CS 401: Computer Algorithm I

Representative Problems / Running Time Analysis

Xiaorui Sun

Staff

- Website: http://www.cs.uic.edu/~xiaorui/cs401
  - Lecture slides, homework

- Piazza:https://piazza.com/uic/spring2024/cs40143452434534345743458
  - Announcements, online discussion forum
  - TA will answer course related questions

- Blackboard
  - Office hour, lecture video recordings, homework submission

- Office hours
- Myself: Tuesday 10am-12pm SEO 1241 and blackboard
- Chenye Zhao: Friday 1pm-3pm TBH180B and blackboard

Stable Matching Summary

- Q: What is a computational problem?
- A: Defined by input and output

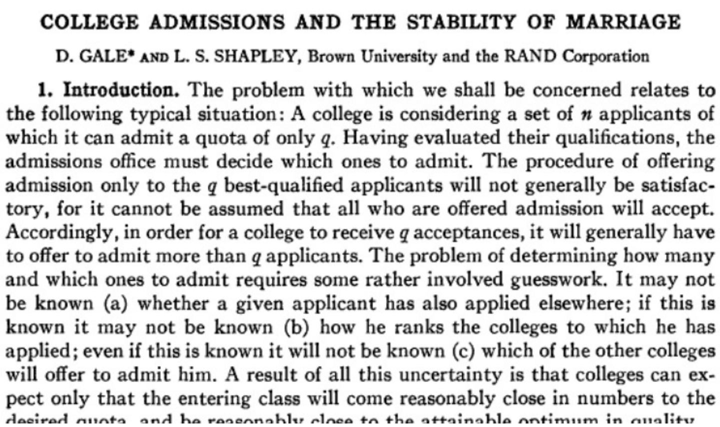
- Q: How to describe an algorithm?
- A: Describe what to do in each step (pseudocode)

- Q: Does an algorithm correctly solve a problem?
- A: Show the algorithm gives the correct solution on each input

- Q: How to implement an algorithm efficiently?
- A: Different implementations may have different running time

Why this problem is important?

In 1962, Gale and Shapley published the paper
"College Admissions and the Stability of Marriage"
To
"The American Mathematical Monthly"



COLLEGE ADMISSIONS AND THE STABILITY OF MARRIAGE

D. GALE* AND L. S. SHAPLEY, Brown University and the RAND Corporation

1. Introduction. The problem with which we shall be concerned relates to the following typical situation: A college is considering a set of n applicants of which it can admit a quota of only q. Having evaluated their qualifications, the admissions office must decide which ones to admit. The procedure of offering admission only to the q best-qualified applicants will not generally be satisfactory, for it cannot be assumed that all who are offered admission will accept. Accordingly, in order for a college to receive q acceptances, it will generally have to offer to admit more than q applicants. The problem of determining how many and which ones to admit requires some rather involved guesswork. It may not be known (a) whether a given applicant has also applied elsewhere; if this is known it may not be known (b) how he ranks the colleges to which he has applied; even if this is known it will not be known (c) which of the other colleges will offer to admit him. A result of all this uncertainty is that colleges can expect only that the entering class will come reasonably close in numbers to the desired quota, and be reasonably close to the attainable optimum in quality

David Gale (1921-2008)
PROFESSOR, UC BERKELEY

Lloyd Shapley
PROFESSOR EMERITUS, UCLA

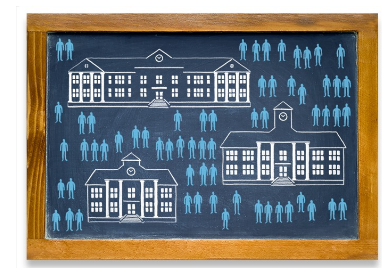
Why this problem is important?

Alvin Roth
PROFESSOR, STANFORD

Alvin Roth modified the Gale-Shapley algorithm and apply it to

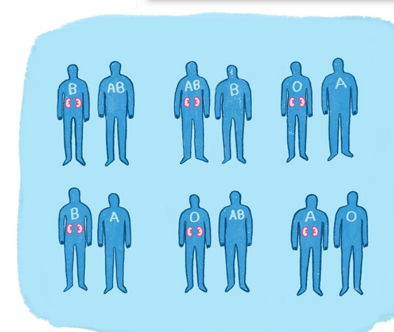
- National Residency Match Program (NRMP), a system that assigns new doctors to hospitals around the country. (90s)

- Public high school assignment process (00s)



- Helping transplant patients find a match (2004)

(Saved >1,000 people every year!)



Blood types: A, B, AB and O

Why this problem is important?

Some of the problems in this course may seem obscure or even pointless.

But their abstraction allows for variety of applications.

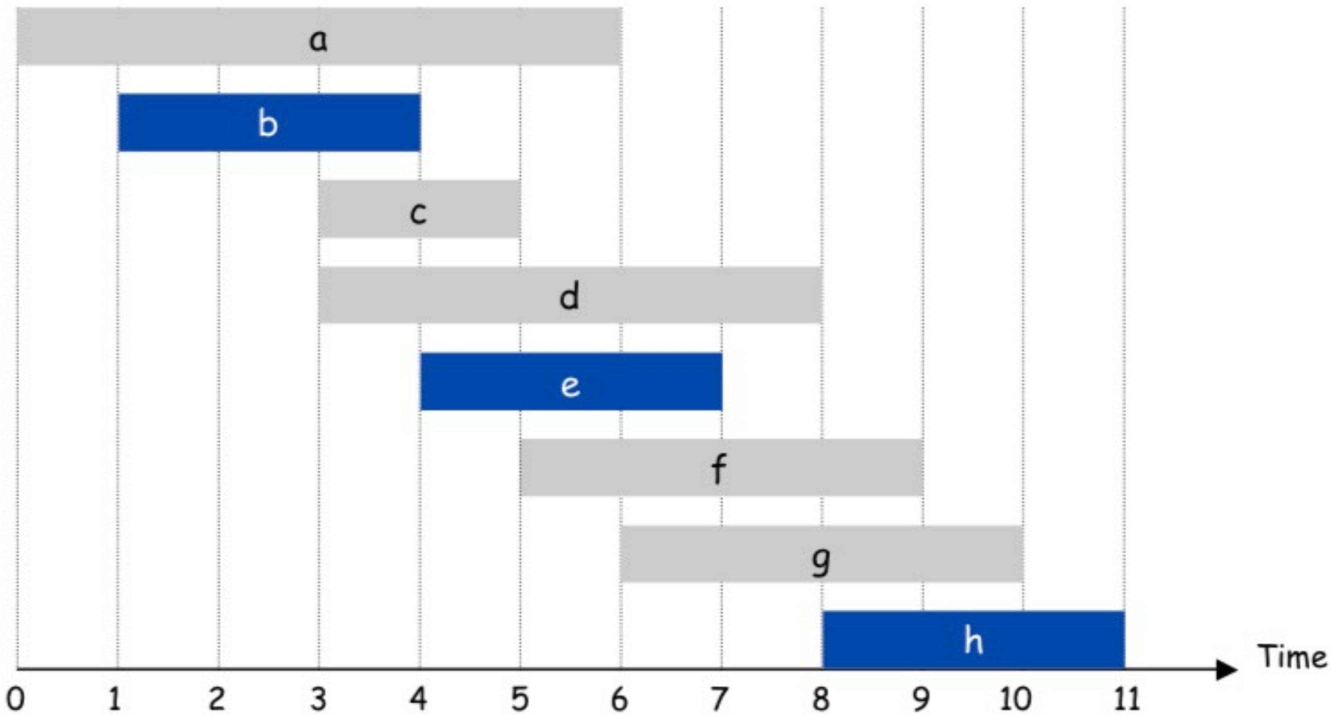
Shapley and Roth got the Nobel Prize (Economic) in 2012.
(David Gale passed away in 2008.)

Representative problems

Interval Scheduling

Input: Set of jobs with start times and finish times

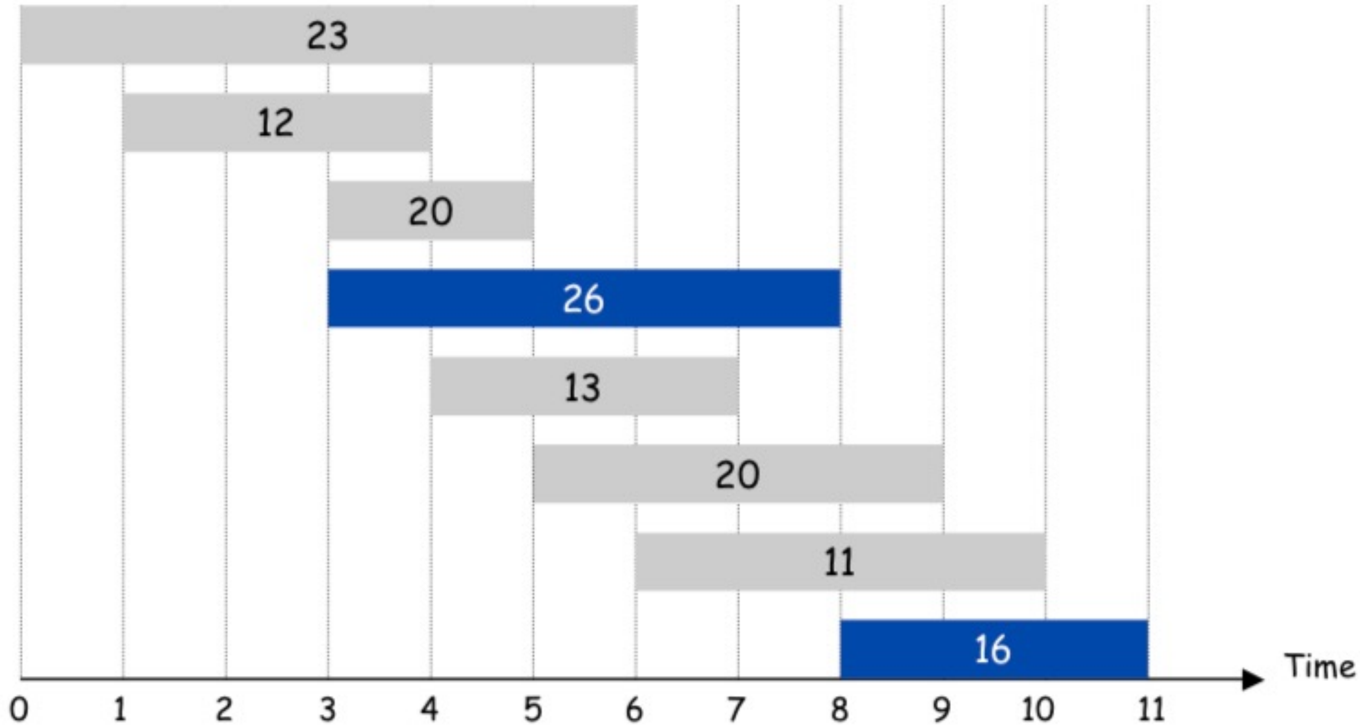
Goal: Find maximum cardinality subset of mutually compatible jobs ← Jobs don't overlap



Weighted Interval Scheduling

Input: Set of jobs with weights, start times and finish times

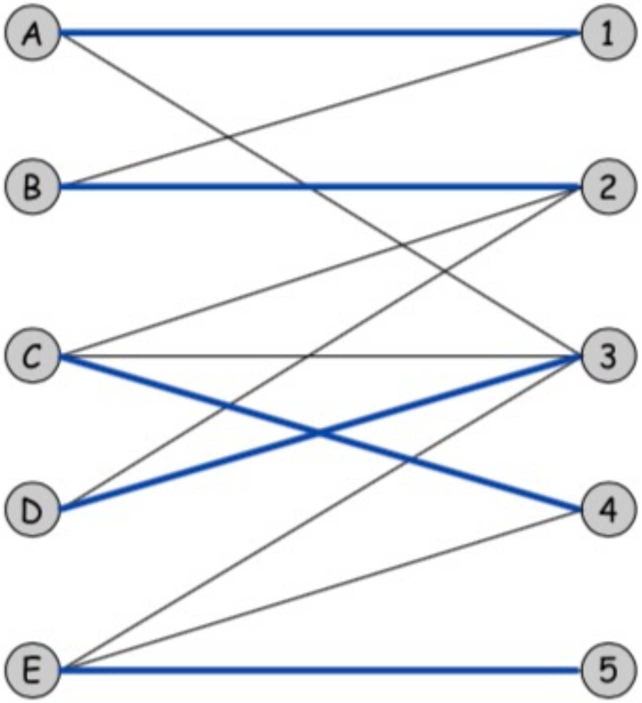
Goal: Find maximum weight subset of mutually compatible jobs



Bipartite Matching

Input: Bipartite graph

Goal: Find maximum cardinality matching

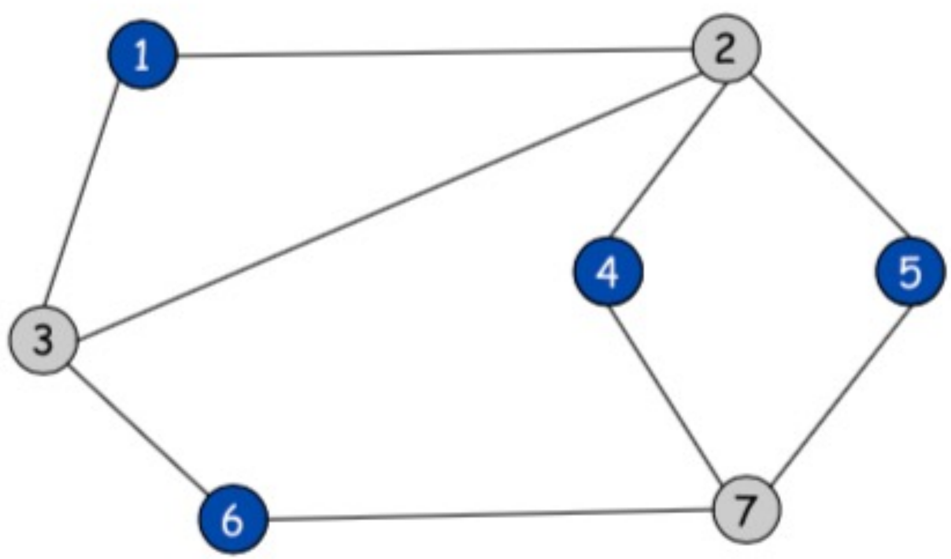


Independent Set

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 competitive 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

Five Representative Problems

Common theme: independent set

Interval scheduling: n log n greedy algorithm

Weighted interval scheduling: n log n dynamic programming algorithm

Bipartite matching: n^k max-flow based algorithm

Independent set: NP-complete

Competitive facility location: PSPACE-complete

Different properties make problems have different running times

Complexity

Defining Efficiency

“Runs fast on typical real problem instances”

Pros:

- Sensible

Cons:

- Moving target (diff computers, programming languages)
- Highly subjective (how fast is “fast”? What is “typical”?)

Measuring Efficiency

Time ≈ # of instructions executed in a simple programming language

- only simple operations (+,*,-,=,if,call,…)
- each operation takes one time step
- each memory access takes one time step
- no fancy stuff (add these two matrices, copy this long string,…) built in

Time Complexity

Problem: An algorithm can have different running time on different inputs

Solution: The complexity of an algorithm associates a number **T(N)**, the "time" the algorithm takes on problem size **N**.

On which inputs of size **N**?

Mathematically,

T is a function that maps positive integers giving problem size to positive integers giving number of simple operations

Time Complexity (N)

Worst Case Complexity: **max** # simple operations algorithm takes on any input of size **N**

This Course

Average Case Complexity: **avg** # simple operations algorithm takes on inputs of size **N**

Best Case Complexity: **min** # simple operations algorithm takes on any input of size **N**