Presentations

- Start next Thursday
- Come prepared to present with slides
- Target length 6-7 minutes
- Summarize only the key critical concepts
- Keep text minimal and use figures
- Demo software if possible within timeframe
- Class will be fully remote on presentation days

P vs NP

- Polynomial time
    - $O(n^k)$ for some constant k
    - All the algorithms we have discussed run in polynomial time
- Exponential time
    - $O(k^n)$ for some constant k
    - Intractable in the worst case - can take longer than a lifetime even for relatively small n

- Factorial time
  - $O(n!)$
  - Even worse!
  - Happens most commonly when you need to generate all permutations

ABC	BCA
ACB	CAB
BAC	CBA

- Problem: Given a graph containing vertices s and t, is there a path from s to t?

  - Use breadth-first search starting at s and see if we can get to t

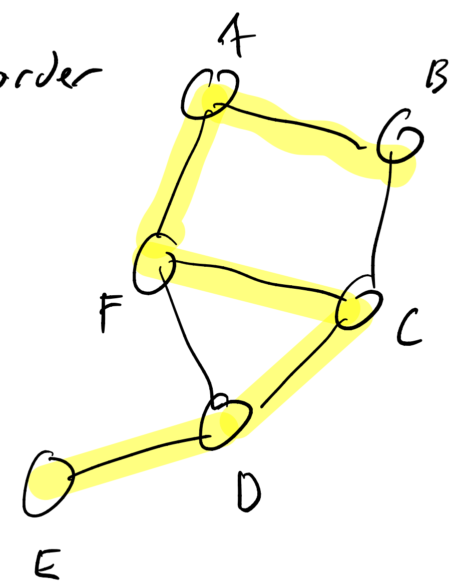
  - BFS is polynomial time

- Problem: Given a graph, is there a path that visits every vertex exactly once?

  - Depth-first search could find it, but we have to choose the neighbors in the right order

  - One way to do a brute-force search would be to try all possible ways of doing a DFS

    - Exponential

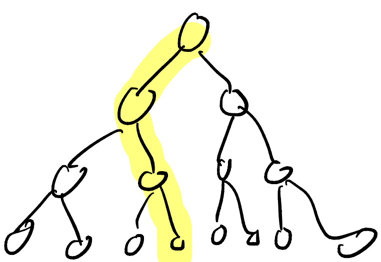


- Even more naive brute force
    - Generate all permutations of vertices and check each one to see if it's a path
        - Factorial number of permutations
        - $O(V)$ to check one permutation
- This type of path is called a Hamiltonian Path

- Nondeterministic Polynomial (NP)
    - A nondeterministic machine is theoretical
    - If there are multiple branches in the search for a solution, the machine can try each one simultaneously

      or

      the machine always chooses the right branch on the first try



- For Hamiltonian path – a depth first search that always chooses the correct neighbor first will find a Ham. path in polynomial time
- An NP problem can be solved in polynomial time on a nondeterministic machine
- A solution to a NP problem can be verified in polynomial time on a deterministic machine (a real machine)

- Decision problems
  - Output is "yes" or "no"
  - often easier to study than optimization problems
- Traveling Salesman Problem

  (least weighted)
  - Optimization version: what is the shortest Hamiltonian path in a graph?
  - Decision version: Is there a Ham. path with weight $\leq k$?

- Reduction
    - Transform an instance of one problem into an instance of another problem
    - Use an algorithm for one problem to solve another problem
    - A reduces to B if a solution to B can solve A

- Boolean Satisfiability problem (SAT)

  - Given a Boolean formula, is there a set of assignments to the variables that makes the expression true?

$a \wedge b$

$a = 1 \quad b = 1$

$a \wedge (\neg a)$

no assignment possible

$\wedge$ - and

$\vee$ - or

$\neg$ - not

- Cook-Levin Theorem
  - SAT is in NP, since a solution can be verified in polynomial time
    - Assign the values and check
  - All problems in NP can be reduced to SAT in polynomial time
    - Complicated proof which requires more theory of computation background

- NP-complete problem
    - Decision problem
    - In NP (verifyable in poly. time)
    - All NP problems can be reduced to it
    - SAT is NP-complete via Cook-Levin theorem
    - To show a new problem is NP-complete
        - Show it is in NP
        - Show another NP-complete problem reduces to it in polynomial time

- Can NP problems be solved in polynomial time? Does P = NP?
  - We don't think so, but nobody has proven that they can't
  - If any one NP-complete problem is shown to have a polynomial time solution, we know P = NP

As far as we know

But maybe