## Randomized algorithms: Complexity classes

**P** - deterministic polynomial algorithm

**NP** - non-deterministic polynomial algorithm (for “yes”)

**co-NP** - non-deterministic polynomial algorithm (for “no”)

**RP** - single-sided Monte Carlo (“nie” is always right)

**co-RP** - single-sided Monte Carlo (“yes” is always right)

**BPP** - two-sided Monte Carlo

**ZPP** - polynomial Las Vegas

**PSPACE** - deterministic algorithm with polynomial space complexity

## What is co-NP?

Recall: if we draw all possible computations of a non-deterministic algorithm as a tree, time is measured as the length of the **shortest path to "yes"**

If answer is "no", **no bound on running time**

For co-NP: **shortest path to "no"**

## RP $\subseteq$ NP

Assume a single-sided Monte Carlo algorithms ("no" is always correct):

- answer "yes" $\Rightarrow$ there exists a sequence of random bits which gives us "yes" (of polynomial length)
- answer "no" $\Rightarrow$ no sequence of random bits gives us "yes"

**What if we change random bit generation to a non-deterministic choice?**

. . . similarly co-RP $\subseteq$ co-NP

## ZPP=RP $\cap$ co-RP

**From Las Vegas to Monte Carlo:** shown in one of the previous lectures

**From Monte Carlo to Las Vegas:**

Assume two algorithms: AlgY: One-sided MC, if it says "yes", it is correct
AlgN: One-sided MC, if it says "no", it is correct

```
repeat:
  if (AlgY == yes) return yes
  if (AlgN == no) return no
```

**What is the expected number of iterations?**

## What is PSPACE?

There is a deterministic algorithm that works in a polynomial memory.