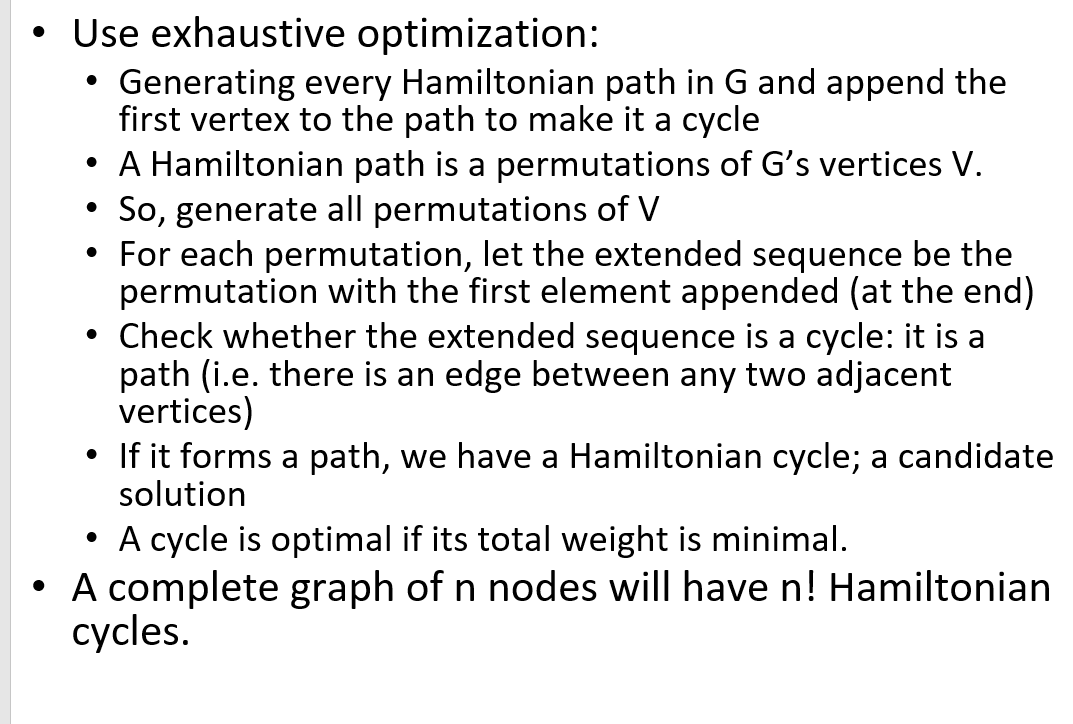
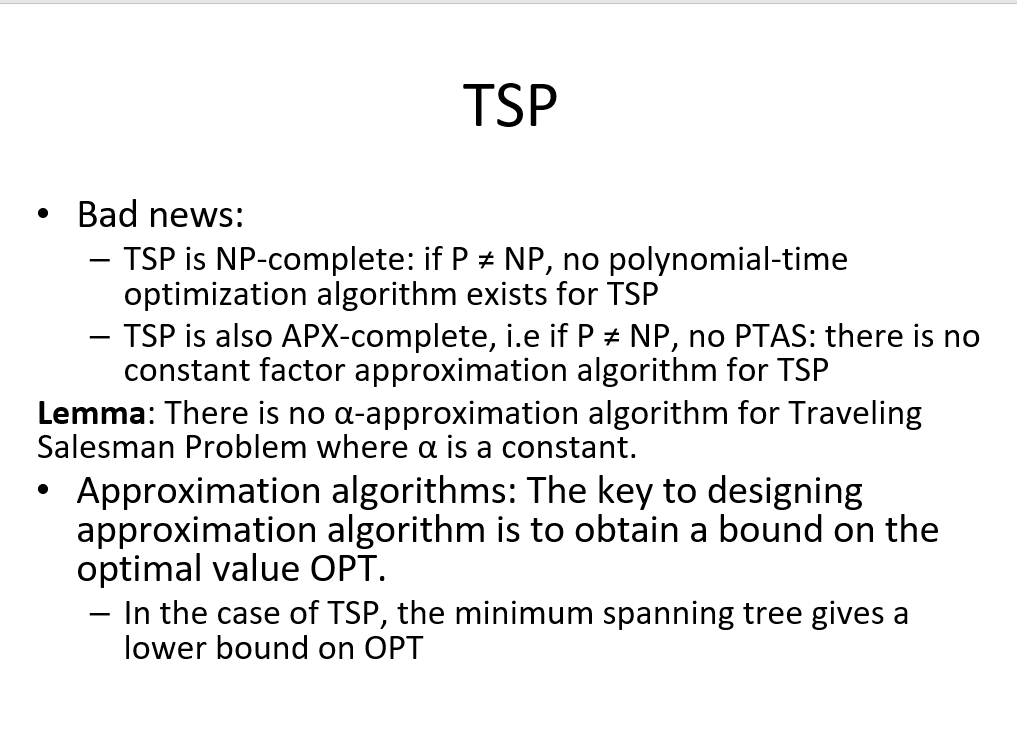
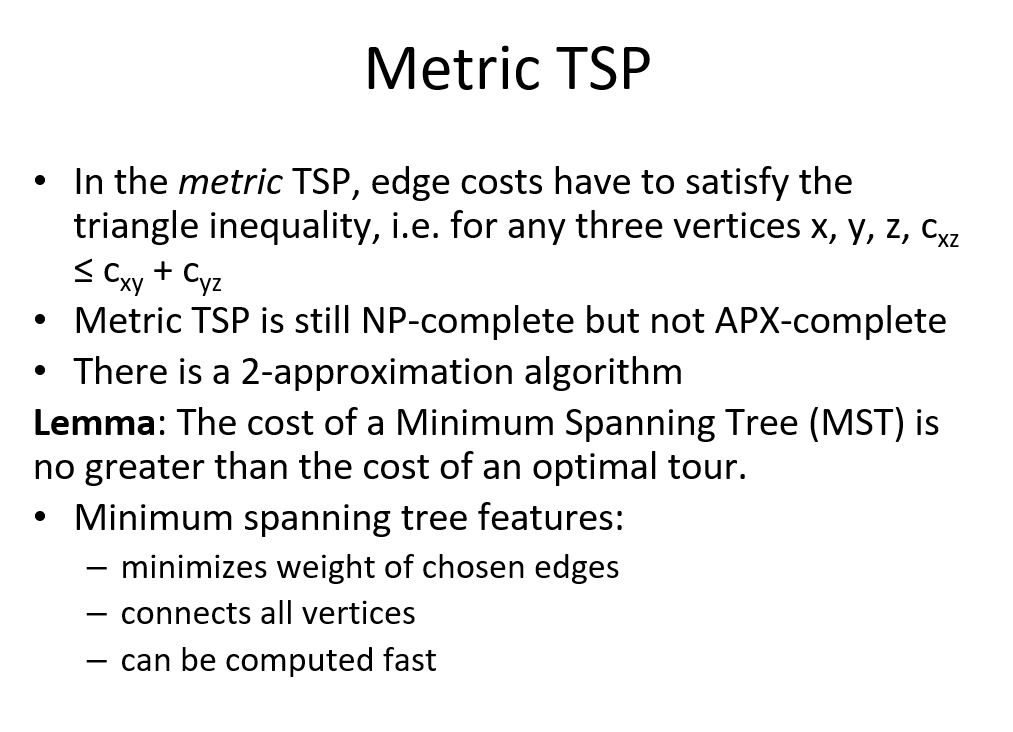
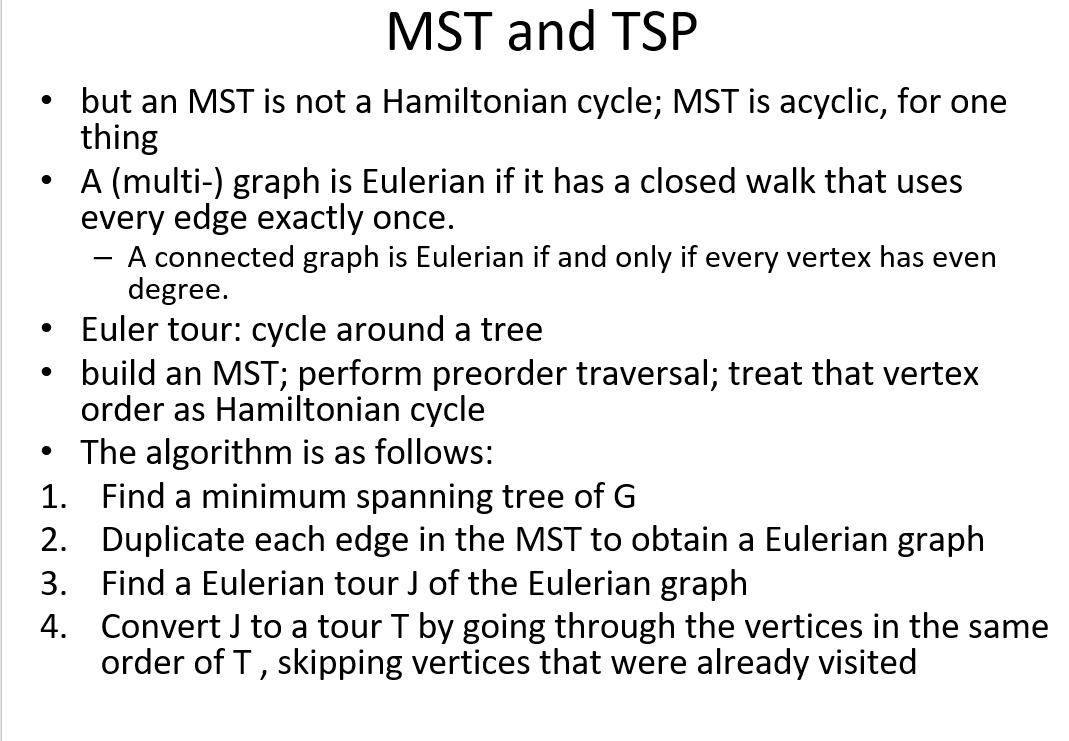
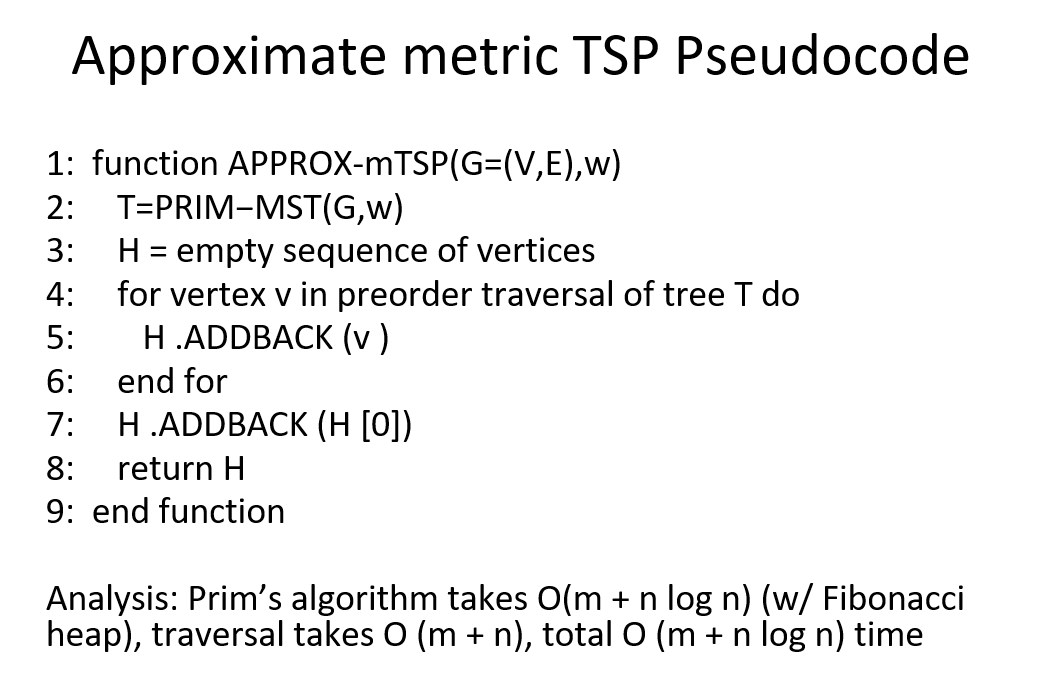
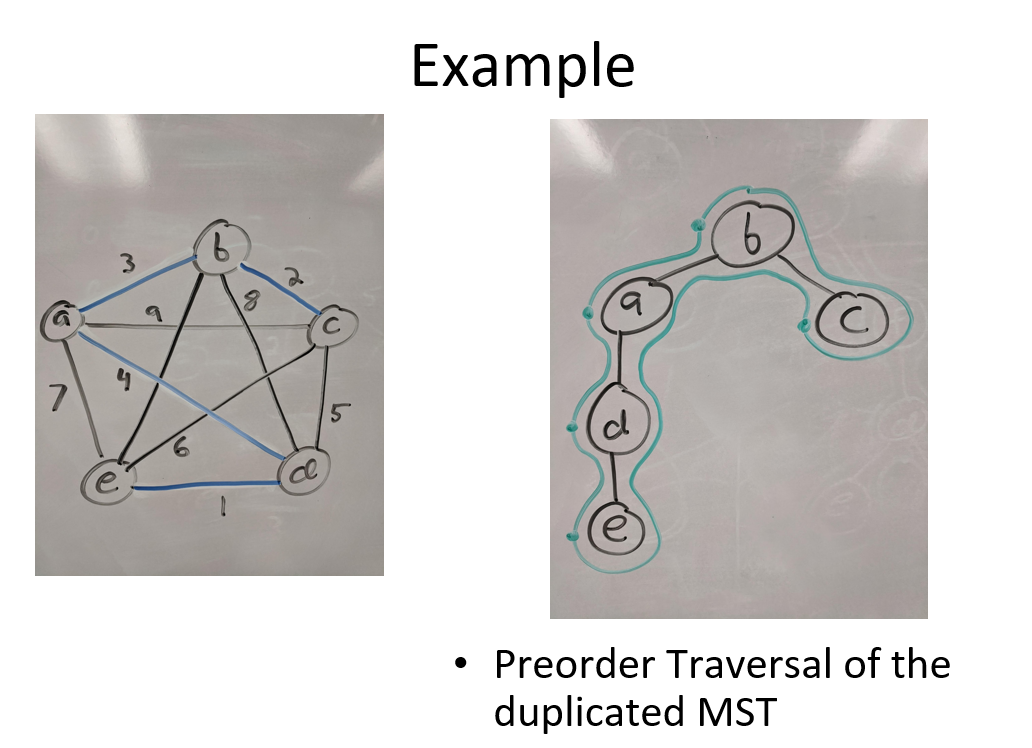
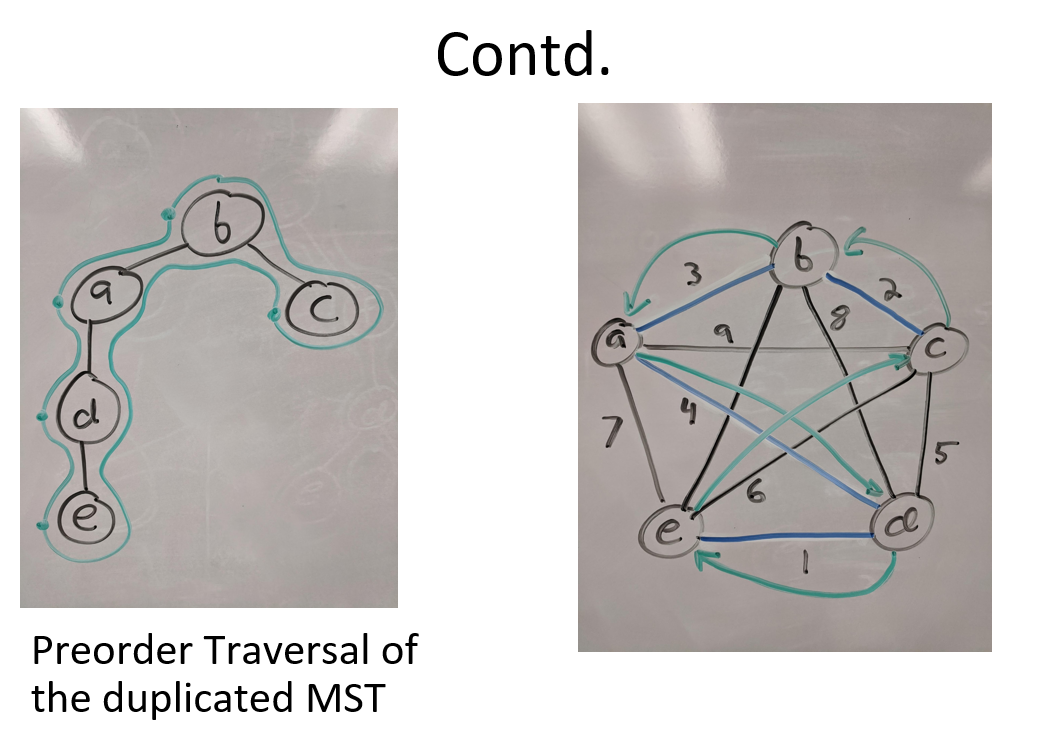
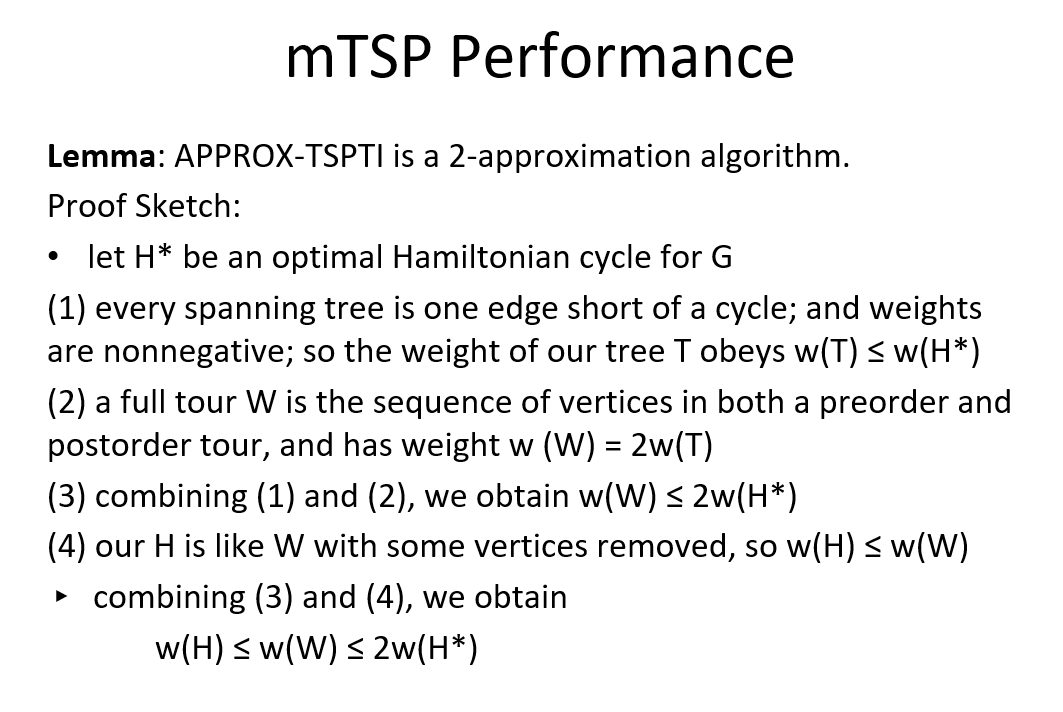
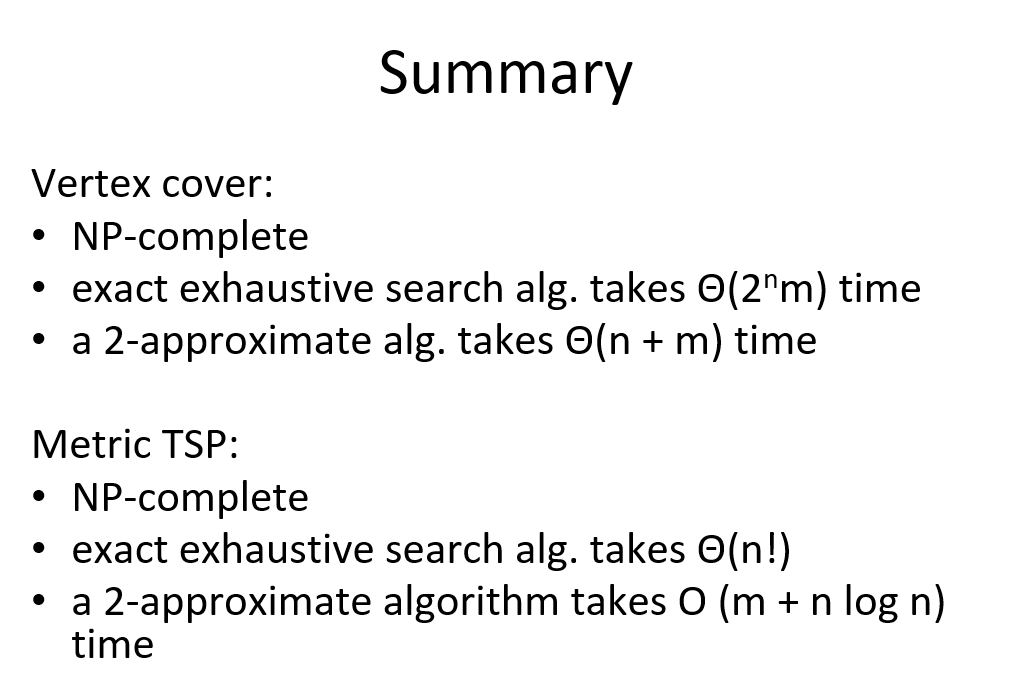
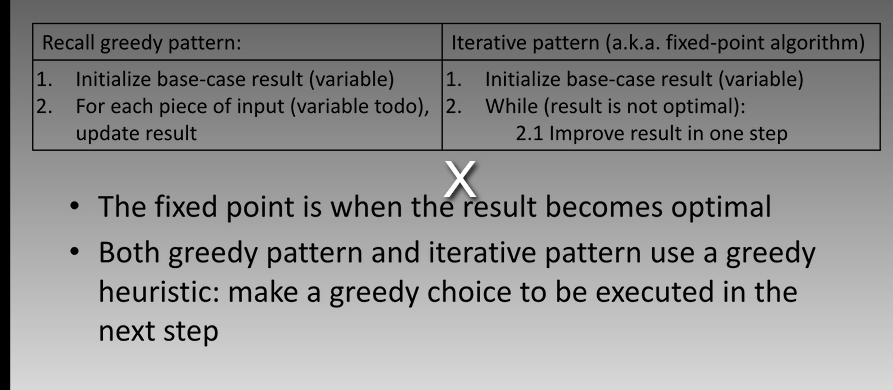
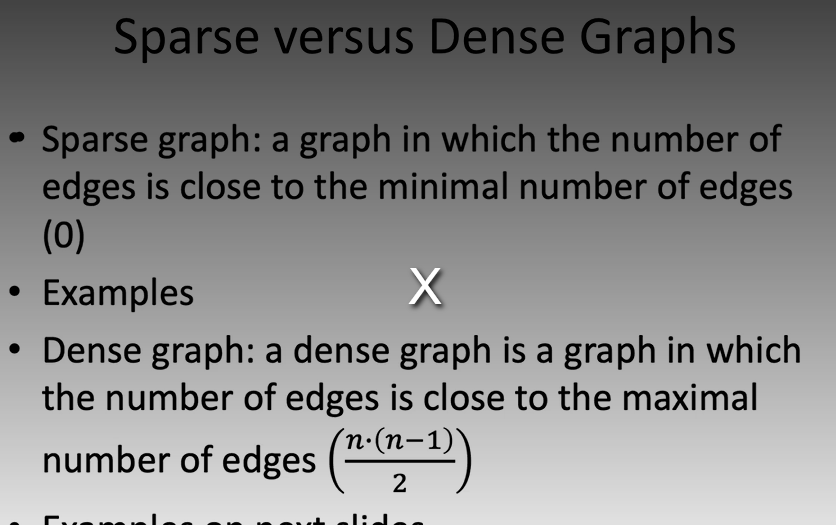
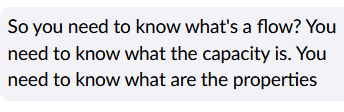
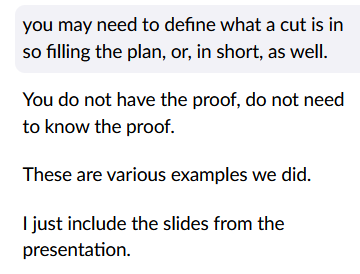
CPSC 535 Review

# Based off of the Final Review Session of 12/7/2022

1. NP Class Problems
   1. If you can verify whether a vanditate is correct, it’s solution nto the problem in polynomial time  
      
   2. Verify\_E is not optimization  
      
      1. We can check to see if they form a Hamiltonian cycle or no
      2. Responsible for checking to see if it fits solution THEN optimization comes separately
      3. For circuit satisfaction: it’s either true or falcse ofr list of values
      4. If it’s vertex cover problem, given set of colors, we want to know if whatever solution we have is eligible for next check in optimization
   3. Proving that problem is in NP  
      
      1. We are not solving problem
      2. Just generate the candidates either using permutations or stop-sets
         1. However even if we verify in polynomial time, soluation may not be ppolynomial
   4. Decidable vs undecidable problems  
        
      Every problem in NP is decidable
      * 1. Problem in P is decidable
        2. Circuit satisfaction nand TSP problems decidable
        3. Every problem in NP is decidable
      1. Undecideable problems
         1. Numerical data for input but problem asking to solve has no connection to the input
         2. i.e. how much time it takes to go to school based off of what you had for breakfast
   5. NP-Hard Problems
      1. 
      2. Problems which are NP-hard but NOT in calss NP
      3. Ex. Graph isomorphism
         1. Sometimes quasi-polynomial meaning you are given 2 drawings of a graph or given a graph set of notes and set of vertices
         2. If same number of nodes, how to tell if isomorphic or not?
         3. So for each set of vertices, generate all possible permtuations of second graph and see whether they map correctly
         4. How mayny permutations?
            1. N-factorio
            2. Proven in polynomial time? No
      4. Problem H
         1. Problem H solves everything
         2. Instead of generating algo or solution from scratch, convert input to problem E which then converts it to input H
         3. Reduce problem E to problem H
         4. If conversion pre-rpocessing of input and post-processing can be done in polyniomal time, it can be reduced tin polynomial time to H



* 1. NP-Complete
     1. 
  2. Boolean Circuit Evaluation  
     
     1. 
     2. Will not be on final exam
  3. Proving NP-Completeness (aka part 2)  
     

1. Approximation
   1. 
      1. For this you need to rewrite the problem definition i.e. for the traveling salesman problem
      2. Inputs start with same output type but then there are different specifications
      3. Example1: Graph coloring
         1. i.e. a graph that cannot be colored with 2 colors
         2. 
         3. 
         4. We can do it with three colors: Red, Yellow, Blue
         5. But approximation will be no more than 3x3 = 9 colors
         6. This is our upper bounds
   2. Strategies to solve NP-Hard problems
      1. 
   3. Approximation ratio
      1. 
      2. 
      3. 
      4. Will not need to go into too much detail
   4. Vertex Cover problem
      1. 
      2. Given a graph, solve for connected components
      3. Example  
         
         1. You have 6 nodes, color with three
         2. Not ideal so you reduce it down
         3. The bottom is the ideal one
      4. Vertex cover is NP-compelte which means currently has exponential time solution
      5. Baseline algo exhaustive search generates a set of vertices which is a power set
      6. It will be 2^nxm for optimized
   5. Greedy-based 2-approximation algo
      1. 
      2. 
         1. Need to show this in polynomial time
         2. State of edges = polynomial time
      3. 
         1. Show that whatever we select never exceeds 2x more
      4. 
         1. 
         2. 
   6. 
      1. 
      2. 
      3. 
      4. 
   7. 
      1. 
2. 
3. Final Exam
   1. No questions from before midterm 3
   2. Will always have a design algo
   3. Based on hashing or dynamic programming
   4. 
   5. 
      1. T/F, finn-in-bblank
   6. Djikstra, bellman ford = expect to see in short answer
   7. Maximum flow
      1. Fill in the blank
   8. Maximum bitrate can show up in short word
      1. Larger graph in algo design
   9. Properties of flow network
      1. Short answer
      2. 
   10. Augmenting path
       1. Edmonds-Karp based on assignment 8
       2. 
   11. Residiual and Moth graph
       1. 

# Lecture 11/30/2022