CS 5112 – 10/29/2024

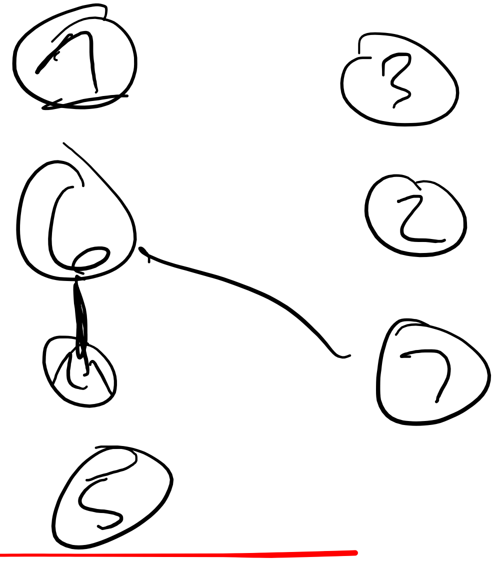
Reductions

Reductions help us related problems

- 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)
  – $P = NP$ problem
- Towards understanding landscape of complexity:
  – How do we reason about relative difficulty of two problems?

Independent set problem

- **Def.** For a graph G = (V,E) an independent set is a set S ⊆ V for which no u,v ∈ 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

The brute force solution

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

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

Fix this

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

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

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

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

Ind-Set-Max-Solver-Guy-Caleb

k = |V|

b = Ind-Set-Dec(G, k)

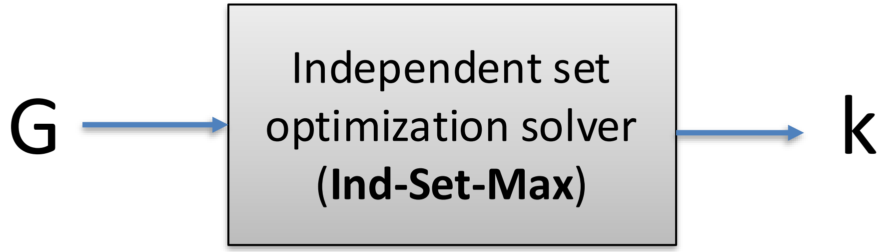
If b = 1 then return k

k = k - 1

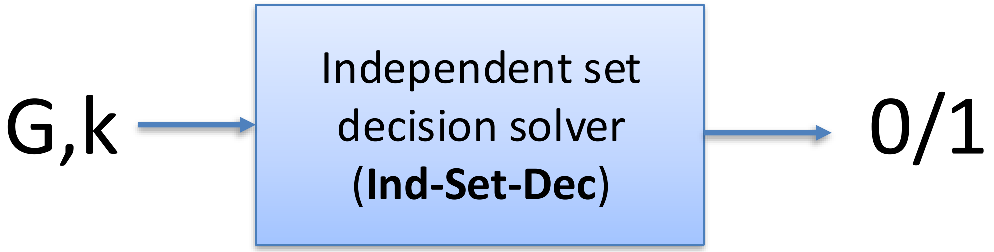
⋮

Solving max using decision

Want:



Given:



Ind-Set-Max(G)

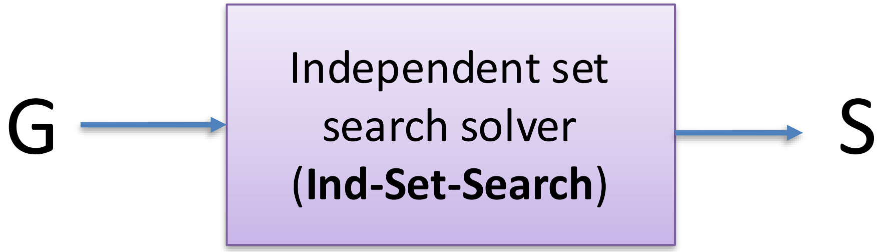
For each k = 1 to |V|:

  If **Ind-Set-Dec**(G,k+1) = 0 then

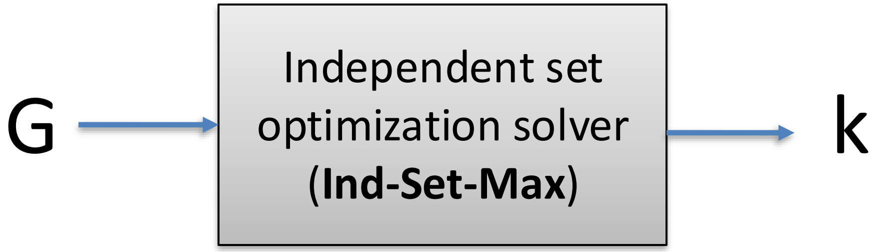
    Return k

Solving search using max

Want:

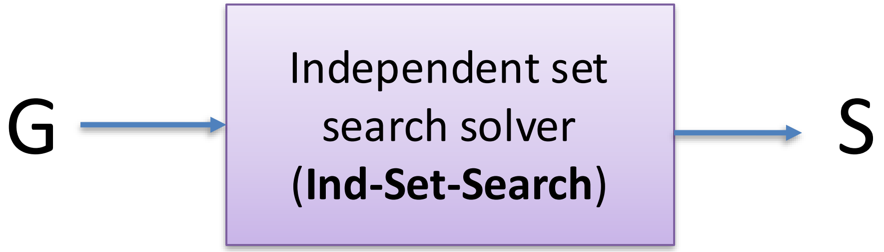


Given:

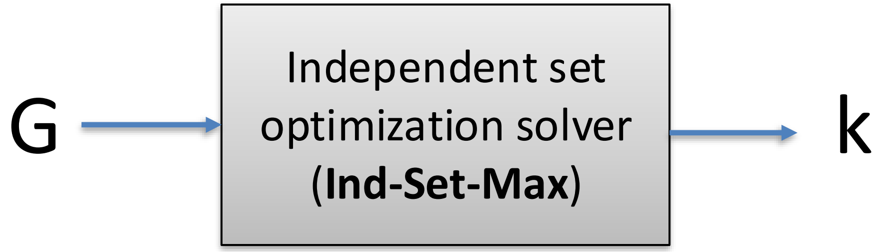


Solving search using max

Want:



Given:



<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 $\boxed{X \le_P Y}$

$$\text{Ind-Set-Search} \le_P \text{Ind-Set-Max} \le_P \text{Ind-Set-Dec}$$

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

$$\text{Ind-Set-Search} \le_P \text{Ind-Set-Dec}$$

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 ⊆ V for which no u,v ∈ S have an edge

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)

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?

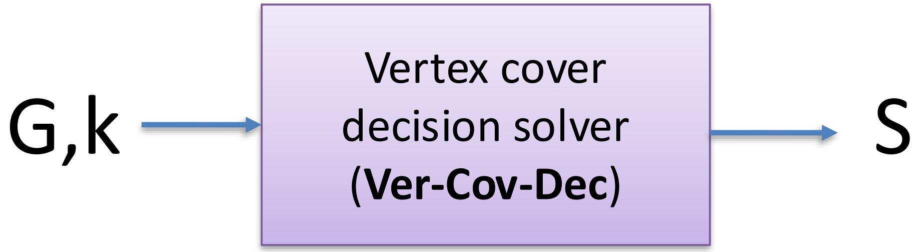
Another example: set cover

- Decision problem: Given a set U of n elements, a collection $\boldsymbol{S} = \{S_1, S_2, \ldots, S_m\}$ 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:



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

Given:

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?

Reductionist approach to cryptography

Formal definitions

Scheme semantics

Security

Security proofs (reductions)

Breaking scheme

Breaking assumptions

Example:

Attacker can not recover credit card

Can not break underlying problem

As long as assumptions holds
we believe in security of scheme!

Provable security yields

1) well-defined assumptions and security goals

2) cryptanalysts can focus on assumptions and models

Base cryptography on conjectured hard problems

A pretty famous problem (1)

- One-wayness of RSA (Rivest-Shamir-Adelman)
  - Given $Y = X^e \bmod N$ for randomly chosen X, find X
  - $N = pq$ is a composite.
  - Factoring N implies ability to invert Y (converse unknown, but conjectured)
- RSA PKCS#1 public-key encryption scheme

Breaking RSA PKCS#1 $\leq_P$ One-wayness of RSA

A pretty famous problem (2)

- Security of block cipher AES (advanced encryption standard)
  - $AES_K(X)$ maps a key K and 128-bit string X to an 128-bit string Y
  - Key recovery security: given X, Y examples, can't recover K efficiently
  - Other types of security: should behave like a truly random permutation
- Encryption protecting online communications use AES

ISP reading encrypted traffic $\leq_P$ Breaking AES

Two important quantum algorithms

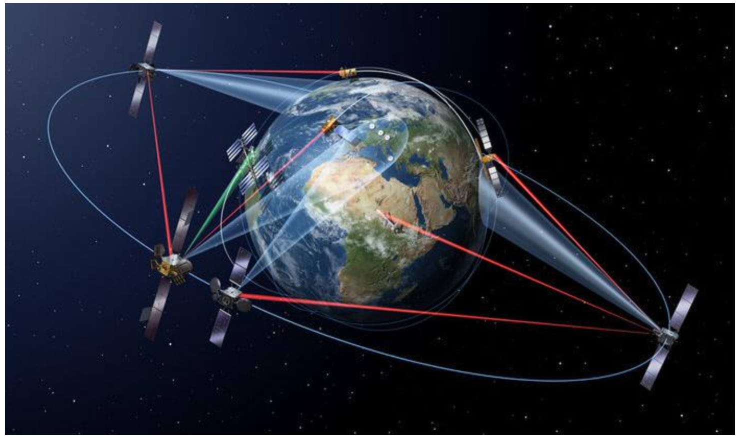
- [Shor 1994] factors composite number N
  - Recall, fastest algorithm we can implement is Number Field Sieve. Runs in time $\mathcal{O}\left(e^{1.9(\log N)^{1/3}(\log\log N)^{2/3}}\right)$ 2^{128} $\frac{1}{2^{128}}$
  - Shor's algorithm gives solution using quantum circuit of size
    $$\mathcal{O}\left((\log N)^2(\log\log N)(\log\log\log N)\right)$$
  - Can be used to compute discrete logs as well

- [Grover 1996] inverts functions with quadratic speedup Shattered
  - Uses $\mathcal{O}\left(\sqrt{2^k}\right)$ to recover K from $AES_K(0^n)$, for |K| = k bits
  - Double key lengths: AES-256 all good

Cryptopoclypse?

- "Post-quantum crypto" (PQC)
  - Asymmetric algorithms with conjectured security against QCs
  - Hash-based signatures, ***lattice-based***, code-based, non-linear systems of equations, elliptic curve isogenies
  - Gaining practical momentum:
    - NIST competition, practical variants of TLS key exchange

- Quantum key distribution
  - Interesting concept, but turned into snake oil
  - Run away!



Cryptopoclypse?

<u>Why are people excited about this now?</u>

1. Quantum computers are getting realer (lots of $$$ thrown at it)
2. Traffic encrypted *now* under possibly future-vulnerable algorithms

Evan Jeffrey's (Google's QC team) at RWC 2017:

Factoring 2048-bit RSA:	250,000,000 physical q-bits with 99.9% accuracy
State-of-the-art QC:	9 physical q-bits with 99.5% accuracy

Classical attacks may invalidate existing algorithms (including PQC!!!)

A conjecture

“Standardized PQC cryptosystems will be broken classically before quantum computers break a non-PQC cryptosystem”

Recent supporting evidence:

- Supersingular Isogeny Diffie–Hellman key exchange [Jao, De fao 2011]
- Supersingular Isogeny Key Encapsulation (SIKE, fourth round candidate for NIST PQC competition) in 2017
- [Castryk, Decru 2022], [Maino, Martindale 2022]:
  - key recovery in one hour on a single core

Reductions powerful tool

- Used to solve problems using subroutines
- Used to relate difficulty of problems
  - Useful in cryptography for “reducing” to a simpler underlying conjectured-hard problem
    - Basis of all modern cryptography
  - Help us define complexity classes

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 ∈ 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?

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