

Complexity and Problem Reduction

– how difficult can it be

David Pisinger
 pisinger@man.dtu.dk
 DTU Management
 Technical University of Denmark

$\mathcal{P}$ and $\mathcal{NP}$

What is more difficult?

- To solve a mathematical problem
- To check the solution to a problem

$$\begin{array}{rclcl} 2x & + & 3y & + & 4z & = & 3 \\ 3x & - & y & + & z & = & 4 \\ -x & + & 3y & - & z & = & 2 \end{array}$$

$$x = 2, \quad y = 1, \quad z = -1$$

$\mathcal{P}$ and $\mathcal{NP}$



Easy and hard problems

$\mathcal{P}$ (easy)	$\mathcal{NP}$ -complete (hard)
Euler cycle Given a graph $G = (V, E)$. Is there a cycle which visits each edge exactly once?	Hamilton cycle Given a graph $G = (V, E)$. Is there a cycle which visits each node exactly once?
Shortest path Given a weighted graph $G = (V, E, c)$. Is there a path from a to b with length $\leq k$?	Longest path Given a weighted graph $G = (V, E, c)$. Is there a path from a to b with length $\geq k$?
Edge cover Given a graph $G = (V, E)$. Is it possible to choose $\leq k$ edges so every node is incident to a chosen edge?	Node cover Given a graph $G = (V, E)$. Is it possible to choose $\leq k$ nodes so every edge is incident to a chosen node?
Chinese postman Given a weighted graph $G = (V, E, c)$. Is there a cycle which visits each edge at least once and has length $\leq k$?	Traveling salesman Given a weighted graph $G = (V, E, c)$. Is there a cycle which visits each node at least once and has length $\leq k$?

Complexity and Problem Reduction

Real world problems may be easy or hard. Complexity theory is an attempt to formalize the classification.

- **Easy problems:** MST, shortest-path, max-flow, min cost flow, uncapacitated lot sizing
- **Hard problems:** 0-1 knapsack, set covering, set partitioning, uncapacitated facility location, general IP

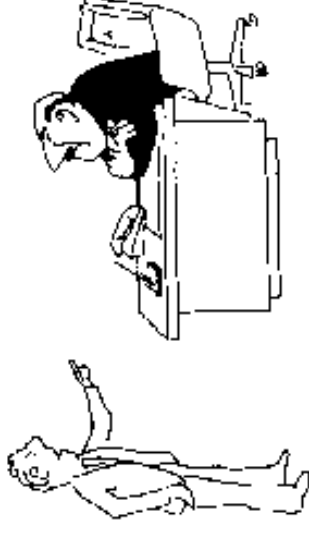
Exact meaning of easy and hard?

- Easy: construct solution in polynomial time
 - Hard: Search through (almost) all possible solutions
- Is a problem difficult just because we have not (yet) found a constructive solution?

$$ax^2 + bx + c = 0$$

Complexity and Problem Reduction – p.5/32

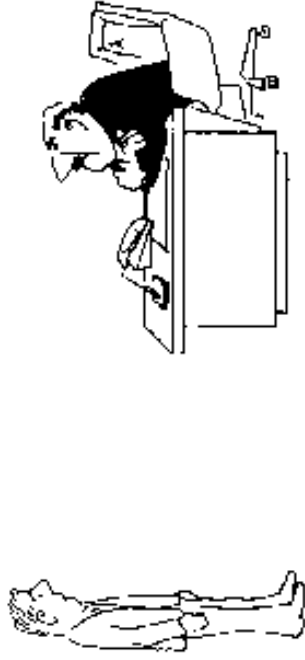
Garey and Johnson



I can't find an efficient algorithm, because no such algorithm is possible

Complexity and Problem Reduction – p.7/32

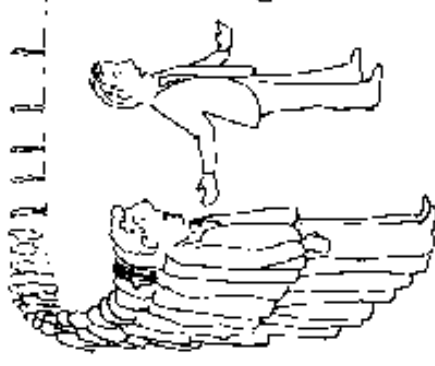
Garey and Johnson



I can't find an efficient algorithm, I guess I'm just too dumb.

Complexity and Problem Reduction – p.6/32

Garey and Johnson



I can't find an efficient algorithm, but neither can all these famous people.

Complexity and Problem Reduction – p.8/32

Classification

In order to be able to classify problems one needs:

- The class of decision problems $\mathcal{NP}$
- The class of easy problems $\mathcal{P} \subseteq \mathcal{NP}$
- The class of hard problems $\mathcal{NPC} \subseteq \mathcal{NP}$
- A relation "no more difficult than" between $P, Q \in \mathcal{NP}$

Complexity and Problem Reduction – p.9/32

Easy problems - the class $\mathcal{P}$

- $\mathcal{P}$: problems which can be solved by polynomial algorithm
- Consider an algorithm to solve instance I of problem P
 - $f_A(I)$ is the number of elementary operations to solve I
 - $|I|$ is the size of the the instance, i.e. the number of digits used to represent I

• **polynomial**

$$f_A(|I|) \in O(|I|^k) \text{ for some } k$$

- Obviously $\mathcal{P} \subseteq \mathcal{NP}$

Complexity and Problem Reduction – p.11/32

Decision problems, the class $\mathcal{NP}$

Each instance may be answered by "yes" or "no"

- Does a given IP have a feasible solution?
- Does a given graph $G = (V, E)$ have a MST of weight $\leq k$ (the threshold k is part of the problem data)

An easy verification of a "yes" answer is a certificate

- If a given IP is a "yes" instance, this can be verified by checking a feasible solution against the constraints.
- If an MST is a "yes" instance, this can be verified by checking a set of $n - 1$ edges for whether they form a tree and whether their total weight is $\leq k$

Complexity and Problem Reduction – p.10/32

Size of instance is important

Knapsack problem as decision problem

$$\begin{aligned} \sum_{j=1}^n p_j x_j &\geq k \\ \sum_{j=1}^n w_j x_j &\leq c \\ x &\in \{0, 1\}^n \end{aligned}$$

- decision problem in $\mathcal{NP}$ (easy to check solution)
 - Size of I is the number of digits in input
- $$\sum \lceil \log_{10} p_i \rceil + \sum \lceil \log_{10} w_i \rceil + \lceil \log_{10} k \rceil + \lceil \log_{10} c \rceil$$
- Dynamic programming algorithm $O(nc)$
 - $c = 10^{\log_{10} c}$ not polynomial in $\log_{10} c$

Complexity and Problem Reduction – p.12/32

Knapsack Problem, example

Instance

j	1	2	3	4	5
p_j	5	5	2	6	1
w_j	6	7	3	7	2

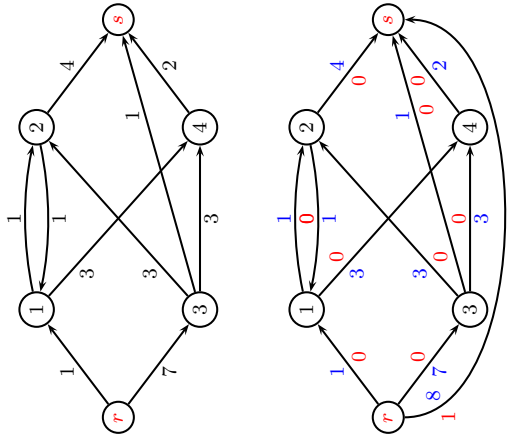
$c = 14$

Dynamic programming table

$\lambda \backslash r = 0$	$r = 1$	$r = 2$	$r = 3$	$r = 4$	$r = 5$
0	0	0	0	0	0
1	0	0	0	0	0
2	0	0	0	0	1
3	0	0	2	2	2
4	0	0	2	2	2
5	0	0	2	2	3
6	0	5	5	5	5
7	0	5	5	6	6
8	0	5	5	6	7
9	0	5	5	7	8
10	0	5	5	7	8
11	0	5	5	7	9
12	0	5	5	7	8
13	0	5	10	10	11
14	0	5	10	10	11

Problem transformation

Max Flow to Min Cost Flow



Is $\mathcal{NP} = \mathcal{P}$



CLAY MATHEMATICS INSTITUTE
Dedicated to increasing and disseminating mathematical knowledge

MILLENNIUM PRIZE PROBLEMS

Statement from the Directors and Scientific Advisory Board

| Birch and Swinnerton-Dyer Conjecture | Hodge Conjecture | Navier-Stokes Equations | P vs NP |
Poincaré Conjecture | Riemann Hypothesis | Yang-Mills Theory | Rules etc |



In order to celebrate mathematics in the new millennium, The Clay Mathematics Institute of Cambridge, Massachusetts (CMI) has named seven Millennium Prize Problems. The Scientific Advisory Board of CMI selected these problems, focusing on important classic questions that have resisted solution over the years. The Board of Directors of CMI have designated a \$7 million prize fund for the solution to these problems, with \$1 million allocated to each. During the Millennium meeting held on May 24, 2000 at the Collège de France, Timothy Gowers presented a lecture entitled "The Importance of Mathematics," aimed for the general public, while John Tate and Michael Atiyah spoke on the problems. The CMI invited specialists to formulate each problem.



Millennium Prize
Problem.iam

Problem transformation

Max Flow to Min Cost Flow

Which problem is harder?

We write

$$\text{MAXFLOW} \leq_{pol} \text{MINCOSTFLOW}$$

Transformation must be polynomial in input size $|I|$

Knapsack to Shortest Path

$$\max \left\{ \sum_{j=1}^n p_j x_j : \sum_{j=1}^n w_j x_j \leq c, x_j \in \{0, 1\} \right\}$$

Transform to minimization problem
(minimize profit of items not chosen)

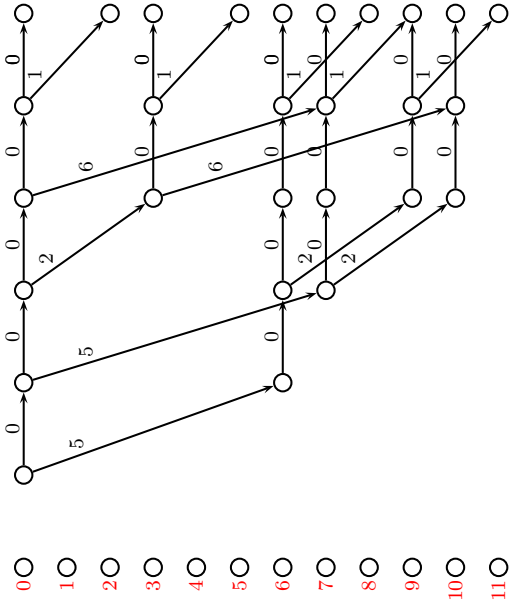
$$\min \left\{ \sum_{j=1}^n p_j x_j : \sum_{j=1}^n w_j x_j \geq \sum_{j=1}^n w_j - c = \hat{c}, x_j \in \{0, 1\} \right\}$$

Example

j	1	2	3	4	5
p_j	5	5	2	6	1
w_j	6	7	3	7	2

$$\hat{c} = 25 - 14 = 11$$

Knapsack to Shortest Path



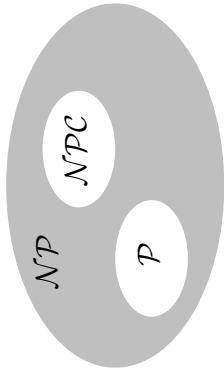
Proving $\mathcal{NP}$ -completeness

A problem P is $\mathcal{NP}$ -complete $\Leftrightarrow$

- (1) $P \in \mathcal{NP}$
- (2) $\forall Q \in \mathcal{NP} : Q \leq_{pol} P$

Proving (2) can be difficult
Instead choose problem $R \in \mathcal{NPC}$ and prove $R \leq_{pol} P$

$$\forall Q \in \mathcal{NP} : Q \leq_{pol} R \leq_{pol} P$$



Proving $\mathcal{NP}$ -completeness

Technique for proving problem $P \in \mathcal{NPC}$

1. Prove that $P \in \mathcal{NP}$ (i.e. can be verified in pol time)
2. Choose a known $\mathcal{NP}$ -complete problem R
3. Show a mapping $f : R \mapsto P$
4. Prove that f satisfies that if x is a "yes"-instance for R
 $\Leftrightarrow f(x)$ is a "yes" instance for P .
5. Prove that f is polynomial

Knapsack is NP-complete

KP in decision form (after adding slack variables)

$$\text{KPDECISION}(p, w, C, h) = \left\{ \begin{array}{l} \text{there exists an } x \text{ with} \\ \sum_{j=1}^n p_j x_j \geq h, \\ \sum_{j=1}^n w_j x_j = C, \\ x_j \in \{0, 1\}, j = 1, \dots, n \end{array} \right\}$$

Binary IP in decision form (after adding slack variables)

$$\text{BIPDECISION}(a, b, c, k) = \left\{ \begin{array}{l} \text{there exists an } x \text{ with} \\ \sum_{j=1}^n c_j x_j \geq k, \\ \sum_{j=1}^n a_{ij} x_j = b_i, i = 1, \dots, m, \\ x_j \in \{0, 1\}, j = 1, \dots, n \end{array} \right\}$$

where a_{ij} is an $n \times m$ integer matrix

Binary Integer Programming

Numerous problems can be formulated as binary integer programming models e.g.

- set-covering problem
- travelings salesman problem
- Hamilton cycle problem
- lot sizing problem

BIPDECISION is NP-complete

Transformation

$$\begin{array}{l} 6x_1 + 8x_2 + 3x_3 + 7x_4 + 6x_5 + 5x_6 + 2x_7 \geq 9 \\ 0x_1 + 1x_2 + 1x_3 + 0x_4 + 1x_5 + 0x_6 + 1x_7 = 2 \\ 3x_1 + 3x_2 + 0x_3 + 2x_4 + 2x_5 + 1x_6 + 2x_7 = 8 \\ 1x_1 + 2x_2 + 1x_3 + 0x_4 + 3x_5 + 1x_6 + 4x_7 = 7 \end{array}$$

Write constraints column-wise

1	3	0	x_1
2	3	1	x_2
1	0	1	x_3
0	2	0	x_4
3	2	1	x_5
1	1	0	x_6
4	2	1	x_7
7	8	2	

Transformation

Transformed problem

1	0	3	0	0	x_1
2	0	3	0	1	x_2
1	0	0	0	1	x_3
0	0	2	0	0	x_4
3	0	2	0	1	x_5
1	0	1	0	0	x_6
4	0	2	0	1	x_7

Resulting KP problem

$$\begin{array}{l} 6x_1 + 8x_2 + 3x_3 + 7x_4 + 6x_5 + 5x_6 + 2x_7 \geq 9 \\ 10300x_1 + 20301x_2 + 10001x_3 + 00200x_4 + 30201x_5 + 10100x_6 + 40201x_7 = 70802 \end{array}$$

Transformation in general

$$f : \text{BIPDECISION}(a, b, c, k) \mapsto \text{KPDECISION}(p, w, C, h)$$

Largest coefficient: $M = \max_{i,j} |a_{ij}|$

Number of zeros to add: $Z = \lceil \log_{10}(nM) \rceil$

$$p_j := c_j$$

$$w_j := \sum_{i=1}^m a_{ij} \cdot 10^{Z \cdot (i-1)}$$

$$C := \sum_{i=1}^m b_i \cdot 10^{Z \cdot (i-1)}$$

$$h := k$$

(Also works for negative coefficients)

Summing up proof

We have

1. KPDECISION $\in \mathcal{NP}$ (can be verified in pol time)
2. chosen a known $\mathcal{NP}$ -complete problem BIPDECISION
3. mapping $f : \text{BIPDECISION} \mapsto \text{KPDECISION}$
4. a "yes" instance for KPDECISION is a "yes" instance for BIPDECISION
5. f is polynomial

KPDECISION is $\mathcal{NP}$ -complete

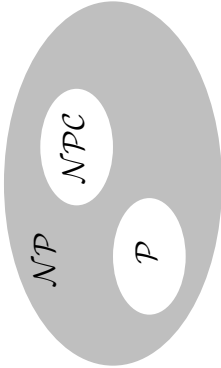
Equivalent problems

$$\begin{array}{r} 6x_1 + 8x_2 + 3x_3 + 7x_4 + 6x_5 + 5x_6 + 2x_7 \geq 9 \\ 0x_1 + 1x_2 + 1x_3 + 0x_4 + 1x_5 + 0x_6 + 1x_7 = 2 \\ 3x_1 + 3x_2 + 0x_3 + 2x_4 + 2x_5 + 1x_6 + 2x_7 = 8 \\ 1x_1 + 2x_2 + 1x_3 + 0x_4 + 3x_5 + 1x_6 + 4x_7 = 7 \end{array}$$

1	0	3	0	0	x_1
2	0	3	0	1	x_2
1	0	0	0	1	x_3
0	0	2	0	0	x_4
3	0	2	0	1	x_5
1	0	1	0	0	x_6
4	0	2	0	1	x_7
7	0	8	0	2	

$$(x_1 = x_2 = x_7 = 1) \text{ or } (x_2 = x_4 = x_6 = x_7 = 1)$$

NP-hard



- $\mathcal{NP}$ only defined for decision problems
- we say that optimization problem is $\mathcal{NP}$ -hard if corresponding decision problem is $\mathcal{NP}$ -complete

Conclusion

- $\mathcal{NP} = \mathcal{P}$ is one of the main questions in math
- The world is full of hard problems
- It is difficult to distinguish easy and hard problems
- 5-step technique for proving problem is $\mathcal{NP}$ -complete
- Reduction (transformation) is a general technique
- Use right techniques to right problems

The growth of functions

	10	50	100	300	1000
$5n$	50	250	100	1500	5000
$n \log_2 n$	33	282	665	2469	9966
n^2	100	2500	10.000	90.000	7 digits
n^3	1000	125.000	7 digits	8 digits	10 digits
2^n	1024	16 digits	31 digits	91 digits	302 digits
$n!$	7 digits	65 digits	161 digits	623 digits	"unimaginable"
n^n	11 digits	85 digits	201 digits	744 digits	"unimaginable"

Atoms in the universe: $10^{78} - 10^{82}$

Time since big-bang: 13.73 billion years = 10^{18} seconds

The growth of functions

When is an algorithm faster than another algorithm?

- Independent of hardware
- Independent of small details (32-bit vs. 64 bit integer)
- Independent of programming language
- Behavior for large input (asymptotic)

Count asymptotic number of operations (multiplication, addition, etc) ignoring constants

The growth of functions

Big-O notation, asymptotic upper bound

A function f is

- $O(n)$ if exists constant k so that $f(n) \leq kn$ for large n
- $O(n^2)$ if exists constant k so that $f(n) \leq kn^2$ for large n
- $O(n^c)$ if exists constant k so that $f(n) \leq kn^c$ for large n
- $O(2^n)$ if exists constant k so that $f(n) \leq k \cdot 2^n$ for large n

What if constant k is very large e.g. $n^3 < 10^{10} n^2$?

- Appears seldom in practice
- For (very) large values of n still better