Advanced Algorithm Design and Analysis

1. Introduction

Mingyu XIAO（肖鸣宇）
School of Computer Science and Engineering
University of Electronic Science and Technology of China

Turing Award

The first Turing Award was given in 1966.
Up to 2022, there are 75 winners in total.

Edsger W. Dijkstra
(1972)

姚期智
(2000)

Stephen A. Cook
(1982)

Yann LeCun **Geoffrey Hinton** **Yoshua Bengio**
（2018）

John Hopcroft
(1986)

Richard M. Karp
(1985)

Donald Knuth
(1974)

Robert Tarjan
(1986)

1.1 Course Information

Course Information

Lecture time and venue:

Time: Monday (9-11), Thursday (5-6). The first eight weeks

Venue: B409, Liren building (Qingshuihe campus)

Instructor:

Dr. XIAO Mingyu (肖鸣宇)

Email: myxiao@gmail.com

Tel: 153-97626165

Teaching assistants:

Tian Bai (白天）

Junqiang Peng（彭俊强）

QQ group: 716389391

2022年8月26日，新生开学期，学校西门

2021年12月26日，研究生入学考试

记住自己曾经战斗过的地方！

送给大家一句话：

靠自己！

Course Information

Syllabus:

Introduction (2 hours)

Basics of algorithm design & analysis (8 hours) ← 算法设计与分析基础

Flows (4 hours) ← 网络流

NP-Completeness and Appro. alg. (16 hours) ← NP完备性和近似算法

Advance topics (6 hours) ← 算法高级讲座

Others (4 hours) ← 习题课

Textbook:

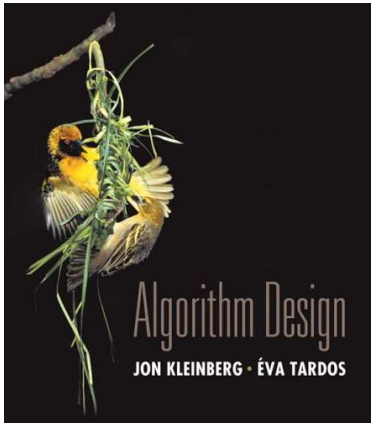
J. Kleinberg， E. Tardos, Algorithm design，Addison Wesley， 2005

Reference:

Cormen，Leiserson，Rivest，Stein，算法导论（影印版），高等教育出版社，2007

Grading Scheme:

Final exam 70% Assignments and others 30%



Course Information

Data from 2021 fall:

考试 **155**

考试160人	90分	80分	70分	60分	不及格	未参加考试
	10	47	69	23	0	6
	6%	30%	45%	15%	0%	4%

Data from 2020 fall:

考试 **160**

考试160人	90分	80分	70分	60分	不及格	未参加考试
	8	48	71	20	2	11
	5%	30%	44%	13%	1%	7%

Course Information

Data from 2019 fall:

考试 **160**

考试160人	90分	80分	70分	60分	不及格	未参加考试
	8	50	54	27	9	12
	5%	31%	34%	17%	6%	8%

Data from 2018 spring:

考试 **202** 考查 **46**

考试202人	90分	80分	70分	60分	不及格		考查 46人
	28	78	38	17	4+24		35人通过
	13.9%	38.6%	18.8%	8.4%	14.7%		76.1%

Course Information

2021年

96分，项浩哲，电子信息
95分，彭俊强，计算机科学与技术
熊子良，计算机科学与技术
94分，刘雨曦，计算机科学与技术
92分，白天，　计算机科学与技术
91分，梁子辉，　计算机科学与技术

2020年

94分，赵景阳，计算机科学与技术
93分，高梓翔，计算机科学与技术
92分，黄超，　计算机科学与技术

2019年

92分，李杰，计算机科学与技术
陈果，计算机技术
91分，王嘉唯，计算机技术
胡兴航 ，计算机科学与技术

2018年

94.2分，周连明，计算机科学与技术
93.4分，施钦凯，通信与信息系统
92.9分，陈静，　计算机技术

Course Information

Advices to who wants to register this course:

1. This course is worthy of your time. You will be proud of yourself when you fall in this course.
2. This course may be a heavy load for you, if you donot want to spend time on it. So DON'T select this course if you just want to get the credit.
3. Read the lecture notes carefully, which will be more helpful for you than the text books.
4. Make sure that you can solve the problems that I put highlines in the course.
5. In some interviews of job hunting, you may find the techniques in the course useful.

1.2 Introduction

Computers v.s. Tractors

What is the difference between the two machines?

V.S.

Computers more powerful?

Computer Problems

Problem: A task to be performed by computers

- Problems ⇔ Mathematical function from inputs to matching outputs.
- A particular input must always result in the same output every time the function is computed
- Problem definition should include constraints on the resources that may be consumed by any acceptable solution

Three Kinds of Problems

Decision Problem (with yes-no answers)

e.g. Is there a solution better than some given bound?

Optimal Value/Optimal Solution

e.g. What is the value of a best possible solution?
e.g. Find a solution that achieves the optimal value

Numerical Calculation

e.g. Solving an equation group

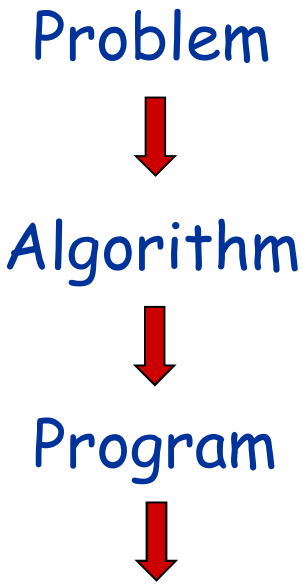
Algorithms

What is an algorithm?

1. a method or a process followed to solve a problem using a computer

2. takes the input of a problem and transforms it to the output

3. A problem can have many algorithms



Programs and algorithms

- A computer program is an instance, or concrete representation, for an algorithm in some programming language.

- A program is to be read by computer
- An algorithm is to be read by human being
- Algorithms can be expressed by pseduocodes or just some short steps that are easy to read and understand

Goals of Algorithm Design

- We want to use computes to solve problems
  --To solve problems under the recourse constrains

Two 'conflicting' goals :
To design an algorithm that is easy to understand, code and debug.
To design an algorithm that makes efficient use of the computer's resources.

Good Algorithms Always Exist?

- Some problems have good algorithms and can be solved quickly. Some may not have (we have not found good algorithms yet).

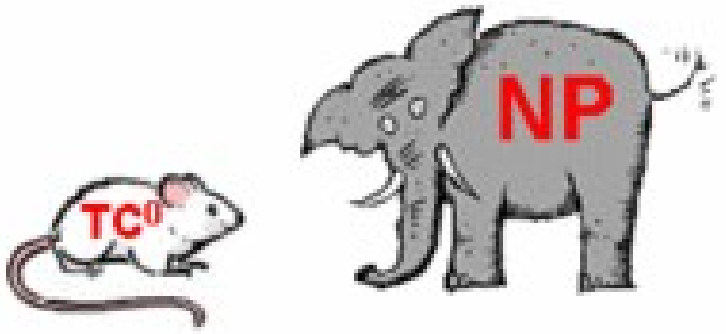
- What should we do for hard problems?
  -believe that there are some good algorithms for them?
  -believe that these problems are fundamentally different from easy problems?

Different Classes of Problems

- 人类认识世界的一个共同规则：物以类聚

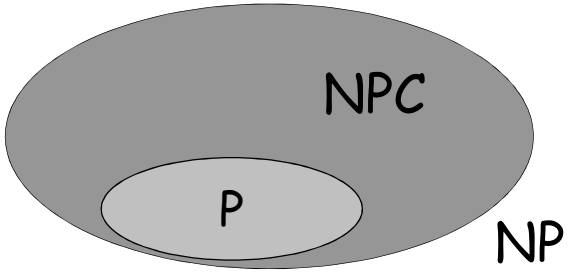
- Computational complexity.
  Classify the problems into different classes by measuring the minimum resource (running time, space, or others) required to solve the problem.

- Some important classes:
  P, NP, NPC, PTAS,...



Different Classes of Problems

- P: a solution can be solved in polynomial time.
- NP: a solution can be checked in polynomial time.
- NPC: problems that may not have polynomial-time algorithms.



P = NP?

植物是否等同动物？

- Many important problems in practice are NPC.

NPC Problems

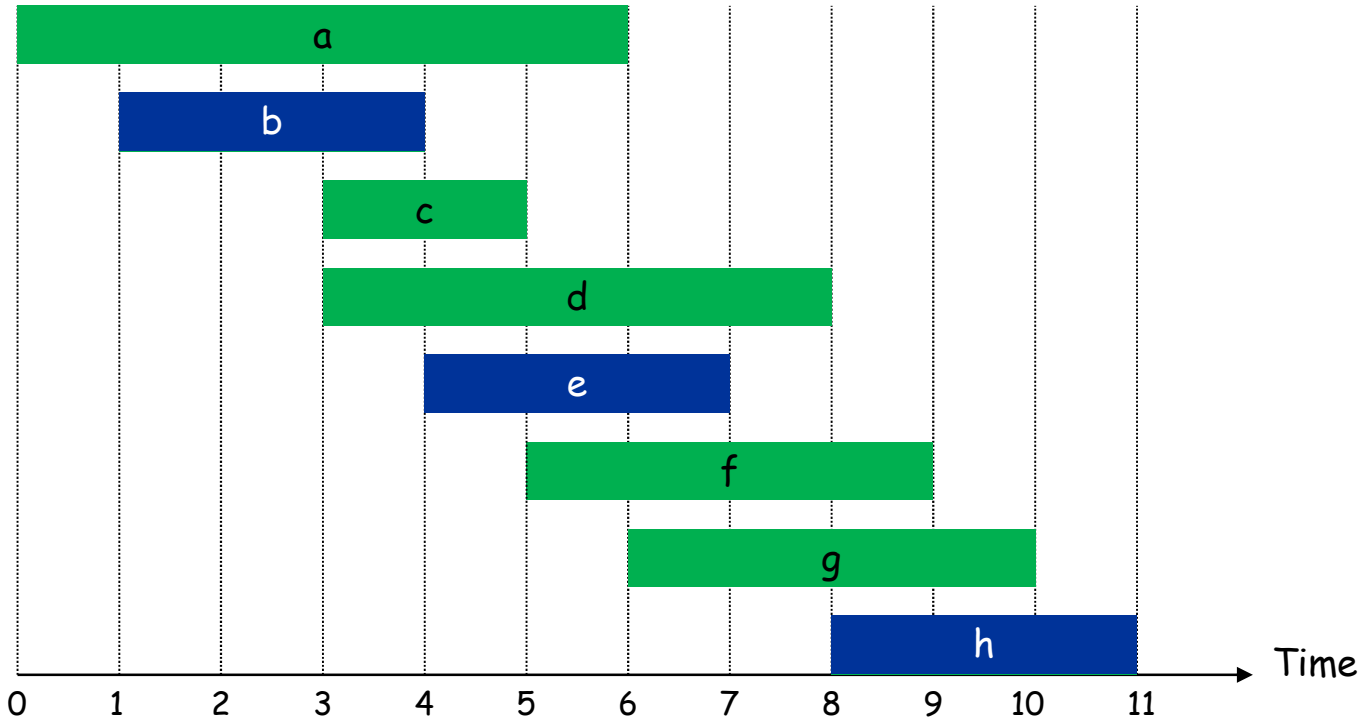
- Many methods to deal with NPC problems:

- Heuristic algorithms: solving the problems by your feeling. Donot know if it is right or not, but the algorithm runs fast.
- Approximation algorithms: the algorithm finds a solution not far away from the optimal solution and runs in polynomial time.
- Fast exact algorithms: effective exponential-time algorithms (must faster than simple search algorithms).
- Parameterized algorithms: the algorithm is effective when the parameter is small.

1.3 Some Representative Problems

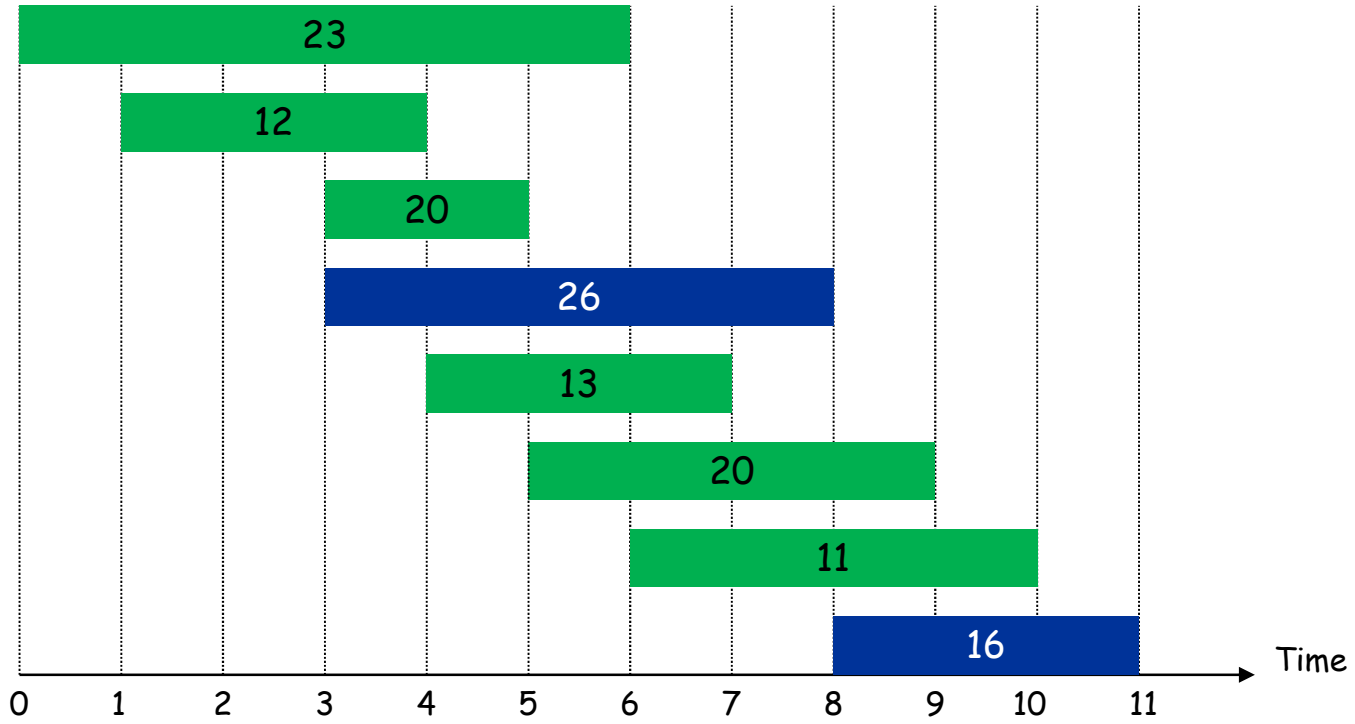
Interval Scheduling

Input. Set of jobs with start times and finish times.
Goal. Find maximum cardinality subset of mutually compatible jobs (donot overlap).



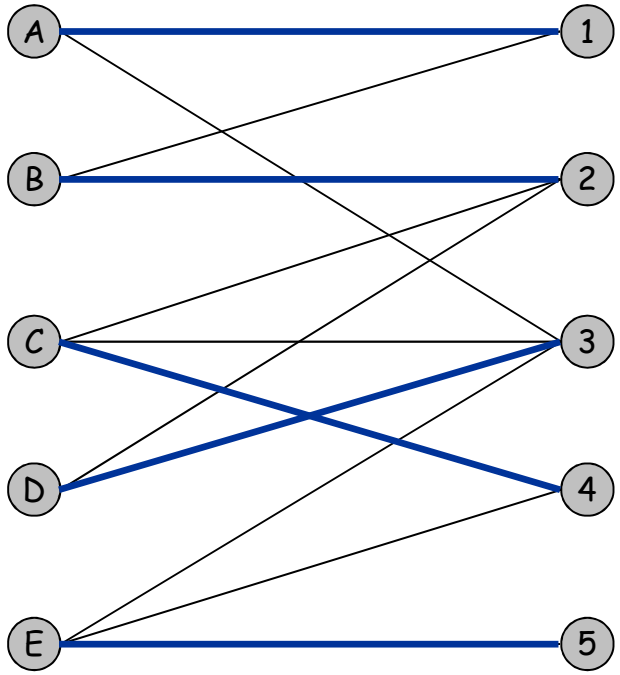
Weighted Interval Scheduling

Input. Set of jobs with start times, finish times, and weights.
Goal. Find maximum weight subset of mutually compatible jobs.



Bipartite Matching

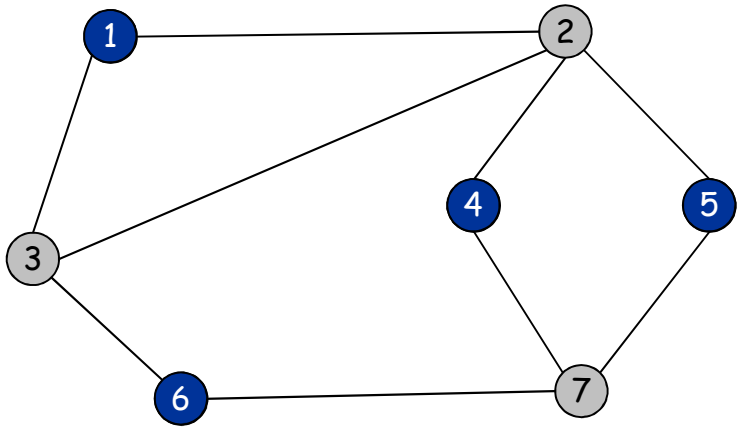
Input. Bipartite graph.
Goal. Find maximum cardinality matching.



Independent Set

Input. Graph.
Goal. Find maximum cardinality independent set.

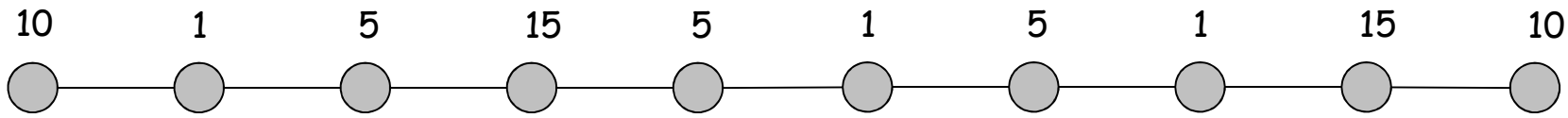
subset of nodes such that no two joined by an edge



Competitive Facility Location

Input. Graph with weight on each node.
Game. Two competing players alternate in selecting nodes. Not allowed to select a node if any of its neighbors have been selected.

Goal. Select a maximum weight subset of nodes.



Second player can guarantee 20, but not 25.

Representative Problems

Interval scheduling: n log n, greedy algorithm.

Weighted interval scheduling: n log n,
dynamic programming algorithm.

Weighted bipartite matching: n m,
Hungarian algorithm.

Independent set: NP-complete.

Competitive facility location: PSPACE-complete.