**CS350 – P = NP?**

**Purpose:** There are several important complexity class of problems: Poly, PolyCheck/NPoly, and Expo. If we restrict ourselves to decisions problems then we have the complexity classes P, NP, and Exp. These classes serve as ways to describe problems that are tractable, verifiable, and practical only for small inputs. The purpose of this module is to become comfortable with these class definitions.

**Knowledge:** This module will help you become familiar with the following content knowledge:

- How to determine whether a problem is in Poly or Expo.
- How to determine whether a problem is in PolyCheck and/or NPoly.

**Activity:** With your group perform the following tasks and answer the questions. You will be reporting your answers back to the class in 60 minutes.

1. We start with the definition of the complexity class Poly:

   Poly is the set of computational problems that can be solved by a program with a running time of $O(n^k)$ for some $k \geq 0$

   Explain why the problem All3Sets is in Poly.

   **Problem** ALL3SETS

   - **Input**: A list of distinct integers in decimal notation separated by whitespace. Example: "`2 5 8 9`".
   - **Solution**: A list of all 3-element subsets (i.e., 3-*sets*) that can be formed from the input. The elements of each set are separated by commas, and the sets are surrounded by braces. Whitespace may be included wherever convenient. Example: On input "`2 5 8 9`", a solution is

     `{2,5,8} {2,5,9} {2,8,9} {5,8,9}`

2. Explain why the ShortestPath problem is in Poly

   **Problem** SHORTESTPATH

   - **Input**: An undirected, weighted graph $G$ and two of $G$'s nodes $v$, $w$ (the *source* and the *destination*). $G$, $v$, and $w$ are separated by semicolons and optional whitespace. Usually, we also assume the weights are positive. Example based on figure 11.5: "`a,b,3 a,c,5 a,d,6 b,c,1 b,d,2 c,d,9 ; a ; c`".
   - **Solution**: A shortest path starting at $v$ and ending at $w$, or "`no`" if none exists. Example: "`a,b,c`" is a solution to the above input.

3. Another definition is the complexity class Expo:

   Expo is the set of computational problems that can be solved by a program with a running time of $O(2^{p(n)})$ for some polynomial $p(n)$

   Explain why the problem AllSubsets is in Expo. Is it possible that this problem is in Poly as well?

**Problem** AllSubsets

- **Input**: A list of distinct integers in decimal notation separated by whitespace. Example: "`2 5 8 9`".
- **Solution**: A list of all subsets that can be formed from the input. The elements of each set are separated by commas, and the sets are surrounded by braces. Whitespace may be included wherever convenient. Example: On input "`2 5 8 9`", a solution is

  `{} {2} {5} {8} {9} {2,5} {2,8} {2,9} {5,8} {5,9} {8,9}`
  `{2,5,8} {2,5,9} {2,8,9} {5,8,9} {2,5,8,9}`

4. Explain why the TSPPath problem is in Expo. Is it possible that this problem is in Poly as well?
   A Hamilton path visits every vertex exactly once.

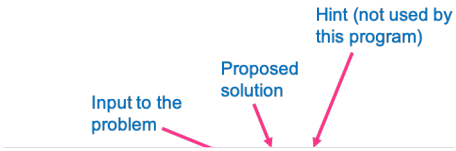
**Problem** TSPPath

- **Input**: An undirected, weighted graph $G$ and two of $G$'s nodes $v$, $w$ (the *source* and the *destination*). $G$, $v$, and $w$ are separated by semicolons and optional whitespace. Example based on figure 11.5: "`a,b,3 a,c,5 a,d,6 b,c,1 b,d,2 c,d,9 ; a ; c`".
- **Solution**: A shortest Hamilton path starting at $v$ and ending at $w$, or "`no`" if none exists. Example: "`a,d,b,c`" is a solution to the above input.

5. Yet another definition is the complexity class PolyCheck:

   A computational problem is in PolyCheck if, when given a proposed solution, a computer program to can check (or verify) that the solution is correct in polynomial time.

   Here is a verifier for a problem of finding a factor of an integer. Explain why Factor is in PolyCheck.




```
def verifyFactor(I, S, H):
    if S == 'no': return 'unsure'
    M = int(I); m = int(S)
    if m>=2 and m<M and M % m == 0:
        # m is a nontrivial factor of M
        return 'correct'
    else:
        # m is not a nontrivial factor of M
        return 'unsure'
```

6. The verifier may also require a hint. To see why we may need a hint, consider the problem TSPD:

**Problem** TSPD

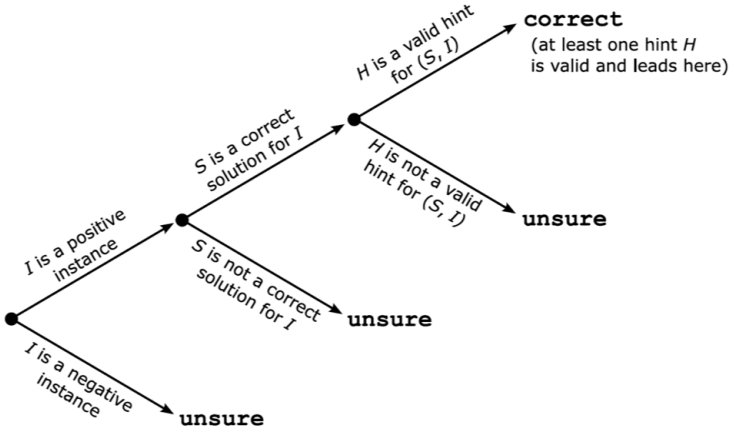
- **Input**: An undirected, weighted graph $G$ and integer threshold $L$ separated by a semicolon. Example: "`a,b,5 b,c,6 c,a,3 ; 20`".
- **Solution**: "`yes`" if $G$ has a Hamilton cycle of length at most $L$, and "`no`" otherwise. Example: With the above input, the solution is "`yes`", because "`a,b,c`" is a Hamilton cycle of length 14, and $14 \leqslant 20$.

```
def verifyTspD(I, S, H):
    if S == 'no': return 'unsure'
    # extract G,L from I, and convert to correct data types etc.
    (G,L) = I.split(';')
    G = Graph(G, directed=False); L = int(L)

    # split the hint string into a list of vertices, which will
    # form a Hamilton cycle of length at most L, if the hint is correct
    cycle = Path(H.split(','))

    # verify the hint is a Hamilton cycle, and has length at most L
    if G.isHamiltonCycle(cycle) and \
            G.cycleLength(cycle) <= L:
        return 'correct'
    else:
        return 'unsure'
```

Trace through with:
I = "a,b,5 b,c,6 c,a,3;20"
S = "yes"
H = "a,b,c"

Explain what is happening. Why do we need the hint?

7. There are four possible outcomes for verification:




Construct values of $I, S$, and $H$ for TSPD that lead to all four outcomes here. Why does verification return "unsure" instead of "incorrect"?

8. The last definition is the complexity class NPoly:

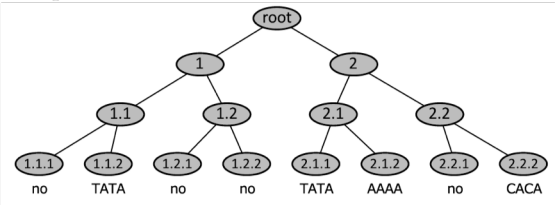
A computation problem is in NPoly that can be solved by a nondeterministic polynomial time program.

Why are the problems Factor and TSPD both in NPoly?

9. It turns out the PolyCheck and NPoly are identical:

We can show that a problem in PolyCheck is in NPoly. Given a polynomial-time verifier V, construct a nondeterministic polynomial time program P as follows. P simulates V, but every time V reads a symbol from the proposed solution or hint, P launches new threads for every possible symbol. If there exists any valid proposed solution and hint, one of P's threads will find it and can return the corresponding solution.

We can show that a problem in NPoly is in PolyCheck. Given a nondeterministic polynomial-time program P, construct a polynomial time verifier V as follows.V simulates one path down P's computation tree. The hint consists of a string describing this path.



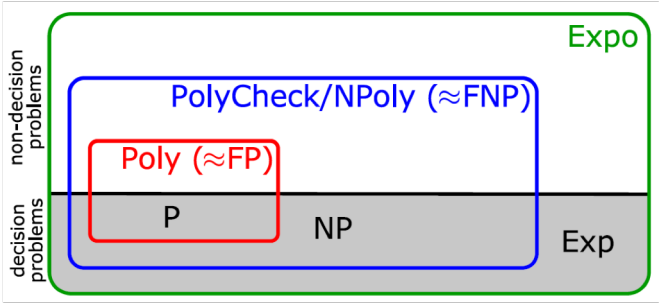

How could we use a hint to describe the path leading to "AAAA" in the tree shown?

10. PolyCheck is sandwiched in between Poly and Expo. There are two steps to prove that.

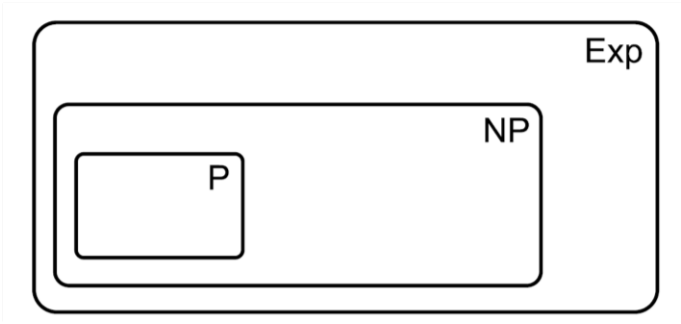
    Proof that Poly $\subseteq$ PolyCheck:
    We have a polynomial-time program P for solving a problem in Poly and we need to construct a polynomial-time verifier.
    How can we use P as a verifier with a proposed solution S? (No hint is required)

    Proof that PolyCheck $\subseteq$ Expo:
    Since PolyCheck = NPoly, we have a nondeterministic polynomial-time program P for solving a problem in PolyCheck and we need to construct a deterministic exponential time program. This can be done with a deterministic simulation of P.
    Why is the running time of this simulation bounded by an exponential function of P's nondeterministic running time?

    We have shown the following relationships of the complexity classes:




    If we look strictly at decision problems we have:



Complete the following assignments for grading. Each should be done individually but you may consult with a classmate to discuss strategies.

**Assignment 1**:
Complete exercise 11.6 a) c) d) and e) on p247 of your text.

**Criteria for Success:** You need a working python program and have determined its running time. Then you need two informal explanations about the complexity class. You may use the chart on p199 to determine the relationships between order of growths.

**Assignment 2**:
Complete 12.2 a) and 12.3 a) on p268 of your text.

**Criteria for Success:** You have descriptions (verbal is sufficient) of algorithms for verifying and nondetermistically solving the problem.

Submit your work in Canvas for grading