



Design and analysis of algorithms

Lecture 36 & 37

Edyta Szymańska

edyta@cc.gatech.edu

Subset sum - numerical problem

Given a sequence of integers a_1, \dots, a_n and a parameter k , decide whether there is a subset of integers whose sum is exactly k .

Subset sum - numerical problem

Given a sequence of integers a_1, \dots, a_n and a parameter k , decide whether there is a subset of integers whose sum is exactly k .

Formally, does there exist a subset $I \subseteq \{1, \dots, n\}$ such that

$$\sum_{i \in I} a_i = k ?$$

Subset sum - numerical problem

Given a sequence of integers a_1, \dots, a_n and a parameter k , decide whether there is a subset of integers whose sum is exactly k .

Formally, does there exist a subset $I \subseteq \{1, \dots, n\}$ such that

$$\sum_{i \in I} a_i = k ?$$

Subset sum - numerical problem

Given a sequence of integers a_1, \dots, a_n and a parameter k , decide whether there is a subset of integers whose sum is exactly k .

Formally, does there exist a subset $I \subseteq \{1, \dots, n\}$ such that

$$\sum_{i \in I} a_i = k ?$$

Theorem *Subset Sum is NP-complete.*

Subset sum - numerical problem

Given a sequence of integers a_1, \dots, a_n and a parameter k , decide whether there is a subset of integers whose sum is exactly k .

Formally, does there exist a subset $I \subseteq \{1, \dots, n\}$ such that

$$\sum_{i \in I} a_i = k ?$$

Theorem *Subset Sum is NP-complete.*

Proof: Reduction from Vertex Cover.

From Vertex Cover to Subset Sum

Idea of the reduction:

- ⑥ Start from a graph G and a parameter k .
- ⑥ Create a set of integers and a parameter k' .
- ⑥ Prove that G has vertex cover of size k if and only if there is a subset of the integers that sum to k' .

From Vertex Cover to Subset Sum

Let G be given with $V = \{1, 2, \dots, n\}$ and some k .

From Vertex Cover to Subset Sum

Let G be given with $V = \{1, 2, \dots, n\}$ and some k . We want a construction with the following properties:

- ⑥ integers: two groups

$$a_i\text{'s } \forall i \in V$$

$$b_{ij}\text{'s } \forall (i, j) \in E$$

- ⑥ parameter k'

- ⑥ if we have a subset I of the integers that sums up to k' , then a_i 's in I correspond to a **vertex cover** C in G , and the b_{ij} 's correspond to edges in G which have exactly one endpoint in C .

- ⑥ the construction will force $|C| = k$.

From Vertex Cover to Subset Sum

Defining the integers:

From Vertex Cover to Subset Sum

Defining the integers:

- ⑥ create a matrix with $n + m$ rows corresponding to all a_i 's and b_{ij} 's and $m + 1$ columns, corresponding to b_{ij} 's plus one extra column X
- ⑥ in X put a 1 for all a_i 's and a 0 otherwise
- ⑥ in column (ij) put a 1 for a_i, a_j and b_{ij} and 0 elsewhere.
- ⑥ define $k' := k \cdot 4^m + \sum_{i=0}^{m-1} 2 \cdot 4^i$.

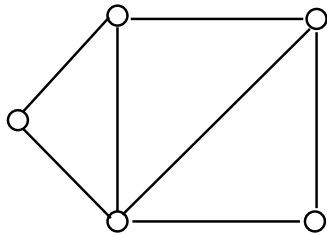
From Vertex Cover to Subset Sum

Defining the integers:

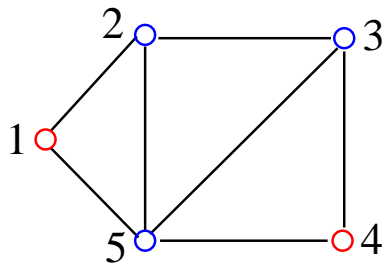
- ⑥ create a matrix with $n + m$ rows corresponding to all a_i 's and b_{ij} 's and $m + 1$ columns, corresponding to b_{ij} 's plus one extra column X
- ⑥ in X put a 1 for all a_i 's and a 0 otherwise
- ⑥ in column (ij) put a 1 for a_i, a_j and b_{ij} and 0 elsewhere.
- ⑥ define $k' := k \cdot 4^m + \sum_{i=0}^{m-1} 2 \cdot 4^i$.

Each row is a representation of an integer in base 4.

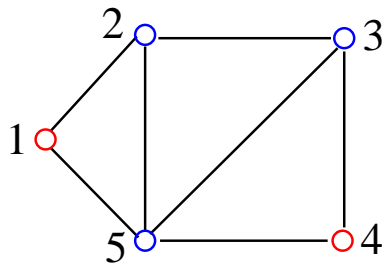
From Vertex Cover to Subset Sum



From Vertex Cover to Subset Sum

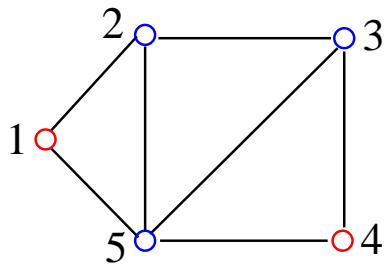


From Vertex Cover to Subset Sum



	X	12	15	25	23	35	45	34
1	1	1	1	0	0	0	0	0
2	1	1	0	1	1	0	0	0
3	1	0	0	0	1	1	0	1
4	1	0	0	0	0	0	1	1
5	1	0	1	1	0	1	1	0
12	0	1	0	0	0	0	0	0
15	0	0	1	0	0	0	0	0
25	0	0	0	1	0	0	0	0
23	0	0	0	0	1	0	0	0
35	0	0	0	0	0	1	0	0
45	0	0	0	0	0	0	1	0
34	0	0	0	0	0	0	0	1

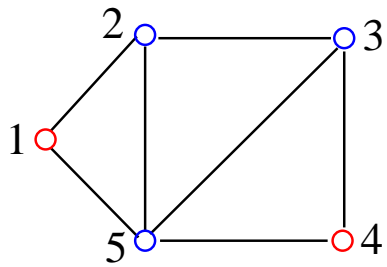
From Vertex Cover to Subset Sum



$$k' = 3 \cdot 4^7 + 2 \cdot \sum_{i=0}^{7-1} 4^i = 60074$$

	X	12	15	25	23	35	45	34
1	1	1	1	0	0	0	0	0
2	1	1	0	1	1	0	0	0
3	1	0	0	0	1	1	0	1
4	1	0	0	0	0	0	1	1
5	1	0	1	1	0	1	1	0
12	0	1	0	0	0	0	0	0
15	0	0	1	0	0	0	0	0
25	0	0	0	1	0	0	0	0
23	0	0	0	0	1	0	0	0
35	0	0	0	0	0	1	0	0
45	0	0	0	0	0	0	1	0
34	0	0	0	0	0	0	0	1

From Vertex Cover to Subset Sum



$$k' = 3 \cdot 4^7 + 2 \cdot \sum_{i=0}^{7-1} 4^i = 60074$$

	X	12	15	25	23	35	45	34	
1	1	1	1	0	0	0	0	0	21504
2	1	1	0	1	1	0	0	0	20800
3	1	0	0	0	1	1	0	1	16704
4	1	0	0	0	0	0	1	1	16389
5	1	0	1	1	0	1	1	0	17684
12	0	1	0	0	0	0	0	0	4096
15	0	0	1	0	0	0	0	0	1024
25	0	0	0	1	0	0	0	0	256
23	0	0	0	0	1	0	0	0	64
35	0	0	0	0	0	1	0	0	16
45	0	0	0	0	0	0	1	0	4
34	0	0	0	0	0	0	0	1	1

Reduction - correctness

From Covers to Subsets:

Reduction - correctness

From Covers to Subsets:

- ⑥ Suppose \exists vertex cover C , $|C| = k$.
- ⑥ Choose a_i 's and b'_{ij} 's as described.
- ⑥ When we sum up these integers (in base 4), then we have a 2 in all digits except for column X . Then we have a 1 in X k times.
- ⑥ Thus the sum is k' .

Reduction - correctness

From Subsets to Covers:

Reduction - correctness

From Subsets to Covers:

- ⑥ Suppose we find a subset $C \subseteq V$ and $E' \subseteq E$ s.t.

$$\sum_{i \in C} a_i + \sum_{(i,j) \in E'} b_{ij} = k'.$$

- ⑥ Observe that we never have a carry in the m less significant digits (≤ 3 ones in every column - that's why base 4 works!)
- ⑥ b_{ij} 's can contribute at most 1 in every column and k' has a 2 in all the m less significant digits and thus, for every (i, j) we have either $i \in C$ or $j \in C$ and C is a cover.
- ⑥ $\forall i, a_i \geq 4^m$, and k' gives a quotient of k when divided by 4^m . Thus C cannot contain more than k elements.

Unary subset sum is in P

Unary Subset sum: given integers a_1, a_2, \dots, a_n and an integer K , in unary, is there a subset of these integers that sum exactly to K ?

Unary subset sum is in P

Unary Subset sum: given integers a_1, a_2, \dots, a_n and an integer K , in unary, is there a subset of these integers that sum exactly to K ?

Input size: $n + K$ bits

Unary subset sum is in P

Unary Subset sum: given integers a_1, a_2, \dots, a_n and an integer K , in unary, is there a subset of these integers that sum exactly to K ?

Input size: $n + K$ bits

Solution: dynamic programming (see the algorithm for Knapsack Problem) will solve it in time $O(n^2 K)$.

Unary subset sum is in P

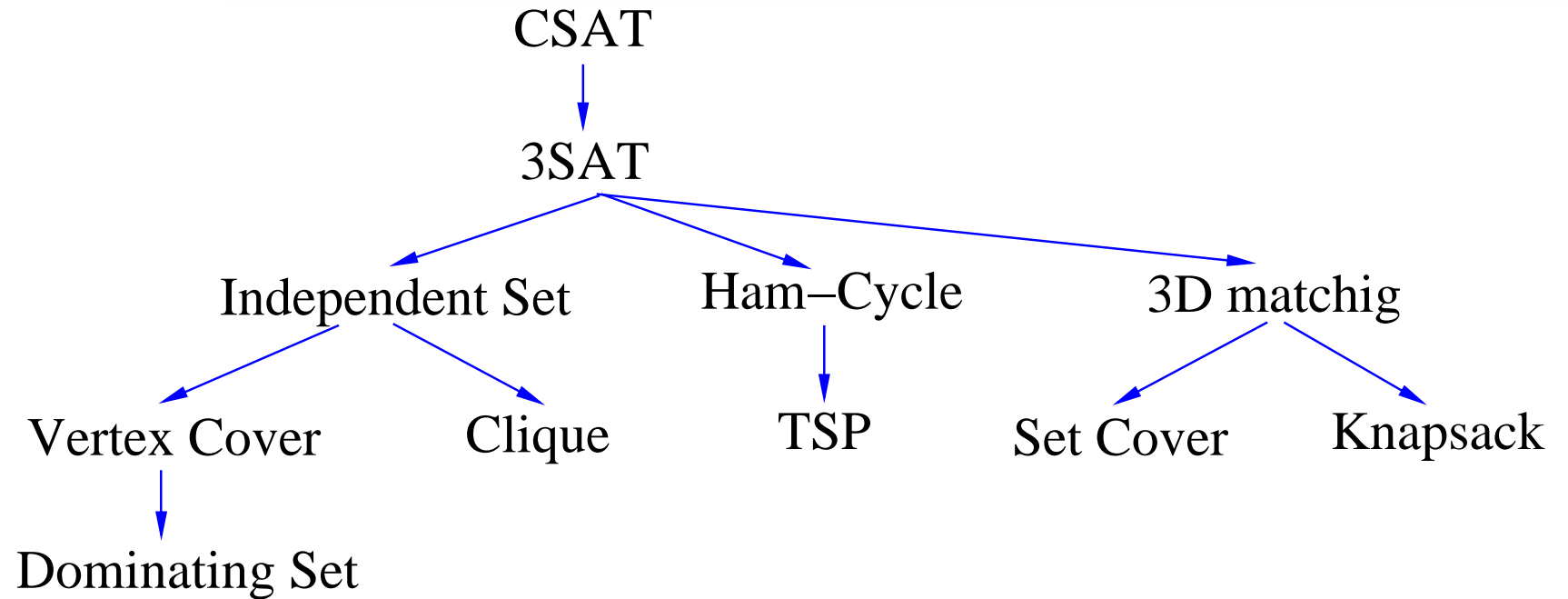
Unary Subset sum: given integers a_1, a_2, \dots, a_n and an integer K , in unary, is there a subset of these integers that sum exactly to K ?

Input size: $n + K$ bits

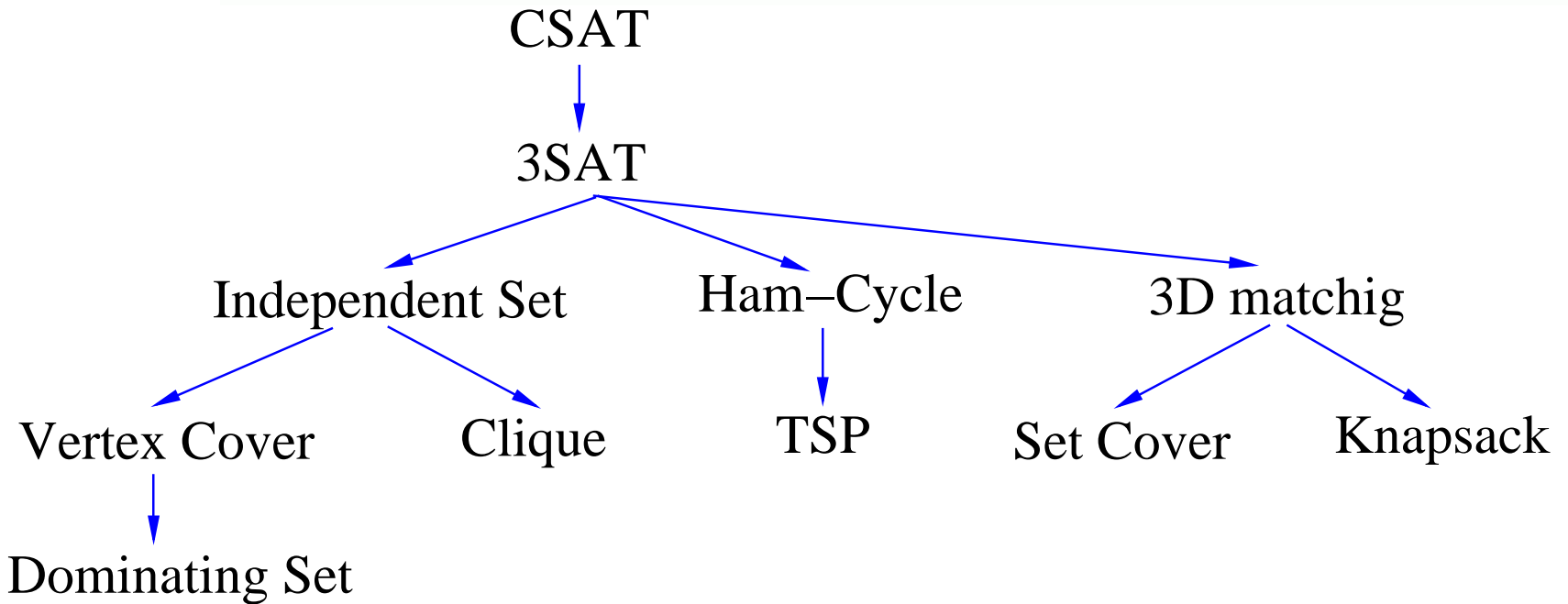
Solution: dynamic programming (see the algorithm for Knapsack Problem) will solve it in time $O(n^2 K)$.

$O(n^2 K)$ is polynomial in the size of input in unary but exponential if K is in binary.

More NP-complete problems



More NP-complete problems



See: *Computers and Intractability: A guide to the theory of NP-completeness* by Michael R. Garey and David S. Johnson, (W. H. Freeman, 1979.)

An Epilogue - Undecidability

Are there problems that are not even in NP?

An Epilogue - Undecidability

Are there problems that are not even in NP?
YES !

An Epilogue - Undecidability

Are there problems that are not even in **NP**?

YES !

There are problems for which there are **NO ALGORITHMS** at all.

An Epilogue - Undecidability

Are there problems that are not even in NP?

YES !

There are problems for which there are NO ALGORITHMS at all.

Example:

There is barber in a town who shaves every man who doesn't shave himself.

An Epilogue - Undecidability

Are there problems that are not even in NP?

YES !

There are problems for which there are NO ALGORITHMS at all.

Example:

There is barber in a town who shaves every man who doesn't shave himself.

Who shaves the barber ?

The halting problem

Consider a task to write the following Boolean function:

`term(P, X)`

- ⑥ `P` is a program in the same language
- ⑥ `X` is a data file.
- ⑥ `term(P, X)` returns
 - △ TRUE if program `P` with input file `X` eventually terminates
 - △ FALSE if program `P` loops on file `X` forever.

The halting problem

Consider a task to write the following Boolean function:

`term(P, X)`

- ⑥ `P` is a program in the same language
- ⑥ `X` is a data file.
- ⑥ `term(P, X)` returns
 - △ TRUE if program `P` with input file `X` eventually terminates
 - △ FALSE if program `P` loops on file `X` forever.

Is this task possible to perform?

The Halting Problem

NO!
Proof:



The Halting Problem

NO!

Proof:

Suppose that we have written such a Boolean function and used it in the following program:

```
Boolean function diag(P)  
if term(P,P) then loop forever.
```

The Halting Problem

NO!

Proof:

Suppose that we have written such a Boolean function and used it in the following program:

```
Boolean function diag(P)  
if term(P,P) then loop forever.  
Run diag(diag). Does it terminate?
```

The Halting Problem

NO!

Proof:

Suppose that we have written such a Boolean function and used it in the following program:

```
Boolean function diag(P)  
if term(P,P) then loop forever.
```

Run `diag(diag)`. Does it terminate?

It does if it does not !

The Halting Problem

NO!

Proof:

Suppose that we have written such a Boolean function and used it in the following program:

```
Boolean function diag(P)  
if term(P,P) then loop forever.
```

Run `diag(diag)`. Does it terminate?

It does if it does not !

Conclusion: There is no program that can be written to solve the termination problem.