COMP 3011
DESIGN AND ANALYSIS OF ALGORITHMS
FALL 2024

NP-completeness & Approximation Algorithm

LI Bo
Department of Computing
The Hong Kong Polytechnic University

INDIVIDUAL PROJECT

You can choose ⋆***one***⋆ of the three types of projects to work on:

- ***Type I:*** Introduce one (new) algorithm and analyze why the algorithm has good performance.
- ***Type II:*** Summarize the real-world applications of an algorithm.
- ***Type III:*** Implement one algorithm on real-world data sets.

(If you have any other ideas, feel free to discuss them with me.)

INDIVIDUAL PROJECT

Report format (e.g. Introduction + Preliminaries+ Results + Discussion):

- Font: 12-point Times New Roman
- Margin and Spacing: 2.5 cm all round, single column and single-line spacing
- Page limit: no more than 3 pages, including references

Deadline: Dec 20, 2024 (two days after the exam).

ESTABLISHING NP-COMPLETENESS

NP-complete
A problem $Y \in NP$ with the property that for every problem $X \in NP$, $X \leq_P Y$.

Remark.
Once we establish first "natural" NP-complete problem, others fall like dominoes.

Proposition.
If $X \in$ NP-complete, $Y \in$ NP, and $X \leq_P Y$, then $Y \in$ NP-complete.

Proof.
Consider any problem $W \in$ NP. Then, both $W \leq_P X$ and $X \leq_P Y$. By transitivity, $W \leq_P Y$.

Example: SAT $\leq_P$ 3-SAT $\leq_P$ INDEPENDENT-SET $\leq_P$ VERTEX-COVER $\leq_P$ SET-COVER.

Recipe.
To prove that $Y \in$ NP-complete:
- Step 1. Show that $Y \in$ NP.
- Step 2. Choose an NP-complete problem X.
- Step 3. Prove that $X \leq_P Y$.

SUBSET SUM

SUBSET SUM

SUBSET-SUM.

- Given n natural numbers $w_1, \ldots, w_n$ and an integer W, is there a subset that adds up to exactly W? ($M = \sum_1^n w_i > W$)

Exercise.

- $\{215, 215, 275, 275, 355, 355, 420, 420, 580, 580, 655, 655\}, W = 1505$.
- Yes: $215 + 355 + 355 + 580 = 1505$.

SUBSET SUM

Theorem. 3-SAT $\leq_P$ SUBSET-SUM★.

Approach: Given an instance Φ of 3-SAT, we construct an instance of SUBSET-SUM that has solution if and only if Φ is satisfiable.

$$C_1 = \neg x_1 \vee x_2 \vee x_3$$
$$C_2 = x_1 \vee \neg x_2 \vee x_3$$
$$C_3 = \neg x_1 \vee \neg x_2 \vee \neg x_3$$

3-SAT instance

Construction

Given 3-SAT instance Φ with n variables and k clauses, form $2n + 2k$ decimal integers, each having $n + k$ digits:

- Include one digit for each variable x_i and one digit for each clause C_j containing it.
- Include two numbers for each variable x_i.

	x_1	x_2	x_3	C_1	C_2	C_3	
x_1	1	0	0	0	1	0	100,010
$\neg x_1$	1	0	0	1	0	1	100,101
x_2	0	1	0	1	0	0	10,100
$\neg x_2$	0	1	0	0	1	1	10,011
x_3	0	0	1	1	1	0	1,110
$\neg x_3$	0	0	1	0	0	1	1,001

★Not Examined.

Construction

Given 3-SAT instance Φ with n variables and k clauses, form $2n + 2k$ decimal integers, each having $n + k$ digits:

- Include one digit for each variable x_i and one digit for each clause C_j.
- Include two numbers for each variable x_i.
- Include two numbers for each clause C_j.
- Sum of each x_i digit is 1; sum of each C_j digit is 4.

***Key property*:**
No carries possible

C_1	=	$\neg x_1$	∨ x_2	∨ x_3	
C_2	=	x_1	∨ $\neg x_2$	∨ x_3	
C_3	=	$\neg x_1$	∨ $\neg x_2$	∨ $\neg x_3$	

3-SAT instance

	x_1	x_2	x_3	C_1	C_2	C_3	
x_1	1	0	0	0	1	0	100,010
$\neg x_1$	1	0	0	1	0	1	100,101
x_2	0	1	0	1	0	0	10,100
$\neg x_2$	0	1	0	0	1	1	10,011
x_3	0	0	1	1	1	0	1,110
$\neg x_3$	0	0	1	0	0	1	1,001
	0	0	0	1	0	0	100
	0	0	0	2	0	0	200
	0	0	0	0	1	0	10
	0	0	0	0	2	0	20
	0	0	0	0	0	1	1
	0	0	0	0	0	2	2
W	1	1	1	4	4	4	111,444

SUBSET-SUM instance 73

Lemma. If Φ is satisfiable then there exists a subset that sums to W.

Proof.

- Suppose 3-SAT instance Φ has satisfying assignment x^*.
- If $x_i^* = \mathit{true}$, select integer in row x_i ; otherwise, select integer in row $\neg\, x_i$.
- Each x_i digit sums to 1.
- Since Φ is satisfiable, each C_j digit sums to at least 1 from x_i and $\neg\, x_i$ rows.
- Select dummy integers to make C_j digits sum to 4.

$$C_1 = \neg x_1 \vee x_2 \vee x_3$$
$$C_2 = x_1 \vee \neg x_2 \vee x_3$$
$$C_3 = \neg x_1 \vee \neg x_2 \vee \neg x_3$$

3-SAT instance

	x_1	x_2	x_3	C_1	C_2	C_3	
x_1	1	0	0	0	1	0	100,010
$\neg x_1$	1	0	0	1	0	1	100,101
x_2	0	1	0	1	0	0	10,100
$\neg x_2$	0	1	0	0	1	1	10,011
x_3	0	0	1	1	1	0	1,110
$\neg x_3$	0	0	1	0	0	1	1,001
	0	0	0	1	0	0	100
	0	0	0	2	0	0	200
	0	0	0	0	1	0	10
	0	0	0	0	2	0	20
	0	0	0	0	0	1	1
	0	0	0	0	0	2	2
W	1	1	1	4	4	4	111,444

dummy integers

SUBSET-SUM instance

73

Lemma. If there exists a subset that sums to W, then Φ is satisfiable.

Proof.
- Suppose there exists a subset S^* that sums to W.
- Digit x_i forces subset S^* to select either row x_i or row $\neg x_i$ (but not both).
- If row x_i selected, assign x_i = *true* ; otherwise, assign x_i = *false*.
- Digit C_j forces subset S^* to select at least one literal in clause. ■

	x_1	x_2	x_3	C_1	C_2	C_3	
x_1	1	0	0	0	1	0	100,010
$\neg x_1$	1	0	0	1	0	1	100,101
x_2	0	1	0	1	0	0	10,100
$\neg x_2$	0	1	0	0	1	1	10,011
x_3	0	0	1	1	1	0	1,110
$\neg x_3$	0	0	1	0	0	1	1,001
dummy integers	0	0	0	1	0	0	100
	0	0	0	2	0	0	200
	0	0	0	0	1	0	10
	0	0	0	0	2	0	20
	0	0	0	0	0	1	1
	0	0	0	0	0	2	2
W	1	1	1	4	4	4	111,444

SUBSET-SUM instance

73

$$C_1 = \neg x_1 \vee x_2 \vee x_3$$
$$C_2 = x_1 \vee \neg x_2 \vee x_3$$
$$C_3 = \neg x_1 \vee \neg x_2 \vee \neg x_3$$

3-SAT instance

More Stories

SUBSET-SUM.
Given n integers $w_1, \dots, w_n$ and an integer W, is there a subset of them that adds up to exactly W.

$$M = \sum_1^n w_i > W$$

PARTITION.
Given integers $v_1, \cdots, v_n$, is there a subset $S \subseteq \{1, \cdots, n\}$ such that

$$\sum_{i \in S} v_i = \sum_{i \notin S} v_i.$$

Theorem. SUBSET-SUM $\le_P$ PARTITION

"Proof".
Given SUBSET-SUM instance $w_1, \dots, w_n$ and an integer W, set $v_i = w_i$ and W $= \frac{1}{2} \sum_{i \in [n]} v_i$.

?

KNAPSACK.
Given a set of n items X, size $s_i \ge 0$, values $v_i \ge 0$, a weight limit B, and a target value V, is there a subset $S \subseteq X$ such that:

$$\sum_{i \in S} s_i \le B \text{ and } \sum_{i \in S} v_i \ge V.$$

Theorem. SUBSET-SUM $\le_P$ KNAPSACK

Proof.
Given SUBSET-SUM instance $w_1, \dots, w_n$ and an integer W, set $s_i = v_i = w_i$ and $B = V = W$.

PARTITION$\le_P$SUBSET-SUM

PARTITION.

Given integers $v_1, \cdots, v_n$, is there a subset $S \subseteq \{1, \cdots, n\}$ such that

$$\sum_{i \in S} v_i = \sum_{i \notin S} v_i.$$

Theorem. SUBSET-SUM $\leq_P$ PARTITION

Proof.

- Given SUBSET-SUM problem $w_1, \cdots, w_n, W$, we construct the following instance of Partition:
- $v_1 = w_1, \cdots, v_n = w_n$, $v_{n+1} = 2M$, $v_{n+2} = 3M - 2W$, where $M = \sum_1^n v_i > W$.
- We prove $\exists\, S \subseteq \{1, \cdots, n\}$ s.t.

$$\sum_{i \in S} v_i = W$$

if and only if $\exists\, S' \subseteq \{1, \cdots, n, n+1, n+2\}$ $s.t.$

$$\sum_{i \in S'} v_i = \sum_{i \notin S'} v_i.$$

- Suppose there is $S \subseteq \{1, \cdots, n\}$ such that $\sum_{i \in S} v_i = W$.
- Then immediately

$\sum_{i \notin S} v_i = M - W.$

$$\sum_{i \in S} v_i + v_{n+2} = \sum_{i \notin S} v_i + v_{n+1}.$$

$X + 3M - 2W = (M - X) + 2M$

- In any partition of $\{1, \cdots, n, n+1, n+2\}$, v_{n+1} and v_{n+2} must be separated, because they add up to $5M - 2W > \sum_i^n v_i$.
- Thus, for $S = S' \backslash \{n+1, n+2\}$,

$$\sum_{i \in S} v_i + v_{n+2} = \sum_{i \notin S} v_i + v_{n+1}$$

- It follows directly from the arithmetic that $\sum_{i \in S} v_i = W$.

Summary

SUBSET-SUM.
Given n integers $w_1, \ldots, w_n$ and an integer W, is there a subset of them that adds up to exactly W.

$$M = \sum_1^n w_i > W$$

PARTITION.
Given integers $v_1, \cdots, v_n$, is there a subset $S \subseteq \{1, \cdots, n\}$ such that

$$\sum_{i \in S} v_i = \sum_{i \notin S} v_i .$$

KNAPSACK.
Given a set of n items X, size $s_i \geq 0$, values $v_i \geq 0$, a weight limit B, and a target value V, is there a subset $S \subseteq X$ such that:

$$\sum_{i \in S} s_i \leq B \text{ and } \sum_{i \in S} v_i \geq V .$$

Theorem. SUBSET-SUM $\leq_P$ KNAPSACK

Theorem. PARTITION $\leq_P$ SUBSET-SUM

Theorem. SUBSET-SUM $\leq_P$ PARTITION

3SAT $\leq_P$ SUBSET-SUM $\equiv_P$ PARTITION $\leq_P$ KNAPSACK

DO NOT forget to show that the problem is NP, in order to claim they are NP-complete!

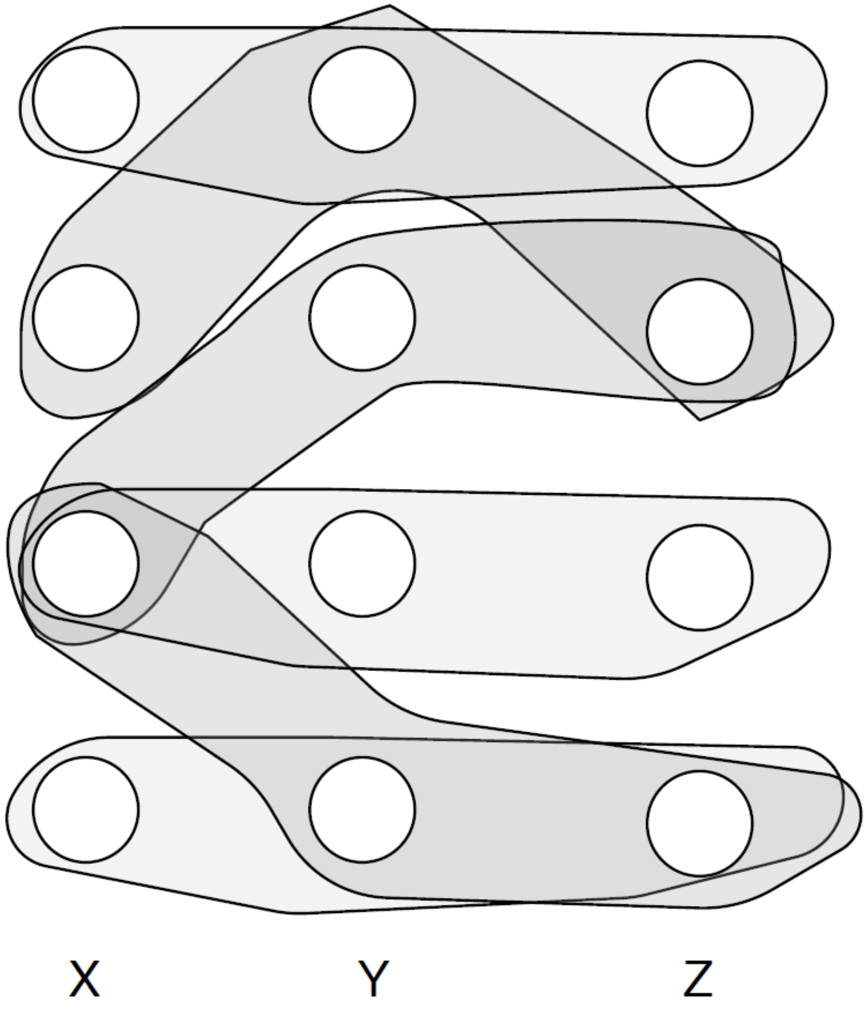
SIX BASIC NP-COMPLETE PROBLEMS

Six Basic NP-Complete problems

- 3-Satisfiability (3-SAT)
- ***3-Dimensional Matching (3DM)*** — 3SAT$\leq_p$3DM

  Exact Cover by 3-Sets (X3C) — 3DM$\leq_p$X3C
- Vertex Cover (VC)
- Independent Set (IS)
- ***Hamiltonian Cycle (HC)*** — 3SAT$\leq_p$HC, VC$\leq_p$HC
- Partition

3-Dimensional Matching (3DM)



Given: Sets X, Y, Z, each of size n, and a set $T \subset X \times Y \times Z$ of order triplets.

Question: is there a set of n triplets in T such that each element is contained in exactly one triplet?

Exact Cover by 3-Sets (X3C)

Given: a set U with $|U| = 3n$ and a collection C of 3-element subsets of U.

Question: Does C contain an exact cover for U, that is, a subcollection $C' \subseteq C$ such that every element of U occurs in exactly one member of C'?

Theorem. 3DM$\leq_p$X3C.

3DM: $T \subseteq X \times Y \times Z$

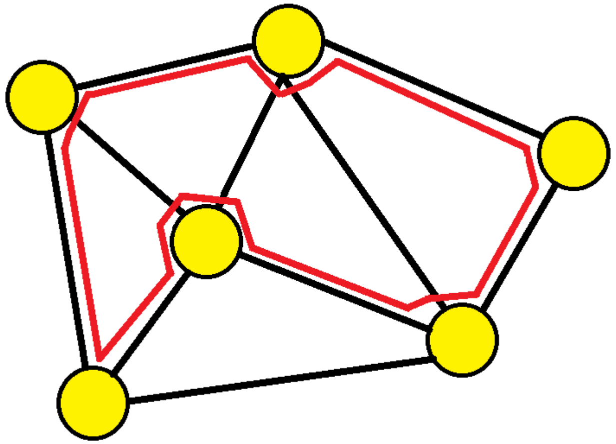
$\downarrow$

X3C: $\mathrm{U} = X \cup Y \cup Z$ (unordered)

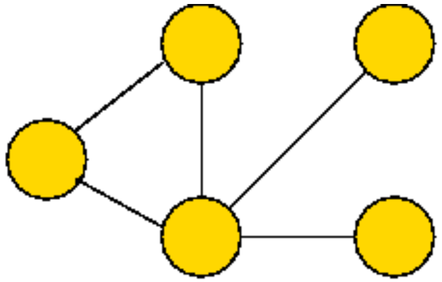
$M \subseteq T$ is a matching with size n for 3DM iff
$M \subseteq U$ is a 3-exact-cover for U

Hamiltonian Cycle Problem

We are given a unweighted and undirected graph $G = (V, E)$. Is there a cycle in G that visits each node exactly once?



Yes

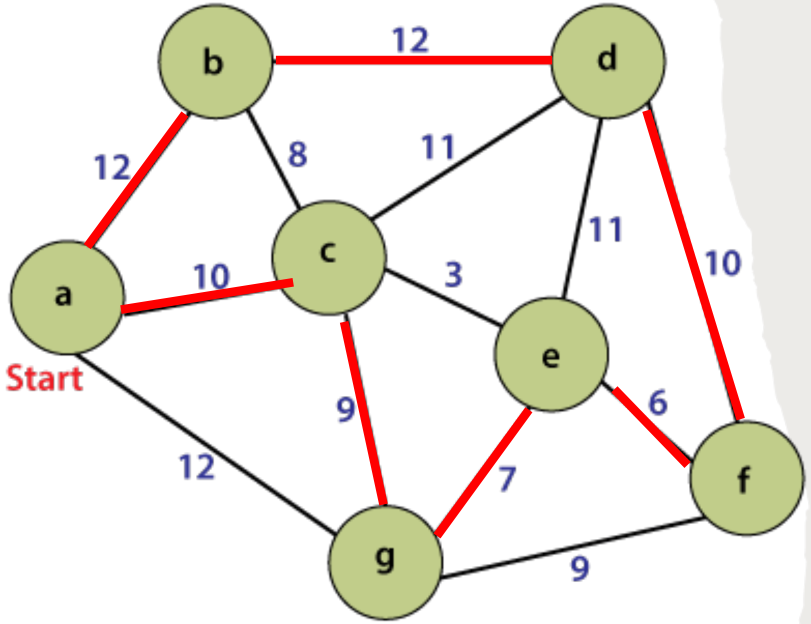


No

Travelling Salesman Problem (TSP)

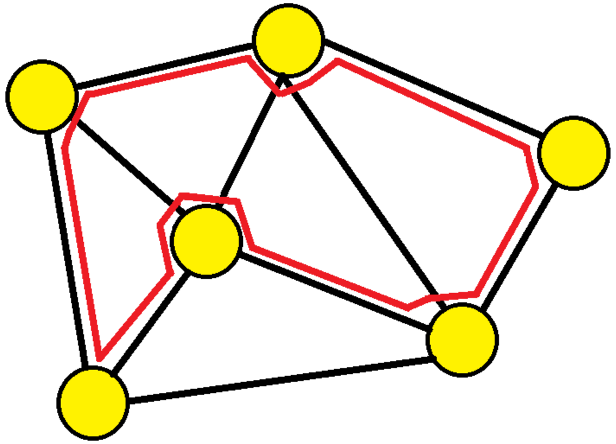
We are given n cities $1, \cdots, n$, and a nonnegative integer distance $l(i, j)$ between any two cities i and j (assume that the distances are symmetric, that is, $l(i, j) = l(j, i)$ for all i and j). We are also given a parameter K.

We are asked to determine if there is a tour of all cities with total distance no more than K.

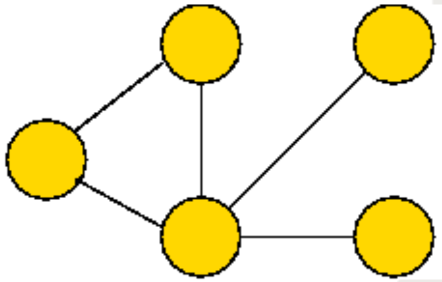


Hamiltonian Cycle Problem (undirected graph)

We are given a unweighted and undirected graph $G = (V, E)$. Is there a cycle in G that visits each node exactly once?



Yes

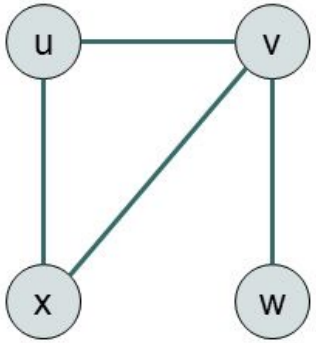


No

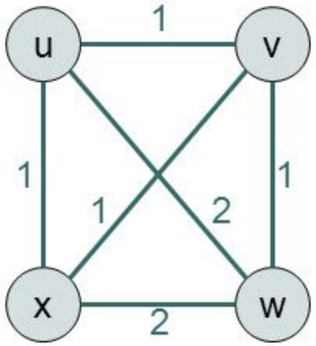
Travelling Salesman Problem (TSP)

Theorem. TSP is NP-complete.

G



K_n



Proof:

- TSP (decision version) is in NP.
- Reduction from Hamiltonian Cycle.
- Let G be an arbitrary undirected graph with n vertices.
- Construct a length function for K_n (complete graph) as follows

$$\ell(e) = \begin{cases} 1 & \text{if } e \text{ is an edge in } G, \\ 2 & \text{otherwise.} \end{cases}$$

- If G has a Hamiltonian cycle, then there is a TSP cycle in K_n whose length is exactly n;
- Otherwise, every TSP cycle in K_n has length at least $n + 1$.

- If G has a Hamiltonian cycle if and only if there is a TSP cycle in K_n whose length is exactly n.

Approximation Algorithms

Knapsack Problem:

KNAPSACK (Decision)

Given a set of items $X = \{a_1, \cdots, a_n\}$, cost $c_i \geq 0$, values $v_i \geq 0$, a budget B, and a target value V, is there a subset $S \subseteq X$ such that:

$$\sum_{i \in S} c_i \leq B \text{ and } \sum_{i \in S} v_i \geq V .$$

KNAPSACK (Optimization)

Given a set of items $X = \{a_1, \cdots, a_n\}$, cost $c_i \geq 0$, values $v_i \geq 0$, a budget B. Find a subset $S \subseteq X$ such that:

$$\sum_{i \in S} c_i \leq B,$$

and $v_i(S)$ is maximized.

Without loss of generality, we can assume $c_i \leq B$ for all a_i.

One More Dynamic Programming Algorithm for Knapsack

- $S_{i,p}$: ***subset*** of $\{a_1, \cdots, a_i\}$ with ***minimal*** cost which has value ***exactly*** p.
- $c(S_{i,p}) = \begin{cases} \infty & \text{if } S_{i,p} \text{ does not exist} \\ \sum_{a_i \in S_{i,p}} c(a_i) & \text{otherwise} \end{cases}$
- Consider the following recurrence:

$$S_{1,p} = \begin{cases} \{a_1\}, & \text{if } p = a_1 \\ -, & \text{otherwise} \end{cases}$$

$$S_{i+1,p} = \begin{cases} \arg\min\{c(S_{i,p}), c(\{a_{i+1}\} \cup S_{i,p-v_{i+1}})\}, & \text{if } v_{i+1} \le p \text{ and } S_{i,p-v_{i+1}} \neq - \\ S_{i,p}, & \text{otherwise} \end{cases}$$

- This recurrence computes all $S_{i,p}$ for i and $p \in \{0,1, \cdots, n \cdot V^*\}$ where $V^* = \max_i v_i$.
- The optimal solution to Knapsack is then given by

$$\arg \max_{p: c(S_{n,p}) \le B} v(S_{n,p}).$$

Dynamic Programming Algorithm for Knapsack

$$S_{1,p} = \begin{cases} a_1 & \text{if } p = v_i \\ \_ & \text{otherwise} \end{cases}$$

for $i \in \{1, \dots, n\}$ **do**

  for $p \in \{1, \dots, n \cdot V^*\}$ **do**

$$S_{i+1,p} = \begin{cases} \text{argmin}\,\{c(S_{i,p}), c(a_{i+1} \cup S_{i,p-v_{i+1}})\} & \text{if } v_{i+1} \leq p \text{ and } S_{i,p-v_{i+1}} \neq \_ \\ S_{i,p} & \text{otherwise} \end{cases}$$

  end for

end for

return $\text{argmax}_{p:c(S_{n,p}) \leq B}\, v(S_{n,p})$

polynomial time algorithm?

- The running time of this algorithm is $O(n^2 V^*)$.

number of bits

- NO! It is ***pseudo-polynomial!***
- Running time is measured by the ***size*** of the input. The value V^* can be written using $\log V^*$ bits, so a polynomial-time algorithm should have complexity $O(\log^c V)$ for some constant c.
- The complexity $O(n^2 V^*)$ is in fact exponential in the size of the input.

$\{215, 215, 275, 275, 355, 355, 420, 420, 580, 580, 655, 655\}$

Approximation Algorithm

$\{2, 2, 2, 2, 3, 3, 4, 4, 5, 5, 6, 6\}$

$k = 100$

Definition

For a maximization problem, an algorithm is called α-approximation if for any input, the algorithm returns a feasible solution S such that

$$\frac{f(S)}{f(O)} \geq \alpha,$$

$\alpha \leq 1$

where O is an optimal solution, and f evaluates the quality of the solution.

Polynomial-time approximation scheme (PTAS) for Knapsack

- Computes all $S_{i,p}$ for i and $p \in \{0,1,\cdots,n \cdot V^*\}$. $\longrightarrow$ Increase by 1 in each step.
  How about skipping some? Increase by $k > 1$?
- Let $0 < \epsilon < 1$ be a sufficiently small constant.
- Set $k = \frac{\epsilon \cdot V^*}{n}$.
- Set $v_i' = \left\lfloor \frac{v_i}{k} \right\rfloor$ for all $i = 1, \cdots, n$.
- Run the DP algorithm for $(\{(c_1, v_1'), \cdots, (c_n, v_n')\}, B)$.

$$O(n^2 \cdot \frac{V^*}{k}) = O(n^2 \cdot \frac{V^* \cdot n}{\epsilon \cdot V^*}) = O(\frac{n^3}{\epsilon})$$

How far away from the optimal solution?

Note: The analysis on this page will not be examined.

Approximation Ratio

- Let S be the solution returned by the algorithm (regarding $v'(\cdot)$).
- Let O be the optimal solution (regarding $v(\cdot)$).

Without loss of generality, we can assume $c_i \leq B$ for all a_i.

S is the optimal solution under $v'(\cdot)$

$|O| \leq n$

$OPT \geq V^*$

$$v(S) \geq k \cdot v'(S) \geq k \cdot v'(O) \geq v(O) - k \cdot |O| \geq v(O) - k \cdot n = OPT - \epsilon \cdot V^* \geq (1-\epsilon) \cdot OPT$$

by the definition of $v'(\cdot)$

$$v_i' = \left\lfloor \frac{v_i}{k} \right\rfloor$$

by the definition of $k = \frac{\epsilon \cdot V^*}{n}$ and $v(O) = OPT$

because of "round-down" operation of $v'(\cdot)$

$$v_i' = \left\lfloor \frac{v_i}{k} \right\rfloor \geq \frac{v_i}{k} - 1$$

$$\boxed{v(S) \geq (1-\epsilon) \cdot OPT}$$

A Few More Definitions

Strong NP-completeness

- The pseudo-polynomial algorithm does not establish that P = NP!
- The NP-completeness proof for SUBSET-SUM, KNAPSACK and PARTITION uses exponentially large integers.
- Problems including VERTEX-COVER, SET-COVER, INDEPENDET-SET, were shown NP-complete via reductions that constructed only polynomially small integers.

Strongly NP-complete

If a problem remains NP-complete even if any instance of length n is restricted to contain integers of size at most $p(n)$, a polynomial, then we say that the problem is ***strongly NP-complete***.

NP-hard

NP-complete

A problem $Y \in NP$ with the property that for every problem $X \in NP$, $X \leq_P Y$.

Recipe.

To prove that $Y \in$ NP-complete:

- Step 1. Show that $Y \in$ NP.
- Step 2. Choose an NP-complete problem X.
- Step 3. Prove that $X \leq_P Y$.

- Sometimes we may be able to show that all problems in NP polynomially reduce to some problem A, but we are unable to argue that A ∈ NP.
- So, A does not qualify to be called NP-complete.
- Yes, undoubtedly A is as hard as any problem in NP, and hence most probably intractable.

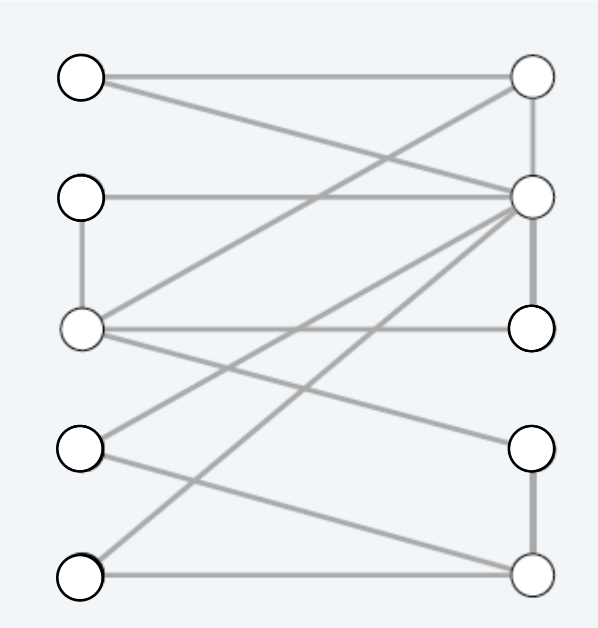
NP-hard

- The term NP-hard is also used in the literature to describe optimization problems (which are not decision problems and thus not in NP).

INDEPENDENT SET

Independent Set Problem

Given a graph $G = (V, E)$ with $|V| = n$ and $|E| = m$. Find a set of maximum number of vertices such that no two are adjacent?



Intuition

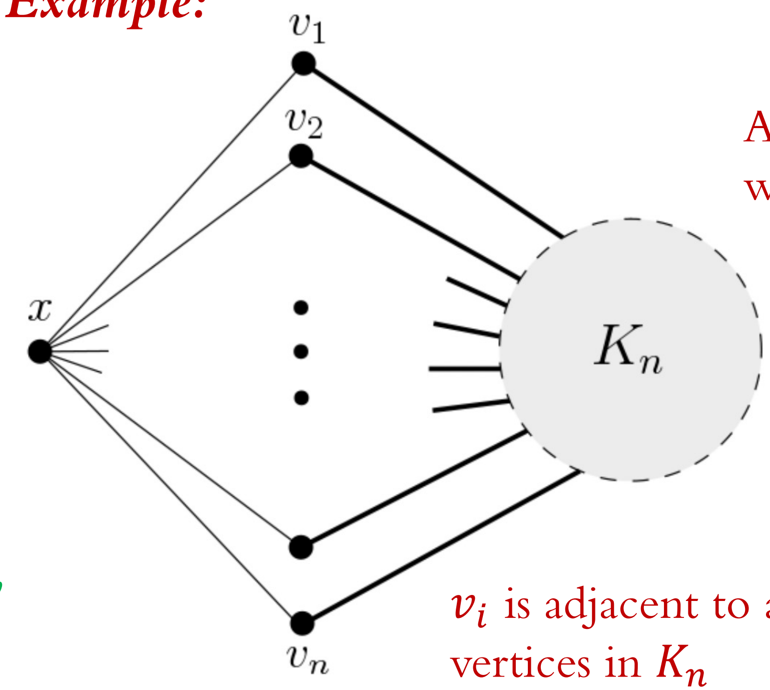
Always select the node with minimum degree.

Greedy Algorithm

Require: a graph $G = (V, E)$
  $W \leftarrow V$
  $S \leftarrow \emptyset$
  while $W \neq \emptyset$ **do**
    Find a vertex $v \in W$ with minimum degree in $G[W]$
    $W \leftarrow W \setminus N_G[v]$
    $S \leftarrow S \cup \{v\}$
  end while
  return S

the subset of vertices adjacent to v and v

A Bad Example:



A complete graph with n vertices

$OPT = n$

$ALG = 2$

v_i is adjacent to all vertices in K_n

Independent Set Problem

Greedy Algorithm

```
Require: a graph G = (V, E)
  W ← V
  S ← ∅
  while W ≠ ∅ do
    Find a vertex v ∈ W with minimum degree in G[W]
    W ← W \ N_G[v]
    S ← S ∪ {v}
  end while
  return S
```

Theorem.
The Greedy Algorithm is $(1/(\Delta+1))$-approximation for graphs with degree at most Δ.

Proof

- Every time a vertex v is picked by Greedy, at most Δ vertices are removed.
- So at the end at most $|S| \cdot (\Delta+1)$ vertices have been removed.
- All nodes have been removed:

$$n \leq (\Delta + 1) \cdot |S|$$

That is

$$|S| \geq \frac{n}{\Delta+1} \geq \frac{OPT}{\Delta+1}$$