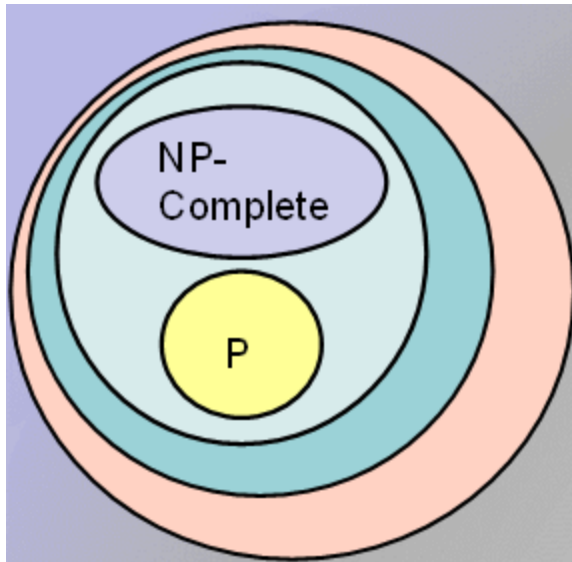


Lecture26



CSE 421 Introduction to Algorithms

Winter 2024

Lecture 26

NP-Completeness and Beyond

1

Announcements

Final Exam: Monday, March 11, 2:30-4:20 PM

- One Hour Fifty Minutes *72.5*
- Comprehensive (but roughly ~~80~~^{72.5}% post midterm)
- Topics will include: dynamic programming, network flow, network flow reductions, NP-completeness, and other stuff

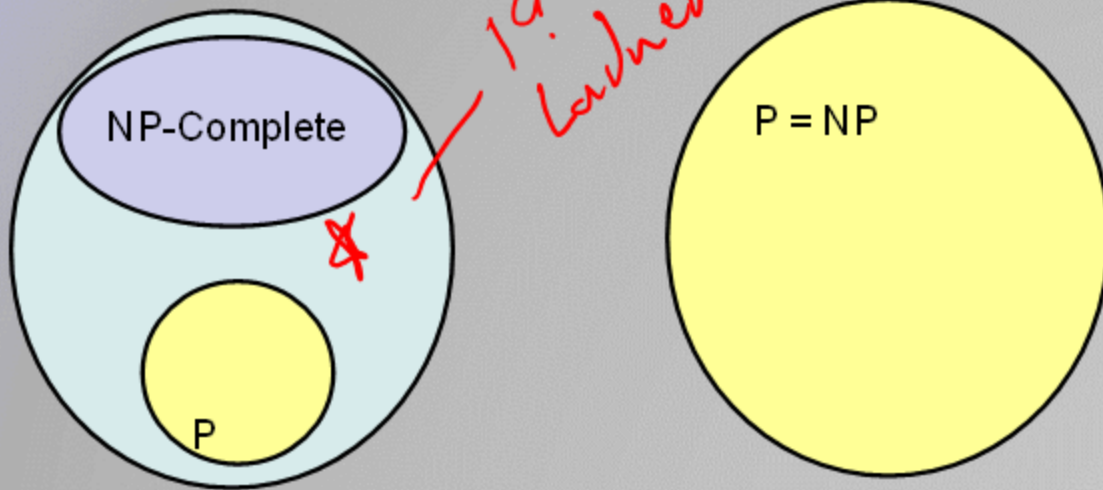
Daylight Saving Time starts 2:00 AM, March 10

NP-Completeness Proofs

- Prove that problem X is NP-Complete
 - Show that X is in NP (usually easy)
 - Pick a known NP complete problem Y
 - Show $Y <_p X$

What we don't know

- P vs. NP

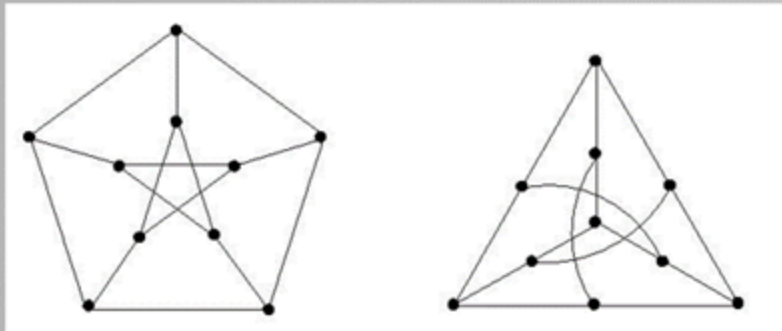


If $P \neq NP$, is there anything in between

- Yes, Ladner [1975]
- Problems not known to be in P or NP Complete
 - Shortest Vector in a Lattice ✓
 - Factorization
 - Discrete Log
 - Graph Isomorphism

Solve $g^k = b$ over a finite group

$O(n^{\log^2 n})$

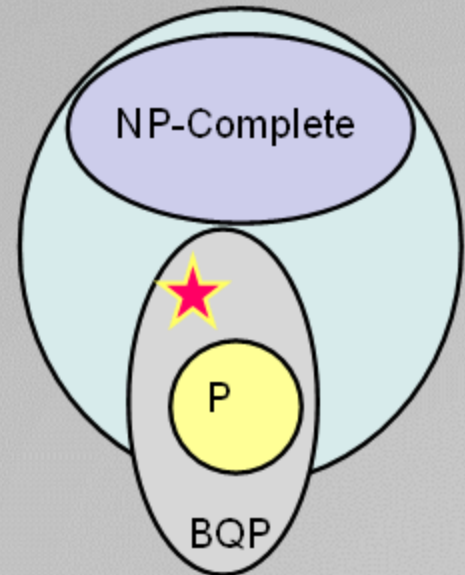


What if?

- 3-SAT can be solved in $O(n^3)$ time
- 3-SAT can be solved in $O(n^{5000})$ time
- Factorization can be solved in $O(n^3)$ time

What about Quantum?

- Computing with Quantum Devices
 - Superposition of states
- Complexity Theory: BQP - Bounded Error Quantum Polynomial Time
- Factorization is in BQP Time (Shor's Algorithm)



Cryptography

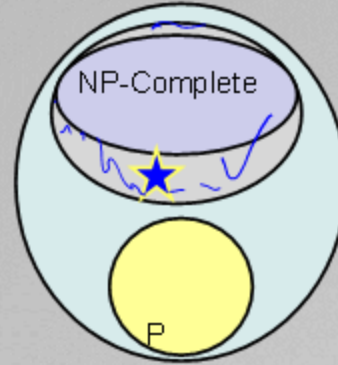
- Standard cryptography depends on number theory problems being hard
 - Keeping factorization secret
- Practical Quantum computing would break RSA
- Post-Quantum Cryptography
 - Find other hard problems to base cryptography on

Shortest Vector in a Lattice

$(1, 4, 3, 4)$

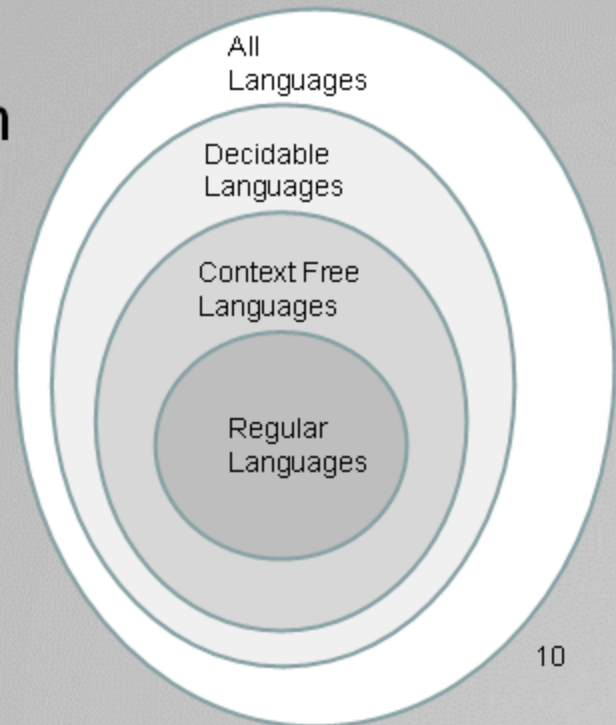
$(2, 7, 2, 3)$

- Given a set of vectors L , what is the shortest non-zero vector that can be formed by integer linear combinations of the vectors?
- The problem is NP-Complete under randomized polynomial time reductions



Complexity Theory

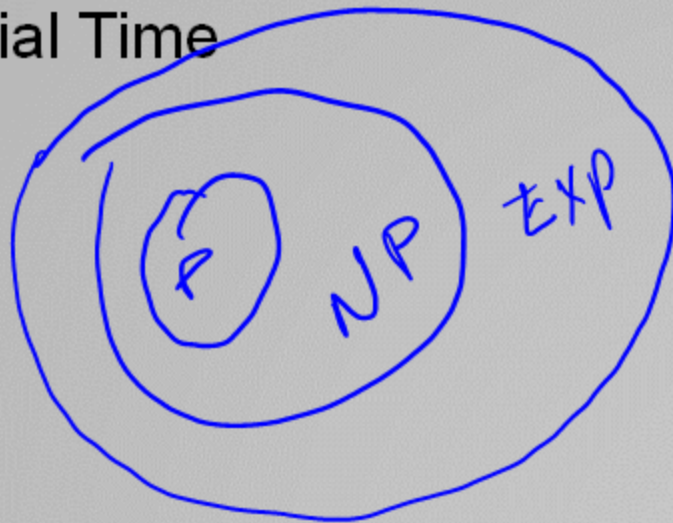
- Computational requirements to recognize languages
- Models of Computation
- Resources
- Hierarchies



10

Time complexity

- P: (Deterministic) Polynomial Time
- NP: Non-deterministic Polynomial Time
- EXP: Exponential Time



Space Complexity

- Amount of Space (Exclusive of Input)
- L: Logspace, problems that can be solved in $O(\log n)$ space for input of size n
 - Related to Parallel Complexity
- PSPACE, problems that can be required in a polynomial amount of space

So what is beyond NP?



NP vs. Co-NP

- Given a Boolean formula, $f(x_1, x_2, \dots, x_n)$, is it true for some choice of inputs

$$\exists x_1, x_2, \dots, x_n f(x_1, x_2, \dots, x_n)$$

- Given a Boolean formula, is it true for all choices of inputs

$$\forall x_1, x_2, \dots, x_n f(x_1, x_2, \dots, x_n)$$



Problems beyond NP

- **Exact TSP**, Given a graph with edge lengths and an integer K , does the minimum tour have length K
- **Minimum circuit**, Given a circuit C , is it true that there is no smaller circuit that computes the same function as C

Polynomial Hierarchy

- **Level 1**

- $\exists X_1 \Phi(X_1), \forall X_1 \Phi(X_1)$

- **Level 2**

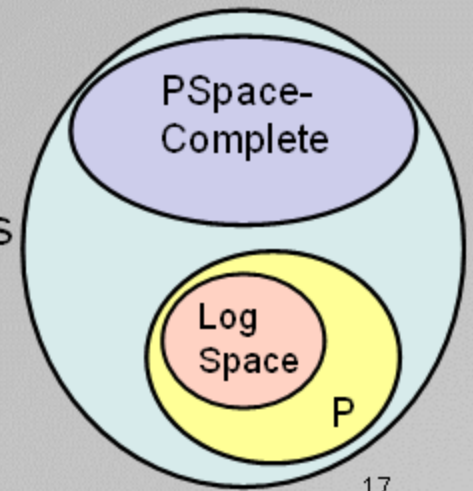
- $\forall X_1 \exists X_2 \Phi(X_1, X_2), \exists X_1 \forall X_2 \Phi(X_1, X_2)$

- **Level 3**

- $\forall X_1 \exists X_2 \forall X_3 \Phi(X_1, X_2, X_3), \exists X_1 \forall X_2 \exists X_3 \Phi(X_1, X_2, X_3)$

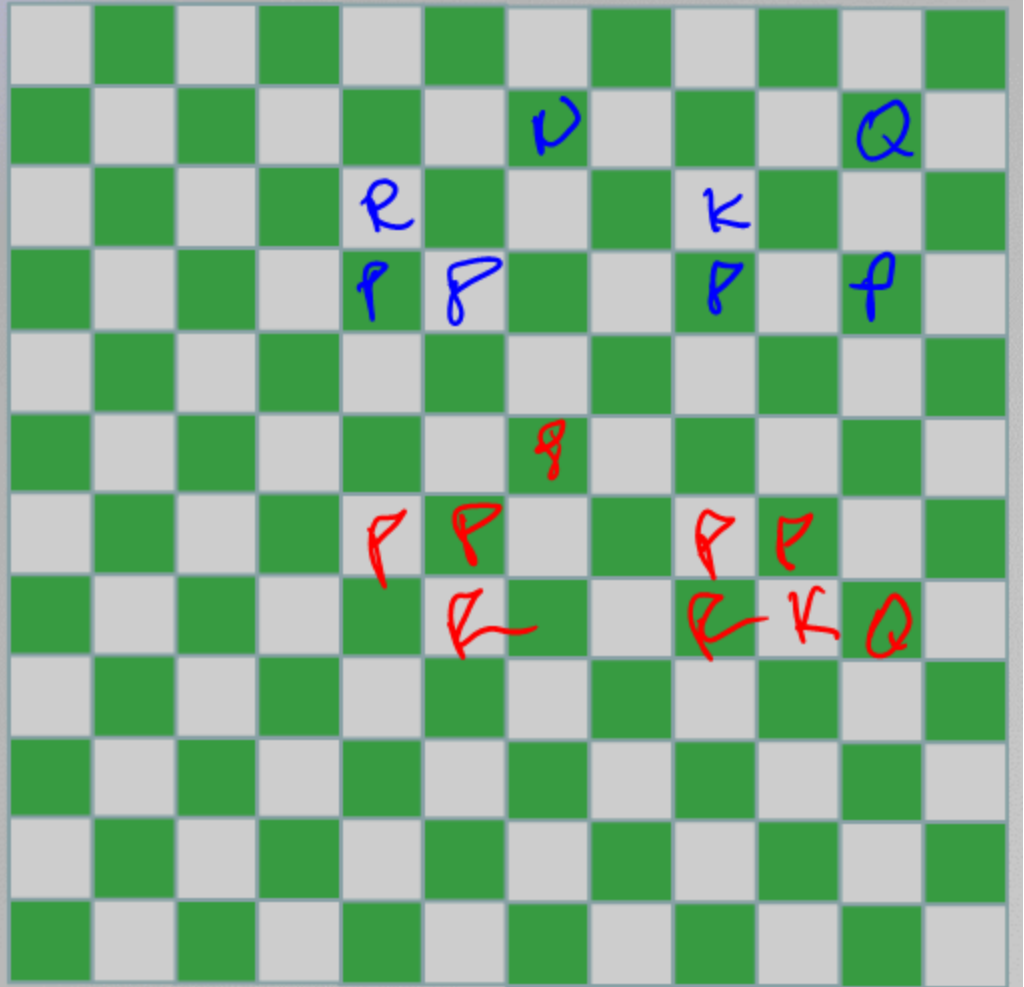
Polynomial Space

- **Quantified Boolean Expressions**
 - $\exists X_1 \forall X_2 \exists X_3 \dots \exists X_{n-1} \forall X_n \Phi(X_1, X_2, X_3 \dots X_{n-1}, X_n)$
- **Space bounded games**
 - Competitive Facility Location Problem
 - N x N Chess
- **Counting problems**
 - The number of Hamiltonian Circuits



17

N X N Chess



Even Harder Problems

```
public int[] RecolorSwap(int[] coloring) {
    int k = maxColor(coloring);

    for (int v = 0; v < nVertices; v++) {
        if (coloring[v] == k) {
            int[] nbdColorCount = ColorCount(v, k, coloring);
            List<Edge> edges1 = vertices[v].Edges;

            foreach (Edge e1 in edges1) {
                int w = e1.Head;
                if (nbdColorCount[coloring[w]] == 1)
                    if (RecolorSwap(v, w, k, coloring))
                        break;
            }
        }
    }
    return coloring;
}
```

Is this code correct?

19

Halting Problem

- Given a program P that does not take any inputs, does P eventually exit?

Impossibility of solving the Halting Problem

Suppose $\text{Halt}(P)$ returns true if P halts, and false otherwise

Consider the program G :

What is $\text{Halt}(G)$?

$\text{If } \text{Halt}(G) = T \Rightarrow$

$\text{If } \text{Halt}(G) = F \Rightarrow$

```

Define G {
  if (Halt(G)){
    while (true) ;
  }
  else {
    exit();
  }
}

```

$\text{Halt}(G) = F$
 $\text{Halt}(G) = T$

Undecidable Problems

- The Halting Problem is undecidable
- Impossible problems are hard . . .