CS 5112 – 10/31
Intractability and reductions

CORNELL TECH
HOME OF THE JACOBS INSTITUTE

So far…

- We’ve been learning all sorts of techniques for solving algorithms

- Are all problems solvable efficiently?
  - Efficiently means running in time polynomial in input length in some reasonable model of computation (polytime)

Independent set problem

- **Def.** For a graph $G = (V,E)$ an independent set is a set $S \subseteq V$ for which no $u,v \in S$ have an edge

- **Problem:** Given a graph $G = (V,E)$, find an independent set S with maximal size

The brute force solution

- **Problem:** Given a graph G = (V,E), find an independent set S with maximal size

<u>Ind-Set-Max</u><u>(G)</u>
For each S ⊆ V
    For each u,v ∈ S
        If (u,v) ∈ E then
            Add S to candidate list
Return largest candidate S

Runs in time O($n^2 2^n$)

Best known exact algorithm
$1.1996^n\, n^{O(1)}$
https://arxiv.org/abs/1312.6260

Different versions of the problem

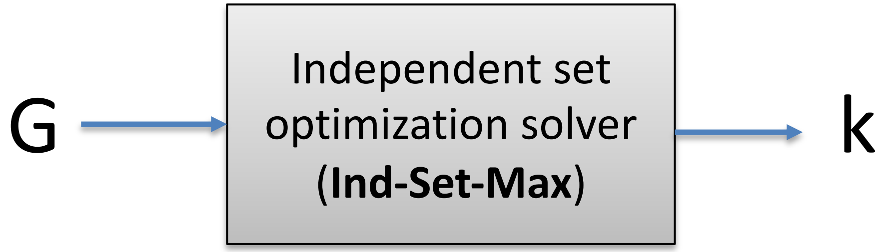
- **Search problem:** Given a graph G = (V,E), find independent set of maximum size

- **Optimization (max) problem:** Given a graph G = (V,E), what is the largest independent set?

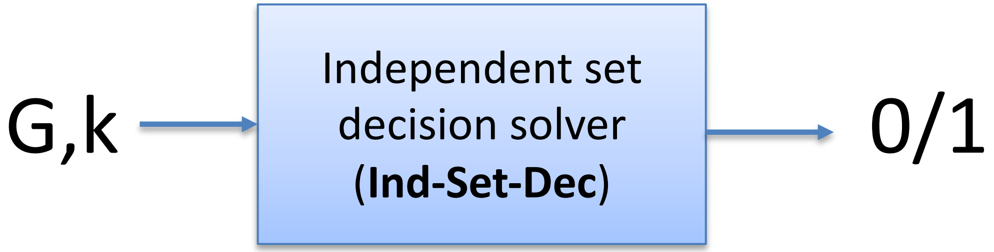
- **Decision problem:** Given a graph G = (V,E) and a number k, does G contain an independent set of size k?

Solving max using decision

Want:



Given:



Solving max using decision

Want:

G → Independent set optimization solver (**Ind-Set-Max**) → k

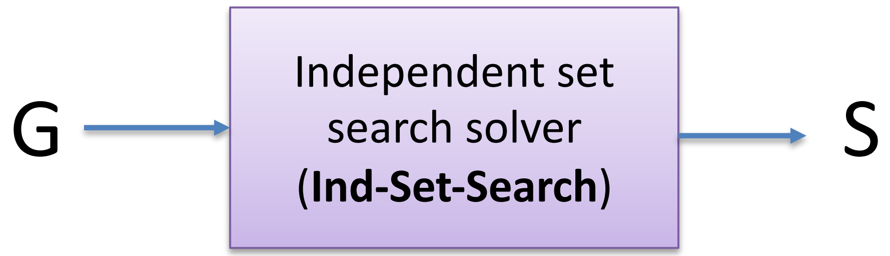
Given:

G,k → Independent set decision solver (**Ind-Set-Dec**) → 0/1

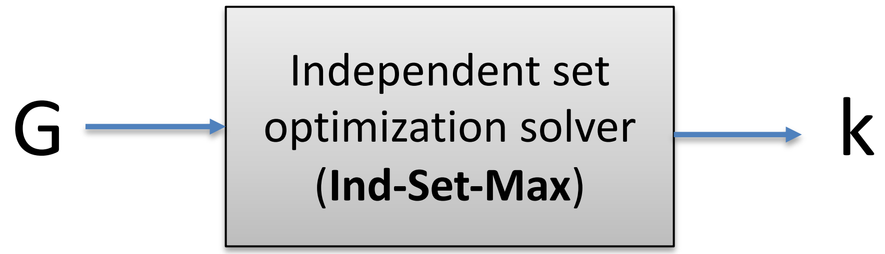
<u>**Ind-Set-Max**(G)</u>
For each k = 1 to |V|:
  If **Ind-Set-Dec**(G,k+1) = 0 then
    Return k

Solving search using max

Want:



Given:



Solving search using max

Want:

G → Independent set search solver (**Ind-Set-Search**) → S

Given:

G → Independent set optimization solver (**Ind-Set-Max**) → k

<u>**Ind-Set-Search**(G)</u>
k = **Ind-Set-Max**(G)
S = {}
For each v ∈ V:
    If **Ind-Set-Max**(G - {v}) = k then
        G = G - {v}
    Else
        G = G - N(v)
        S = S ∪ {v}
Return S

N(v) is the set of all neighbors of v

Different versions of the problem

- **Search problem:** Given a graph G = (V,E), find independent set of maximum size

- **Optimization (max) problem:** Given a graph G = (V,E), what is the largest independent set?

- **Decision problem:** Given a graph G = (V,E) and a number k, does G contain an independent set of size k?

Problems are all equivalent!

Reductions

What we did for independent set problem is use ***reductions***

Def. Problem X reduces to problem Y if arbitrary instances of problem X can be solved using:

1. Polynomial number of standard computational steps, plus
2. Polynomial number of calls to subroutine (oracle) that solves Y

Notation: we write this as X $\leq_P$ Y

Ind-Set-Search $\leq_P$ Ind-Set-Max $\leq_P$ Ind-Set-Dec

Transitivity: X $\leq_P$ Y and Y $\leq_P$ Z implies X $\leq_P$ Z

Ind-Set-Search $\leq_P$ Ind-Set-Dec

Uses for reduction X $\leq_P$ Y

- ***Constructive***: build an algorithm for X using Y
- ***Equivalence***: if can also show that Y $\leq_P$ X then problems are equivalent
- ***Intractability***: if we know Y is not solvable (efficiently), then X isn't either
- ***Evidence***: If lots of people tried to solve Y and failed, then you're unlikely to solve X

Independent set and Vertex cover

- **Def.** For a graph G = (V,E) a *vertex cover* is a set $T \subseteq V$ for which every edge $(u,v) \in E$ has $u \in T$ or $v \in T$ (or both)

Independent set and Vertex cover

- **Def.** For a graph G = (V,E) a *vertex cover* is a set T ⊆ V for which every edge (u,v) ∈ E has u ∈ T or v ∈ T (or both)

- Search problem: Given a graph G = (V,E), find a vertex cover of minimal size
- **Decision problem**: Given a graph G = (V,E) and number k, does G have a vertex cover of size k?

Vertex-Cover-Search $\leq_P$ Vertex-Cover-Dec

Independent set and Vertex cover

Def. For a graph G = (V,E) an *independent set* is a set $S \subseteq V$ for which no $u,v \in S$ have an edge

Def. For a graph G = (V,E) a *vertex cover* is a set $T \subseteq V$ for which every edge $(u,v) \in E$ has $u \in T$ or $v \in T$ (or both)

Independent set and Vertex cover

Lemma. Let G = (V,E) be a graph. Then S is an independent set if and only if its complement V - S is a vertex cover

Vertex-Cover-Dec $\leq_P$ Ind-Set-Dec

Ind-Set-Dec $\leq_P$ Vertex-Cover-Dec

Problems are polytime equivalent!

Does this mean we know how to solve either, efficiently?

Another example: set cover

- Decision problem: Given a set U of n elements, a collection ***S*** = $\{S_1, S_2, \ldots, S_m\}$ of subsets of U, and a number k, does there exist a collection of at most k of these sets whose union is U?

Vertex-Cover-Dec $\leq_P$ Set-Cover-Dec

Vertex-Cover-Dec $\leq_P$ Set-Cov-Dec

Want:

G,k → Vertex cover decision solver (**Ver-Cov-Dec**) → S

Given a graph G = (V,E) and number k, does G have a vertex cover of size k?

Given:

(U,***S***,k) → Set cover decision solver (**Set-Cov-Dec**) → 0/1

Given a set U of n elements, a collection ***S*** = $\{S_1, S_2, \ldots, S_m\}$ of subsets of U, and a number k, does there exist a collection of at most k of these sets whose union is U?

Vertex-Cover-Dec $\leq_P$ Set-Cov-Dec

- Universe U = E
- For each v ∈ V, add subset S_v = { e ∈ E : e incident to v}

- **Lemma.** Can show that if G has vertex cover of size k iff (U,S,k) contains set cover of size k

- What do we know about Ind-Set-Dec versus Set-Cov-Dec?

- Are the all these problems polytime equivalent?

The class P

- We let P be the set of all decision problem problems for which there exists a polytime algorithm solver

- We have seen lots of examples of problems in P

Is Ind-Set-Dec in P?

The class NP

- We let NP be the set of all decision problem problems for which there exists a polytime certifier

- A certifier for decision problem X is an algorithm B that:
  - B runs in polytime in inputs s, t
  - There is a polynomial function p so that for every string s, we have $s \in X$ iff there exists string t such that $|t|$ is polynomial function of $|s|$ and $B(s,t) = 1$

- In words: we can check a solution in polytime, but not necessarily find it in polytime
- **Example:** Vertex cover certificate would be list of vertices in supposed cover. Linear time to check that is a cover

P and NP

- What can we say about P and NP?

Next up: we'll do NP-completeness

- Problems that are in NP that are the "hardest" possible.
  - This means that there is a polytime reduction from any problem in NP to it.
  - In other words: it can be used to solve any problem in NP

- This gives us some understanding of the landscape of complexity of problems, and rise to the foundational question of whether P = NP

Modern cryptography built using reductions

- Reduce breaking X to breaking Y

- If we can’t break Y then we have confidence we can’t break X