Last topic: P vs NP

Given a computational problem, show that there is (probably) not a polynomial time algorithm for it.

Types of computational problems:

1. Decision problems: output yes/no
   - connectivity: given G, can we get from s to t?

2. Optimization problem: output the best numerical value
   - distance: what is the length of the shortest s-t path?

3. Search problem: identify a particular object.
   - find max element in array

DP? 1, 2, or 3?

2

Min # of minutes needed to get p points on given assignment

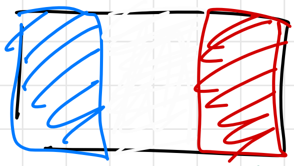
One problem, 3 ways:

→ Decision: Is there a way to get p points in only 3 minutes?
k

Optimization: min # minutes

Search: give the set of problems to answer to get p points in min # minutes

get p points in no more than 3 minutes
k



Reductions

Want to solve decision problem A.

Transform an instance of A into an instance of decision problem B.

Given G w/ edges colored blue, white, or red, is there a walk from s to t that goes blue, white, red, blue, ...

Use known alg. for B ← (BFS) to solve instance.

Given G, is s connected to t?

Return answer.

$O(n^c)$ n ✓ $n \log n$ ✓ $n!$

$\boxed{A \leq_p B}$ "A reduces to B" in polynomial time

Frenchflag walk $\leq$ connectivity

Why did we use $\leq$?

If we give a poly time reduction from A to B,

- if B can be solved in poly time, A can be too
- B is at least as hard as A

no slower to solve

- A is no harder than B
- Reductions give lower bounds on how hard problems are.

A , B
1 , 2

which prob. do we have a lower bound on?

which problem have we given an upper bound on?

A is ~~UB~~ UB by B

~~LB~~

B is LB by A

no poly alg