Introduction to quantum computing and simulability

Introduction to computational complexity theory II

<u>Daniel J. Brod</u>

Leandro Aolita

Ernesto Galvão

Instituto de Física
Universidade Federal Fluminense

Outline: Computational complexity theory II

- Review of last lecture;
- Computational complexity conjectures;
- The polynomial hierarchy;
- The magical power of postselection;
  - The postselection argument for demonstrating quantum advantage;
- Counting problems (**#P**)

Church-Turing Thesis

Church-Turing Thesis (physical version)

All computational problems solvable by a realistic physical system can be solved by a Turing machine.

Church-Turing Thesis (**Strong** version)

Any problem that can be solved **efficiently** by a realistic computational device can be solved **efficiently** by a Turing machine.

Complexity classes: **P**

Definition: **P** (complexity class)

(formal) A problem is in **P** if and only if there is a uniform family of efficient classical circuits* such that, for all n-bit inputs x,

- In a YES instance the circuit outputs 1;
- In a NO instance the circuit outputs 0;

* Uniform family of efficient classical circuits:

- depend **only** on size n of input;
- have at most poly(n) gates;
- can be described in poly(n) time

Complexity classes: **NP**

> Definition: **NP** (complexity class)
>
> (informal) Decision problems whose solution can be **checked** efficiently by classical computers.

- Example: Factoring

Hard →

$$67030883744037259 = 179424673 \times 373587883$$

← Easy

Complexity classes: **NP**

Definition: **NP** (complexity class)

(formal) A problem is in **NP** if and only if there is a uniform family of efficient classical circuits that takes as inputs an n-bit string x and a witness y such that

- In the YES instance, there is y of length poly(n) such that the circuit outputs 1;
- In the NO instance, for all y of length poly(n) the circuit outputs 0;

Complexity classes: Reductions

Definition: Reduction

(Informal) Problem **A** reduces to problem **B** if an algorithm for **B** can be used to find a solution fo **A**, and the mapping between them can be done efficiently.

Intuitively, this says **B** is at least **as hard as A**.

Example: **3-SAT** reduces to k-**Clique**.

Complexity classes: Reductions

Definition: **NP-complete**

(Informal) A problem is **NP-hard** if any other **NP** problem reduces to it.

It is also **NP-complete** if it is in **NP** and is **NP-hard**.

Cook-Levin Theorem (1971/1973)

3-SAT is **NP-complete.**

Complexity classes: **NP** - more examples

- **Hamiltonian cycle**: In a graph of n vertices, is there a cycle that visits each vertex exactly once?
- **Subset sum**: Given a collection of n integers, is there a subset of them that sums to exactly x?
- **Graph isomorphism**: Are two n-vertex graphs identical up to relabelling?
- Protein folding, vehicle routing, scheduling.
- Sudoku, tetris and Minesweeper
- A **huge** number of others!
  - Of the **NP** problems listed so far, only **Factoring** and **Graph isomorphism** are <u>not</u> **NP**-complete!

Complexity classes: **BQP**

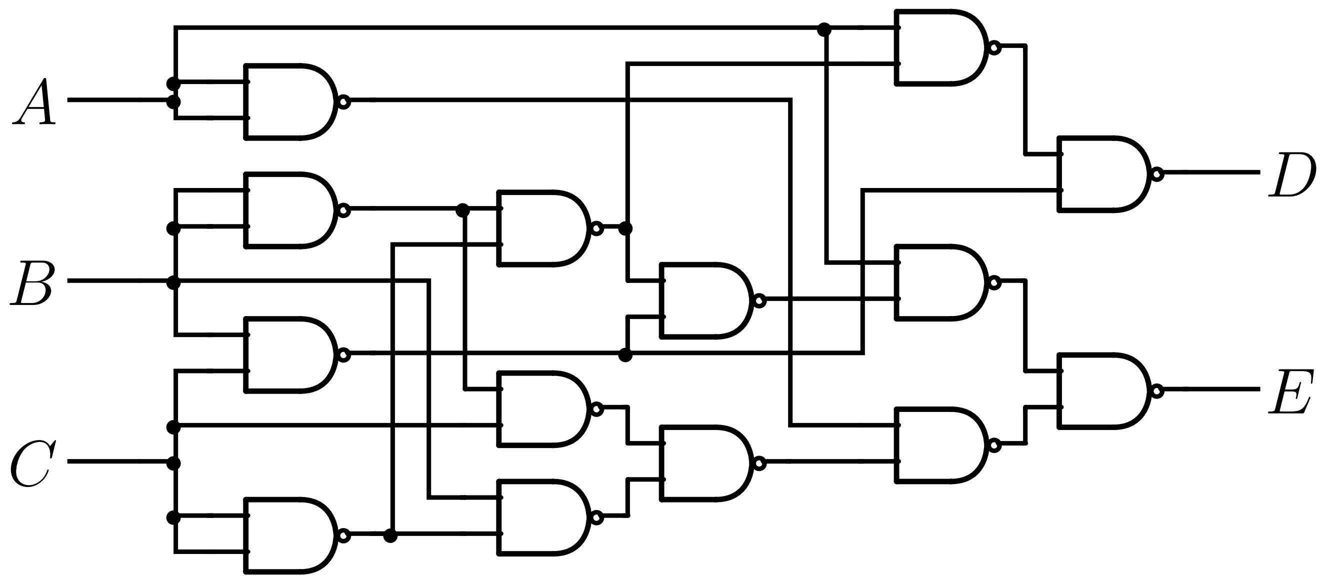
Definition: **BPP** (complexity class)

(formal) A problem is in **BPP** if and only if there is a uniform family of efficient classical circuits such that, for all n-bit input x,

- The circuits have access to a source of random bits;
- In a YES instance the circuit outputs 1 with probability > 2/3;
- In a NO instance, the circuit outputs 0 with probability > 2/3;

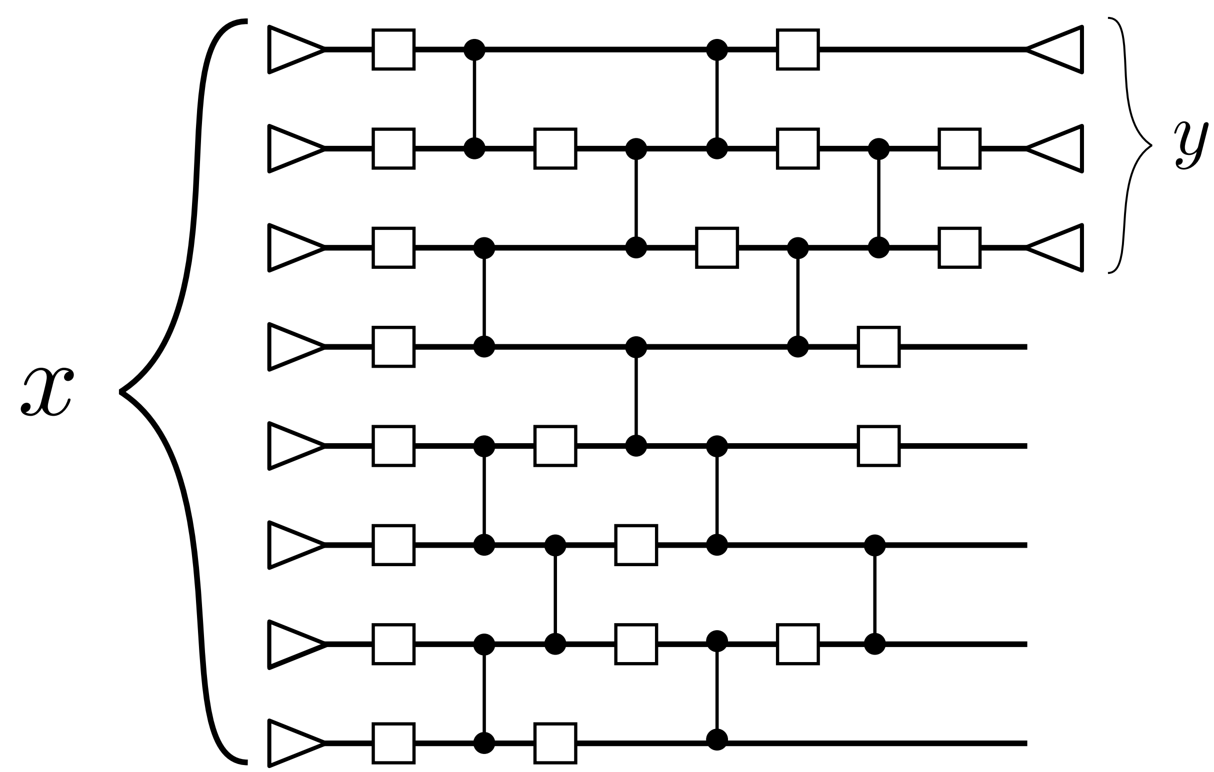
* Computer scientists believe **BPP** = **P**, although there are problems in **BPP** currently not known to be in **P.**

Complexity classes: **BQP**



P
(or **BPP** if we have random bits)

BQP



Complexity classes: **BQP**

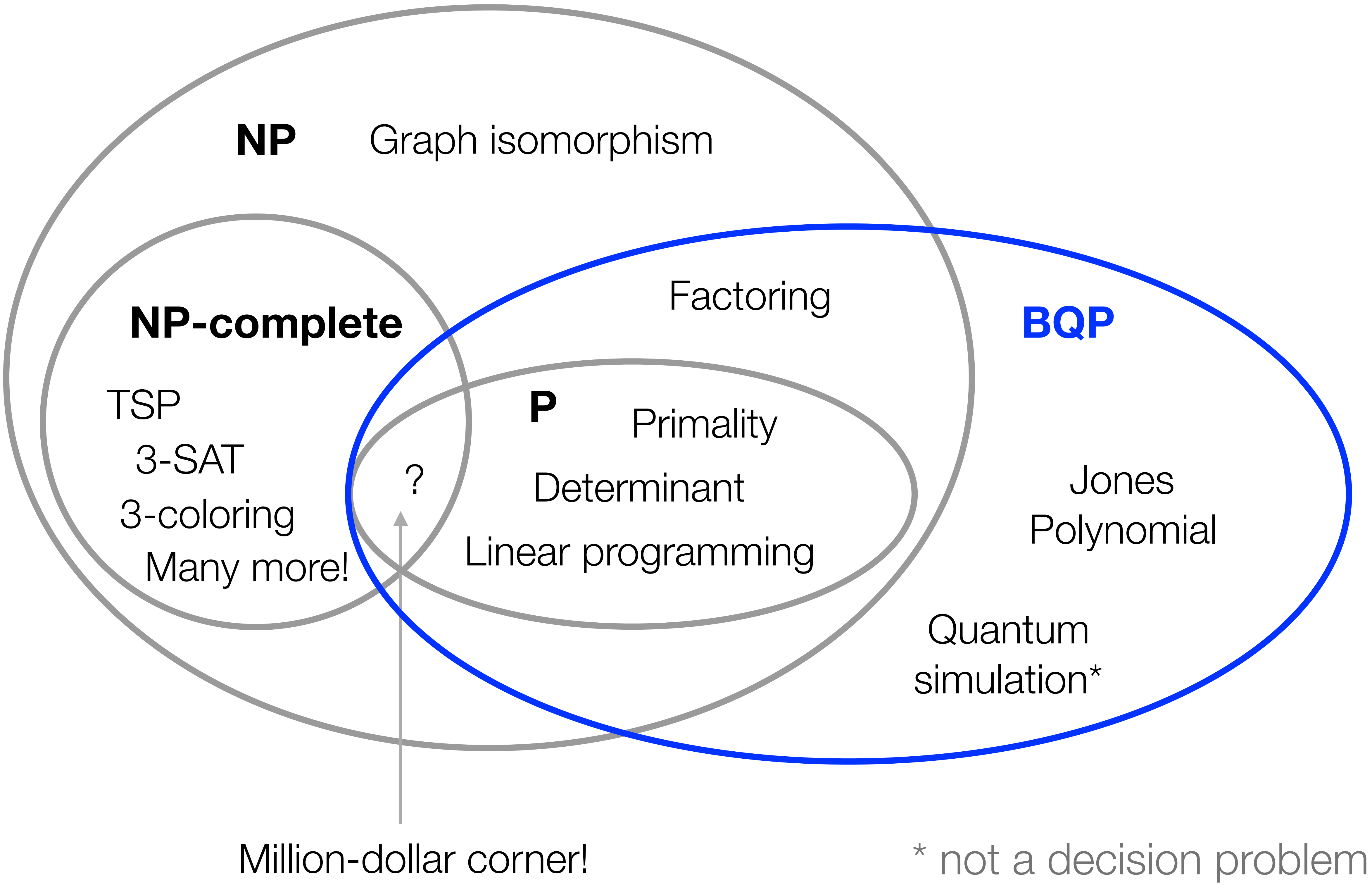
Definition: **BQP** (complexity class)

(formal) A problem is in **BQP** if and only if is exists a uniform family of efficient quantum circuits such that, for all n-qubit input x,

- In a YES instance the output qubit is 1 with probability > 2/3;
- In a NO instance, the output qubit is 0 with probability > 2/3;

* Randomness is built in!

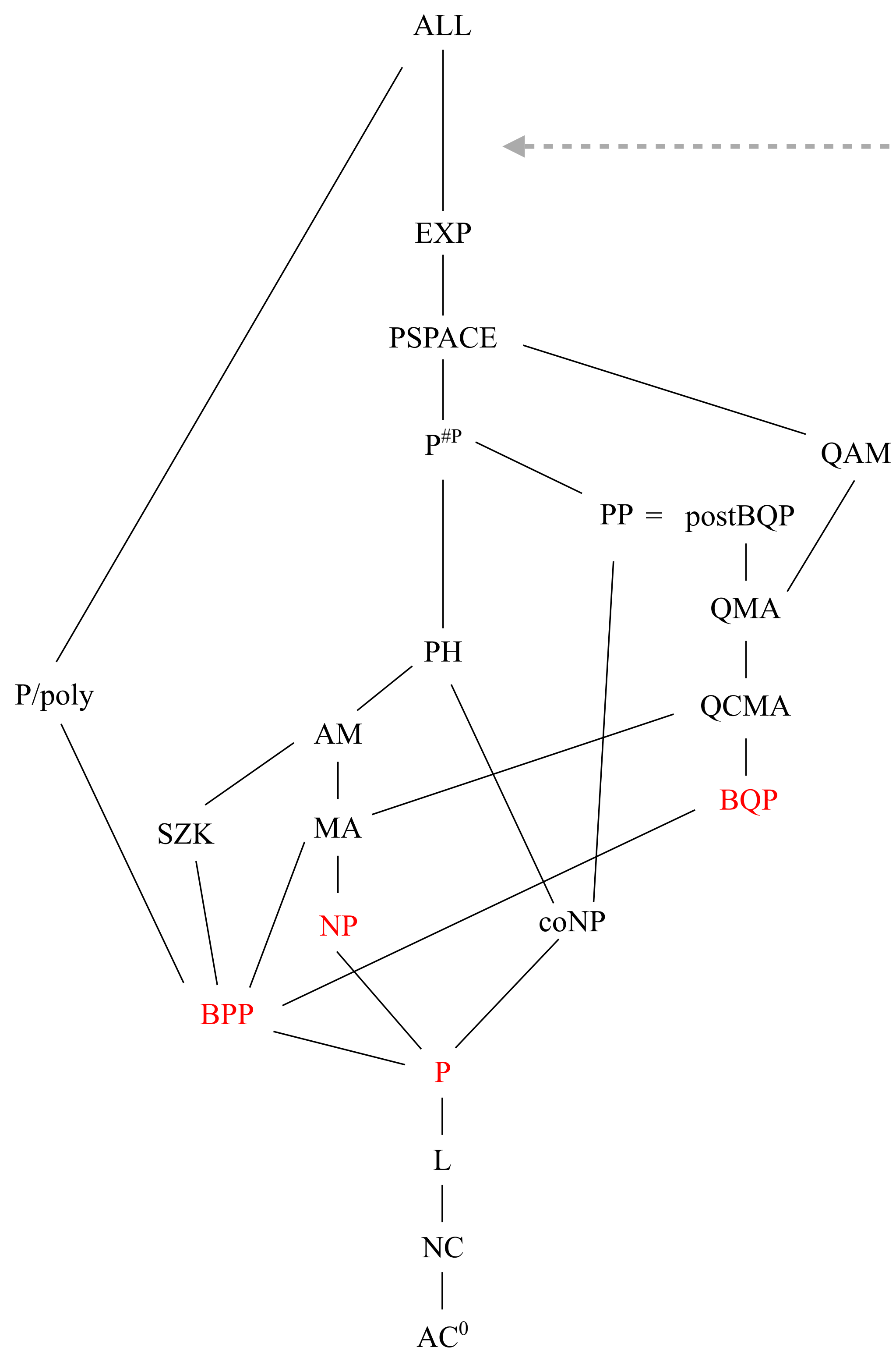
Complexity classes



Outline: Computational complexity theory II

- Review of last class;
- Computational complexity conjectures;
- The polynomial hierarchy;
- The magical power of postselection;
  - The postselection argument for demonstrating quantum advantage;
- Counting problems (**#P**)

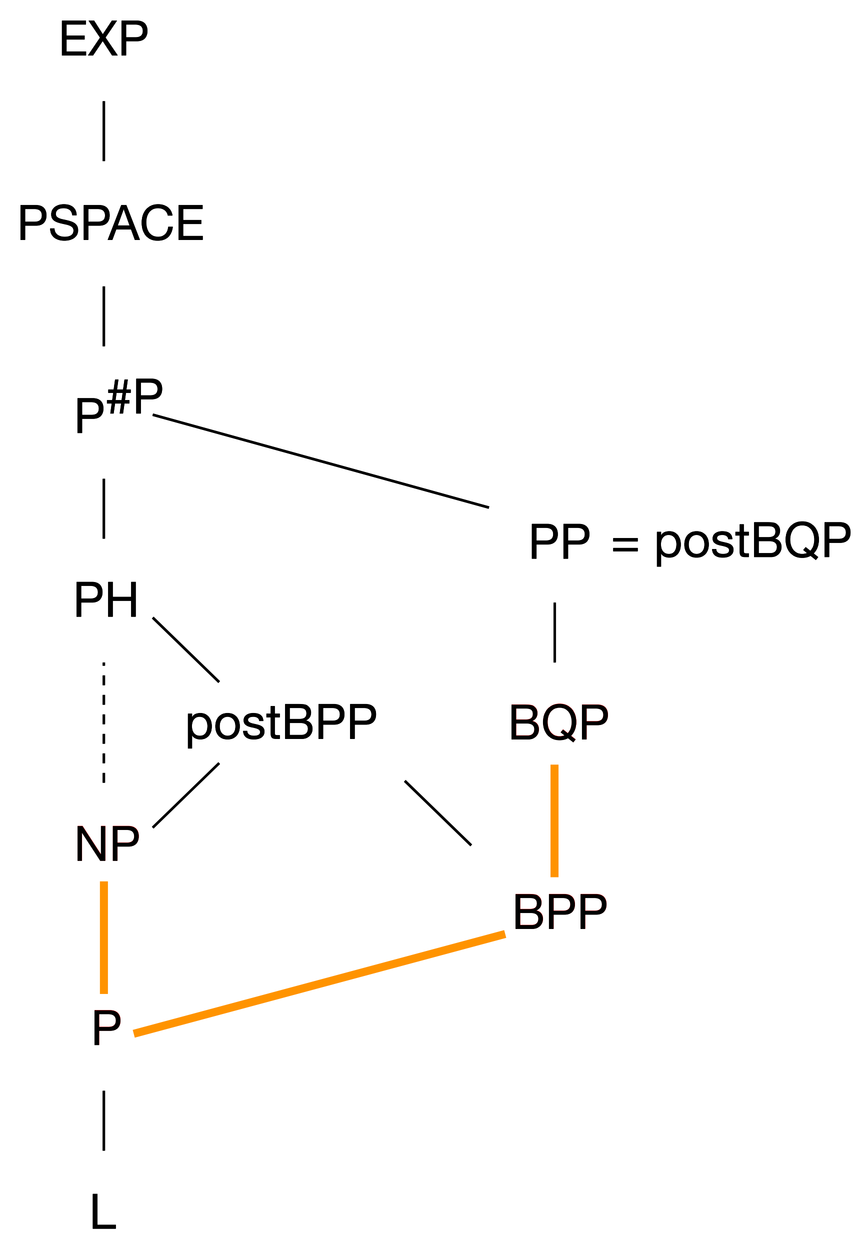
The Complexity (Petting) Zoo



Lines indicate **proven** inclusions (from bottom to top)

Scott Aaronson (Complexity Zoo)

The Complexity (Petting) Zoo



Exercise:
Prove the following inclusions

P ⊆ NP

P ⊆ BPP

BPP ⊆ BQP

While you're at it…

BPP $\overset{?}{\subseteq}$ P

NP $\overset{?}{\subseteq}$ P

Scott Aaronson (Complexity Zoo)

Complexity-theoretic conjectures

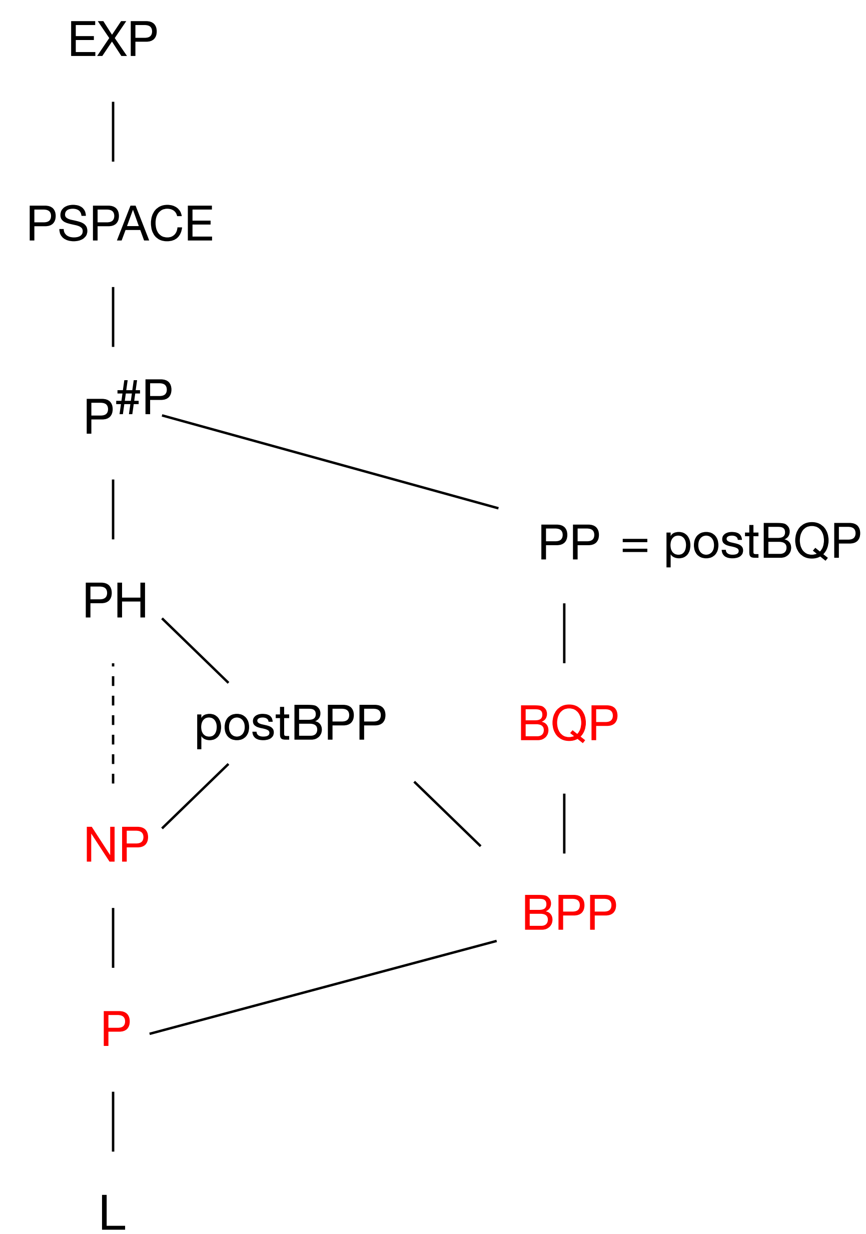
- Proving complexity classes are different is **hard!**
- e.g. we know that

$$\mathbf{L} \subseteq \mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PSPACE} \subseteq \mathbf{EXP}$$

Logarithmic space (L)

Polynomial space (PSPACE)

Exponential time (EXP)



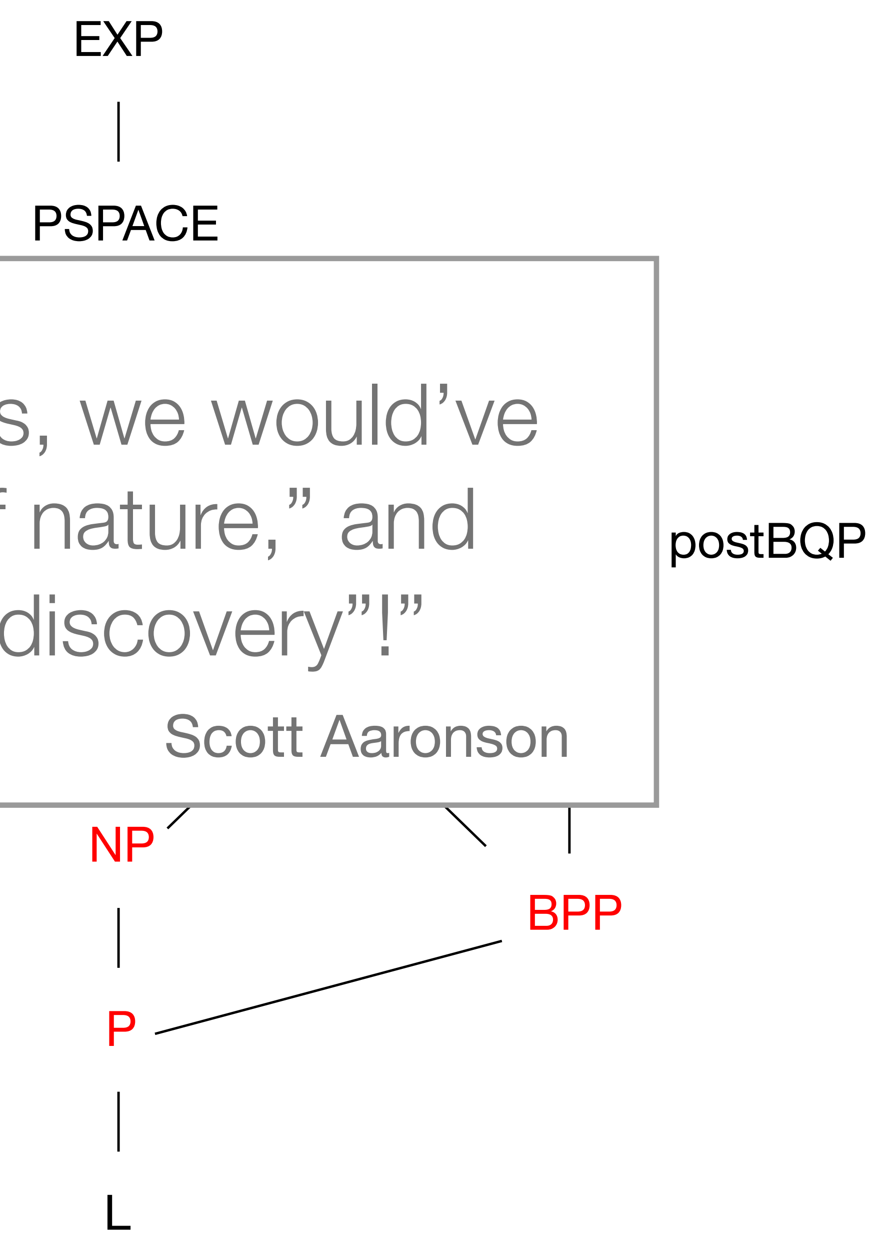
- But we can only **prove** that:

$$\mathbf{PSPACE} \not\subseteq \mathbf{L}$$

$$\mathbf{EXP} \not\subseteq \mathbf{P}$$

Complexity-theoretic conjectures

- Proving complexity classes are different is **hard!**



> "I like to joke that if we were physicists, we would've simply declared **P≠NP** to be a "law of nature," and given ourselves Nobel Prizes for our "discovery"!"
>
> Scott Aaronson

Complexity-theoretic conjectures

- Proving complexity classes are different is **hard!**
- Many arguments have the following structure:

  "If X was true, it would have an unexpected consequence for the structure of complexity classes, therefore X is probably not true"

  e.g. If **3-SAT** has an efficient classical algorithm, then **P** = **NP.**

Outline: Computational complexity theory II

- Review of last class;
- Computational complexity conjectures;
- The polynomial hierarchy;
- The magical power of postselection;
    - The postselection argument for demonstrating quantum advantage;
- Counting problems (**#P**)

Complexity classes: **PH**

Definition: **P** (complexity class)

(alternative informal) Problems of type

"Given input x, is $f(x)=1$?"

Definition: **NP** (complexity class)

(alternative informal) Problems of type

"Given input x, does there exist y such that $f(x,y)=1$?"

* where x and y have length poly(n) and f is efficiently computable;

Complexity classes: **PH**

- Generalization:

> Definition: **PH** (complexity class)
>
> (informal) Problems of type
>
> "Given input x, does there exist y, such that for all z, there exists w such that for all… $f(x,y,z,w,\ldots)=1$?"

- Not an actual complexity class. It is the union of a (presumably) infinite tower of complexity classes!

* $x,y,z,w,\ldots$ have length poly(n) and f is efficiently computable.

Complexity classes: **PH**

"Given input x, does there exist y, such that for all z, there exists w such that for all… $f(x,y,z,w,\dots)=1$?"

n variables → n-1th level of **PH**

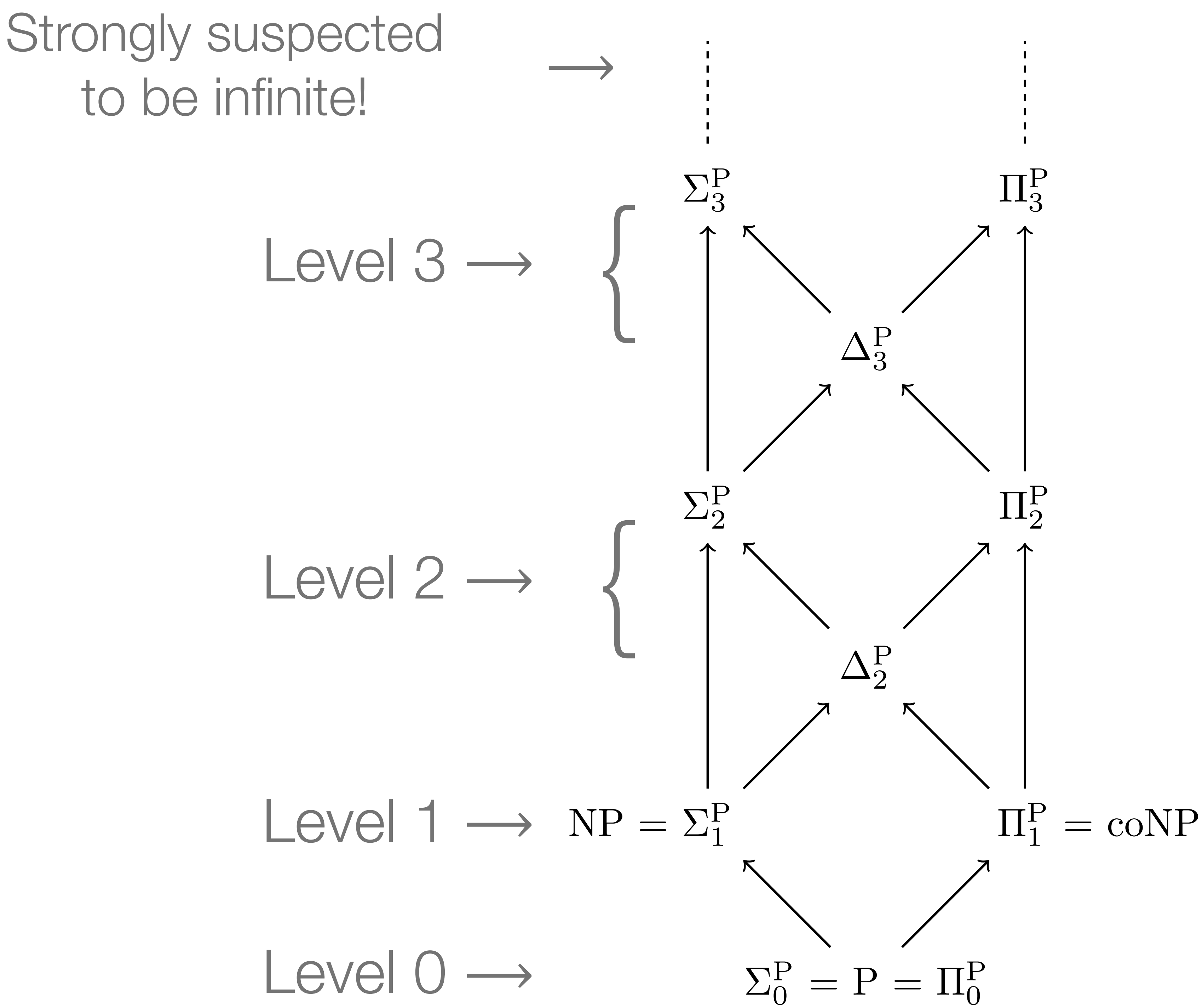
Level 0 ⟶ **P** ("Given input x, is $f(x)=1$?")

Level 1 ⟶ **NP** ("Given input x, is there y s.t. $f(x,y)=1$?") + **co-NP**

Level 2 ⟶ Π_2^P and Σ_2^P

Ex.: Given circuit A that computes a function, is there circuit B of size ≤ k that computes the same function?

Complexity classes: **PH**



Complexity classes: **PH**

- Another variant of a conjecture-based argument:

  "If X was true, **PH** would collapse to its nth level, therefore X is probably not true"

- e.g. "If restricted quantum devices (e.g. **IQP** or linear optics) could be simulated classically, **PH** would collapse to 3rd level!"
  - Ernesto and I will use this a lot in the next lectures!

Outline: Computational complexity theory II

- Review of last class;
- Computational complexity conjectures;
- The polynomial hierarchy;
- The magical power of postselection;
  - The postselection argument for demonstrating quantum advantage;
- Counting problems (**#P**)

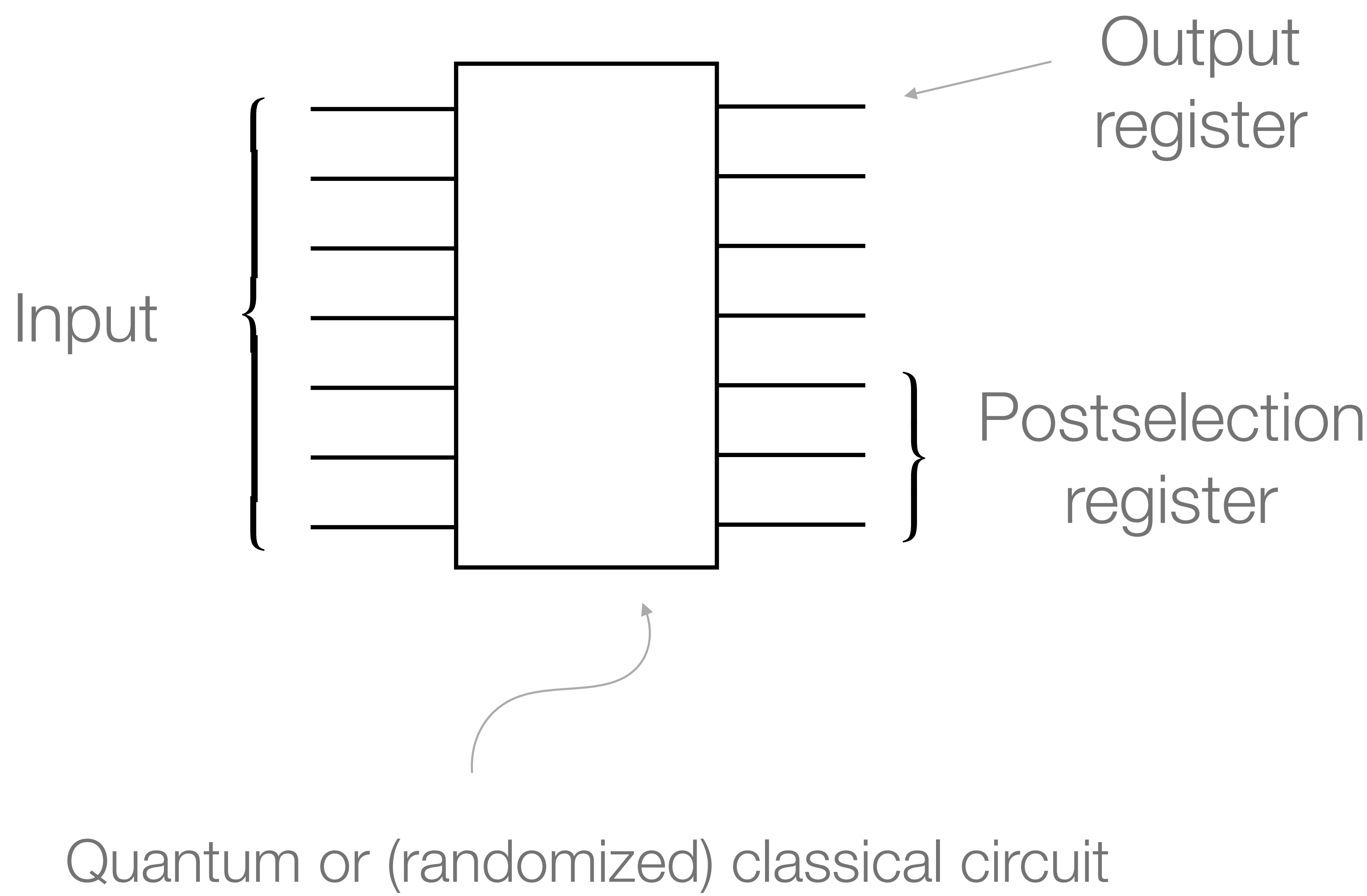
Complexity classes: **postBPP** and **postBQP**

- Let us now give our quantum and classical computers magic powers!

post-selection!

Complexity classes: **postBPP** and **postBQP**

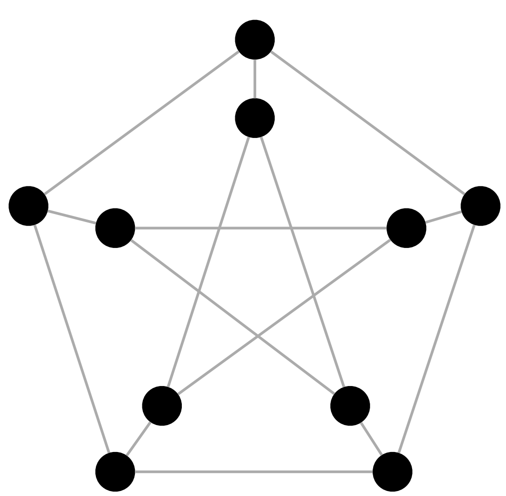
- Postselection: The ability to condition acceptance on some (not-impossible) event, no matter how unlikely.



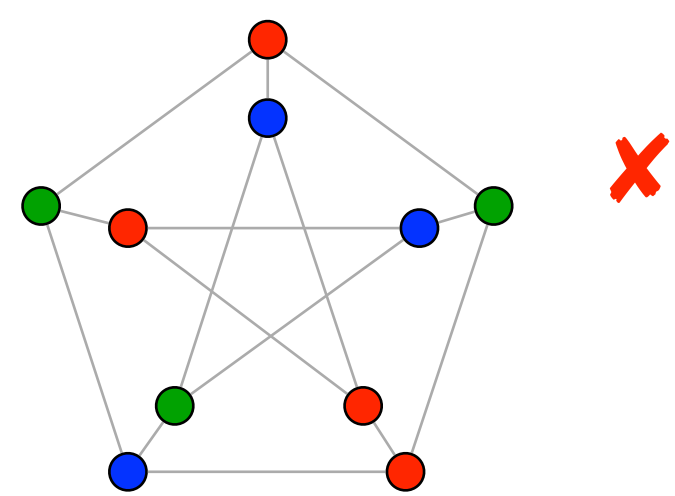
Complexity classes: **postBPP** and **postBQP**

- Why is postselection magic?
  - e.g. it lets classical computers solve **NP** problems efficiently!

Q: Can we color a graph with 3 colors?



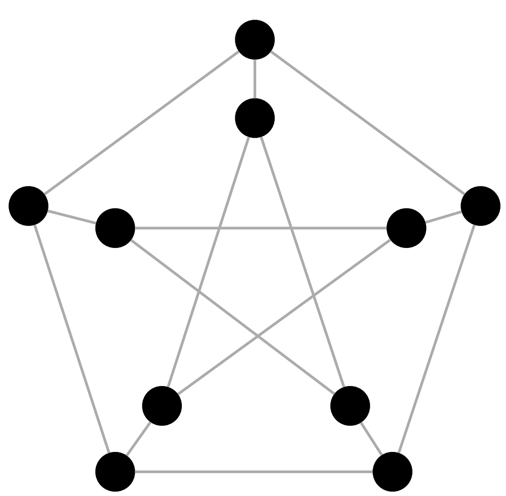
Randomly assign colors. Post-select on a valid coloring!



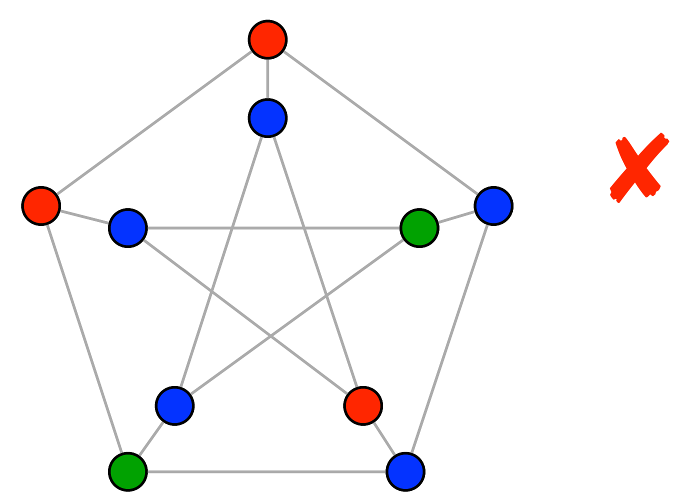
Complexity classes: **postBPP** and **postBQP**

- Why is postselection magic?
    - e.g. it lets classical computers solve **NP** problems efficiently!

Q: Can we color a graph with 3 colors?



Randomly assign colors. Post-select on a valid coloring!

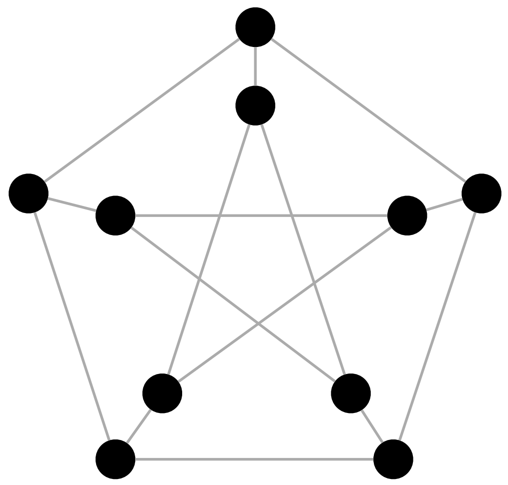


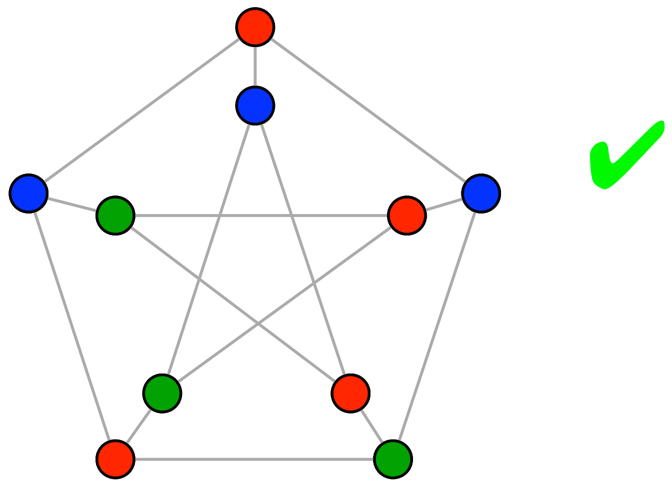
Complexity classes: **postBPP** and **postBQP**

- Why is postselection magic?
  - e.g. it lets classical computers solve **NP** problems efficiently!

Q: Can we color a graph with 3 colors?

Randomly assign colors. Post-select on a valid coloring!





Complexity classes: **postBPP** and **postBQP**

Definition: **postBPP** (complexity class)

A problem is in **postBPP** if and only if there is a uniform family of randomized classical circuits such that, for all n-bit inputs x,

- The postselection register is 1 with probability > 0;

Conditioned on the postselection register outputting 1:

- In a YES instance, the output bit is 1 with probability > 2/3;
- In a NO instance, the output bit is 0 with probability > 2/3;

Complexity classes: **postBPP** and **postBQP**

Definition: **postBQP** (complexity class)

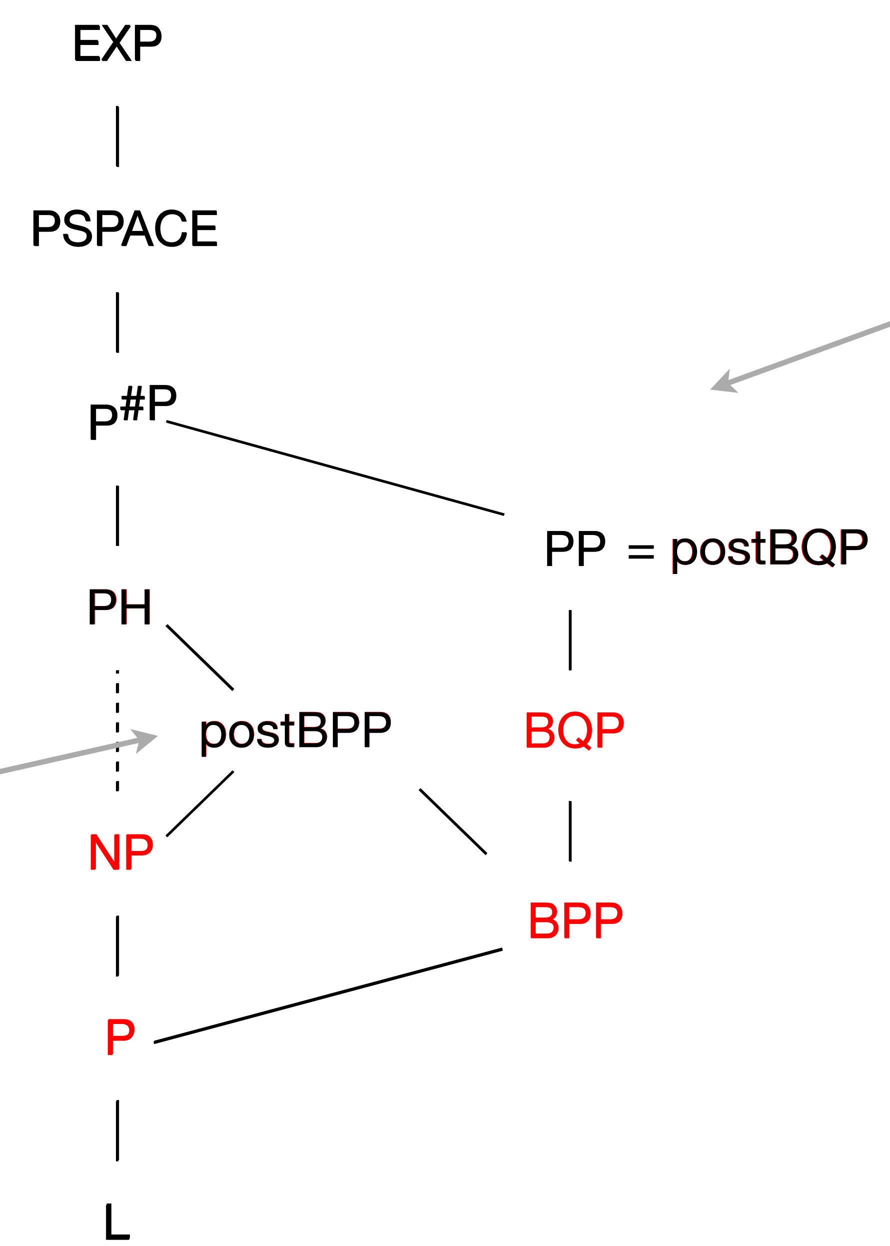
A problem is in **postBQP** if and only if there is a uniform family of quantum circuits such that, for all n-bit input x,

- The postselection qubit outputs 1 with probability > 0;

Conditioned on the postselection register outputting 1:

- In a YES instance, the output qubit is 1 with probability > 2/3;
- In a NO instance, the output qubit is 0 with probability > 2/3;

Complexity classes: **postBPP** and **postBQP**



postBQP
lives outside
PH!*

postBPP
lives inside
PH (in the 3rd level)

* Fine-print: actually, $\mathbf{P}^{\mathbf{postBQP}}$ lives outside **PH**

Recipe for demonstrating quantum advantage

1 - Take a restricted model of quantum computing **A.**
e.g. circuits of commuting gates or linear optics

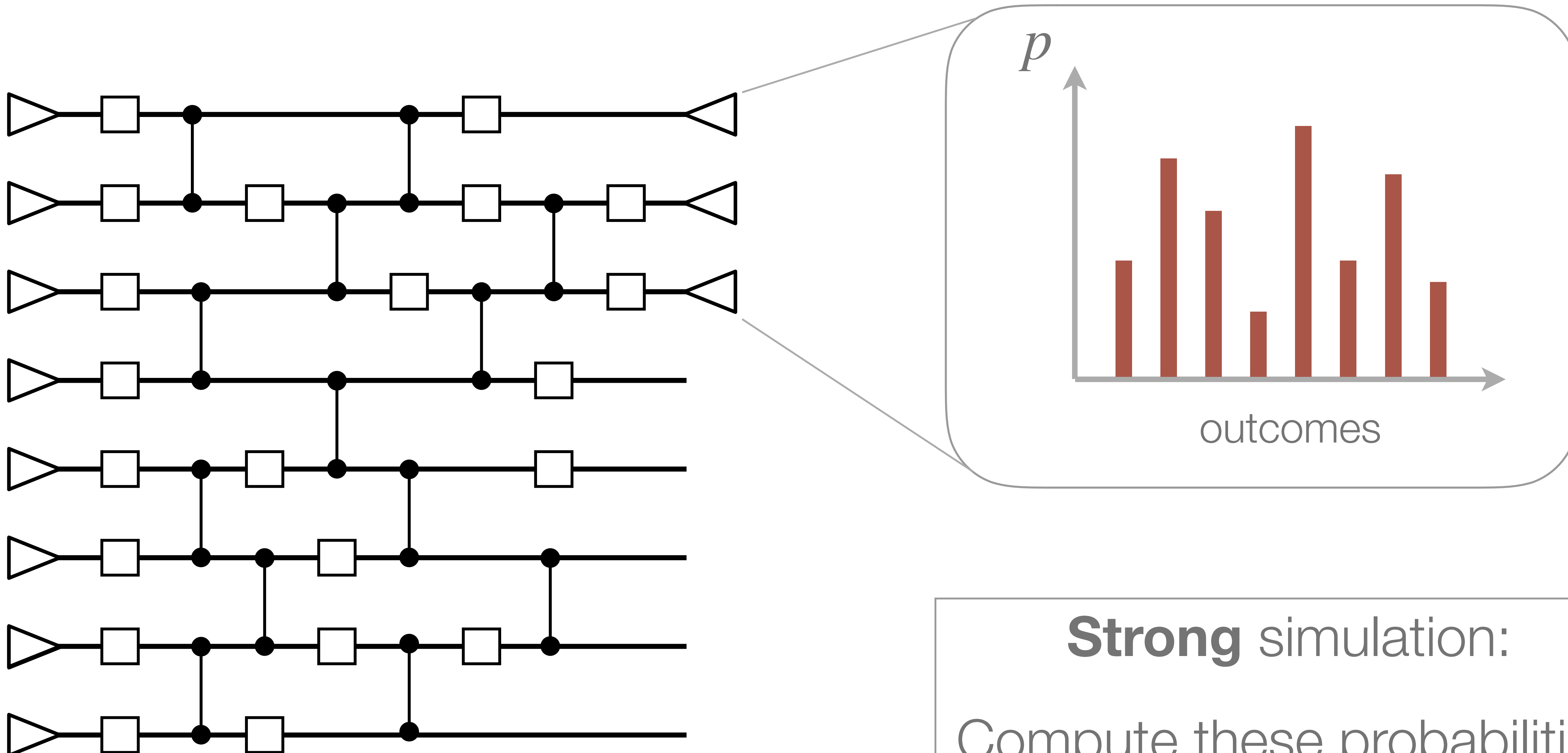
2 - Give it postselection, and see what comes out.
(call it **postA**)

3 - If **A** + post-selection includes quantum computing,
then **postA** = **postBQP**

4 - Suppose there is a classical algorithm to efficiently
simulate **A** (i.e. sample from same distribution).
Then **postA** ⊆ **postBPP**.

5 - But then **postBQP** ⊆ **postBPP** and **PH collapses!**

Interlude: What do we mean by simulation?

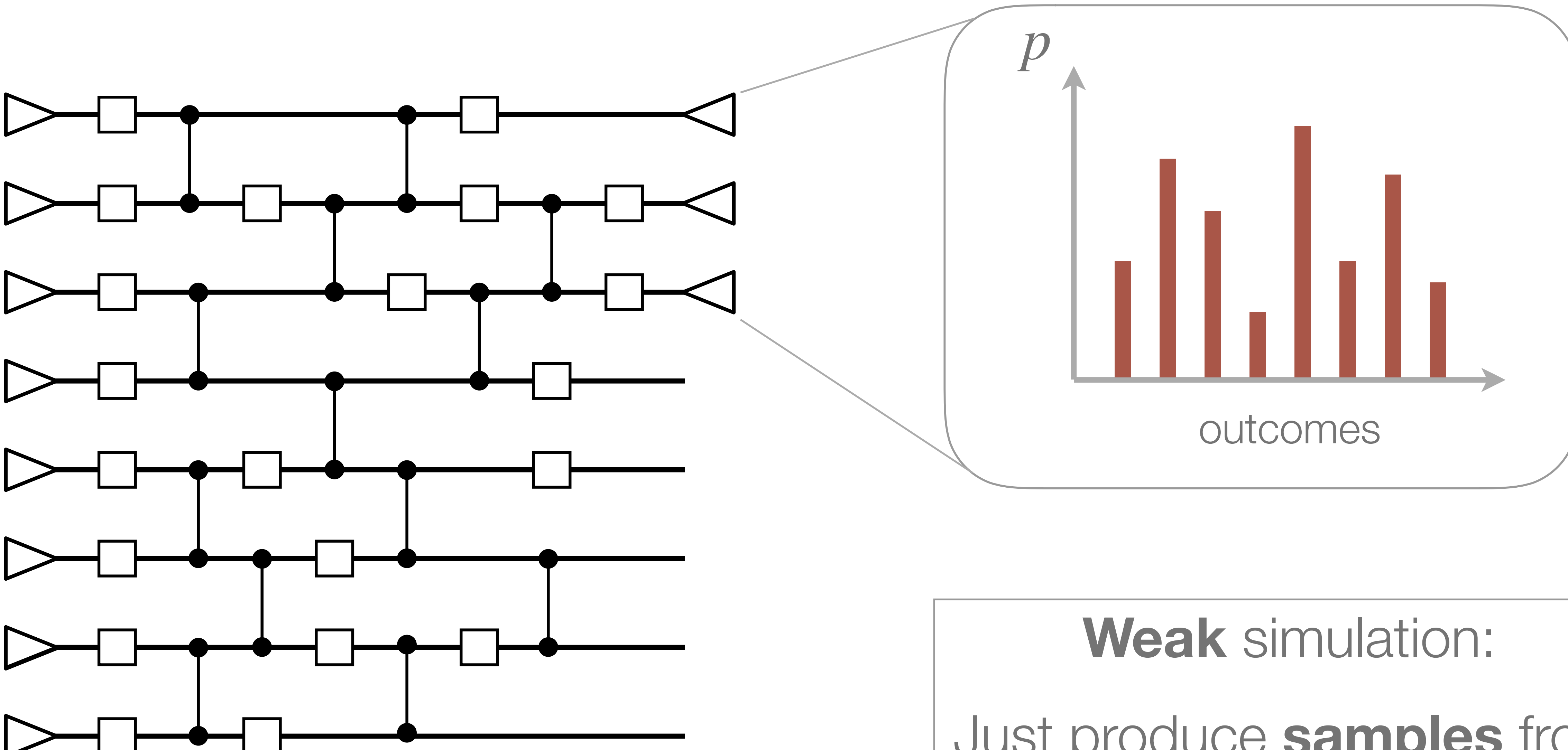


Strong simulation:

Compute these probabilities

Not fair! A quantum computer can't do this either!

Interlude: What do we mean by simulation?



Weak simulation:

Just produce **samples** from this distribution.

This can be refined (exact vs approximate weak simulation)

Recipe for demonstrating quantum advantage

1 - Take a restricted model of quantum computing **A.**
e.g. circuits of commuting gates or linear optics

2 - Give it postselection, and see what comes out.
(call it **postA**)

3 - If **A** + post-selection includes quantum computing,
then **postA** = **postBQP**

4 - Suppose there is a classical algorithm to efficiently
simulate **A** (i.e. sample from same distribution).
Then **postA** ⊆ **postBPP**.

5 - But then **postBQP** ⊆ **postBPP** and **PH collapses!**

Complexity classes: **postBPP** and **postBQP**

- Subtle but important point: This does **not** say anything about **BQP** vs **BPP**! That is:

$$\mathbf{BPP} = \mathbf{BQP} \not\Rightarrow \mathbf{postBPP} = \mathbf{postBQP}$$

- The only conclusion we can draw is about an efficient classical **simulation** of restricted model **A**!

4 - Suppose there is a classical algorithm to efficiently simulate **A** (i.e. sample from same distribution). Then **postA** $\subseteq$ **postBPP**.

Complexity classes: **postBPP** and **postBQP**

- Subtle but important point: This does **not** say anything about **BQP** vs **BPP**! That is:

$$\mathbf{BPP} = \mathbf{BQP} \not\Rightarrow \mathbf{postBPP} = \mathbf{postBQP}$$

- The only conclusion we can draw is about an efficient classical **simulation** of restricted model **A**!

4.1 - Suppose there is a classical algorithm to efficiently **sample** from the output distribution of **A.**

4.2 - Take any problem solvable by some routine in **postA**.

4.3 - To solve the same problem in **postBPP**, just:

4.3.1 - Sample from the output distribution of **A**;

4.3.2 - Apply the same post-selection rule;

Outline: Computational complexity theory II

- Review of last class;
- Computational complexity conjectures;
- The polynomial hierarchy;
- The magical power of postselection;
  - The postselection argument for demonstrating quantum advantage;
- Counting problems (**#P**)

Complexity classes: **#P**

Definition: **#P** (complexity class)

(informal) **#P** is a class of **counting** problems. For example, counting the number of solutions to an **NP** problem.

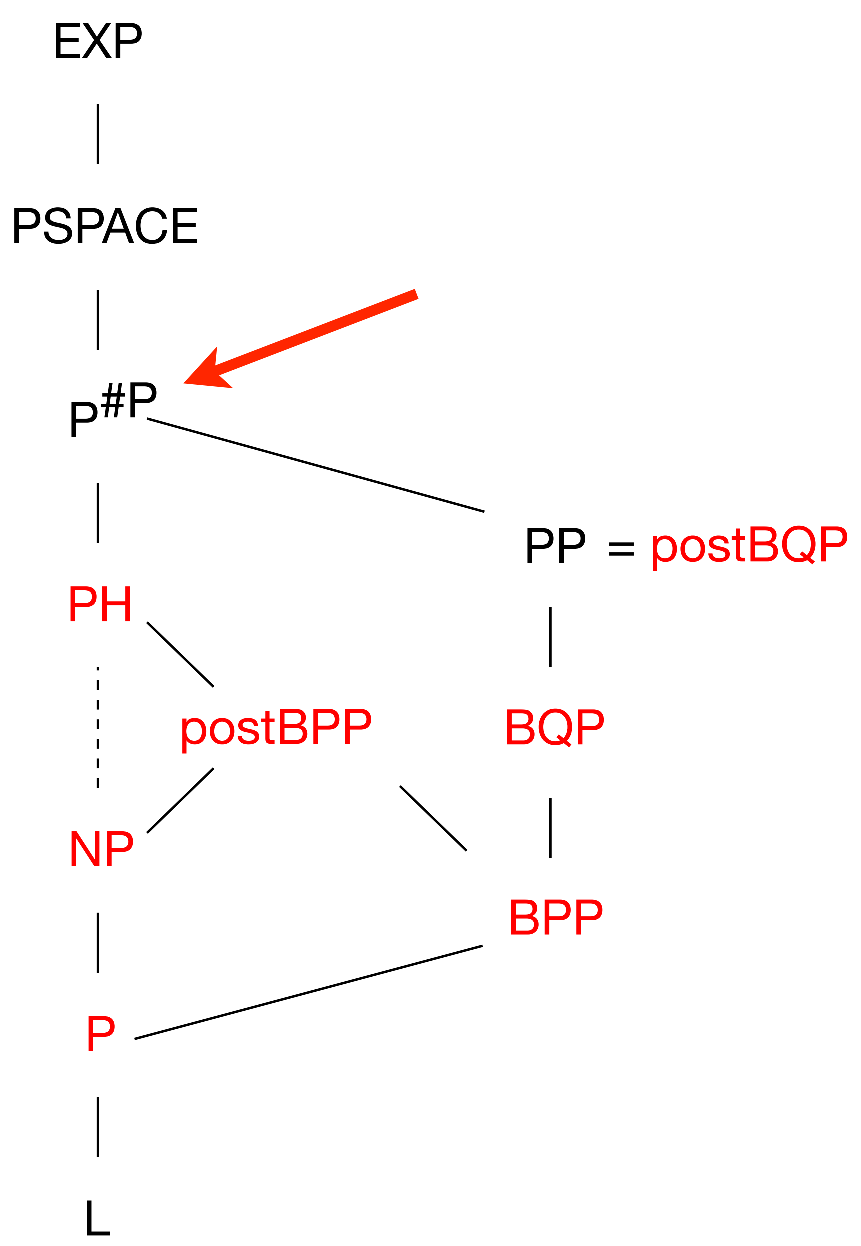
- How hard is counting the number of solutions to an **NP** problem?
  - Very! Finding **one** solution might already be very hard, but there could be exponentially many of them!

Complexity classes: **#P**

Definition: **#P** (complexity class)

(informal) **#P** is a class of **counting** pro
counting the number of solutions to an

- How hard is counting the number of so
problem?
    - Very! Finding **one** solution might already b
    be exponentially many of them!



Complexity classes: **#P** examples

- Counting the number of perfect matchings of a graph.

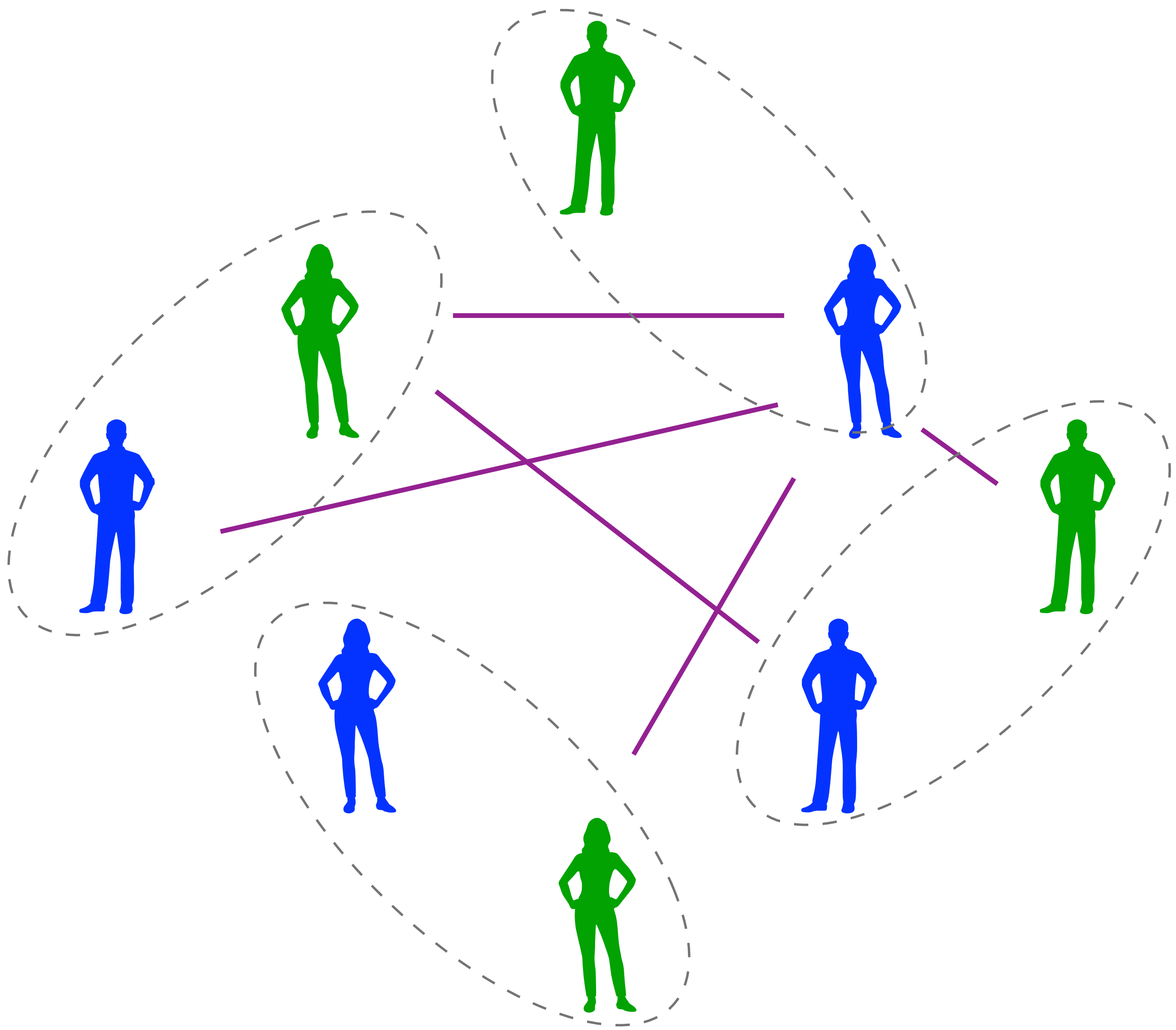
We want to pair n students for an assignment. We want to pair stronger students with weaker ones;

But some of them **hate** each other!

Finding **one** perfect pairing is in **NP**…

(in fact, it is in **P**!)

But counting **all** of them is **#P-hard**!



Complexity classes: **#P** examples

- Computing the Permanent of a matrix:

$$\mathrm{Per}(A) = \sum_{\sigma \in S_m} \prod_{i=1}^{m} a_{i,\sigma(i)}$$

Similar to determinant but without the - signs!

- Permanent is **#P-hard** even if matrix has only 0's and 1's
  - Can be used to encode the number of perfect matchings of a graph!
  - Similar to determinant in form but not complexity! (determinant is in **P**)

Complexity classes: **#P** examples

- Computing the Permanent of a matrix:

Per(A

he - signs!

- ven if matrix has only 0’s and 1’s
  - to encode the number of perfect matchings of a graph!
  - Similar to determinant in form but not complexity! (determinant is in **P**)

Tomorrow

A **shocking** appearance of the permanent in optics!

Also: Find all about what all this has to do with bosons and fermions!