# CMPSC 465
# Data Structures and Algorithms
# Spring 2022

Instructor: Chunhao Wang

# Greedy algorithms

# Finding optimal schedule using matroid

How to find an optimal schedule?

1. Optimizing over tasks in the canonical form:
   1.1 Find a set $A$ of tasks that are early
   1.2 Sort the tasks of $A$ in increasing deadlines
   1.3 Add late tasks in any order
2. Minimize penalties of late tasks $\equiv$ maximize penalties of early tasks

Modeled by a matroid $M = (S, \mathcal{I})$, where

$S = \{a_1, \ldots, a_n\}$

$\mathcal{I} = \{A \subseteq S : \exists \text{ a way to schedule the tasks in } A \text{ s.t. no task is late}\}$

$w$ : penalty

Finding an optimal schedule $\equiv$ finding max-weighted indep. subset of $M$

# Such $M$ for task scheduling is a matroid

$M = (S, \mathcal{I})$ is a matroid

- $\mathcal{I}$ has the hereditary property:
  if $A \subseteq B$ and $B \in \mathcal{I}$ then $A \in \mathcal{I}$
- Exchange property:
  Say $A, B \in \mathcal{I}$ and $|B| > |A|$.
  Assume $A$ and $B$ are sorted in increasing order of deadlines
  We need to show there exists an $x \in B - A$ s.t. $A \cup \{x\} \in \mathcal{I}$
  See Cormen et al. proof of Theorem 16.13

# Greedy algorithm for finding optimal scheduling

```
1 def GREEDY(M = (S, I), weights w):
2     Set A := { };
3     Sort S in decreasing order of w ;          // O(n log n)
4     for x ∈ S:
5         if A ∪ {x} ∈ I:
6             A := A ∪ {x};
7     return A;
```

**Running time:** let $n = |S|$

Assume checking if $A \cup \{x\} \in \mathcal{I}$ takes $O(f(n))$. Lines 5-6 takes $O(n \cdot f(n))$

**Claim:** $f(n) = O(n)$ for task scheduling problem (Homework)

Total running time: $O(n^2)$

# Greedy algorithms

## Horn formulas (Textbook Section 5.3)

# Horn formulas — Prelude

Consider the following puzzle

- If Alice has a dog, then Bob has a cat
- If Charlie and Bob both have pets of the same species, then Alice has a cat
- Charlie and Alice don't share a pet of the same species

Question: what pets do they have?

# Boolean formulas

Basics of boolean formulas

- **Variables:** possibilities
  Knowledge about variables is represented by a special type of boolean formulas
  Goal; find a *consistent* explanation of the knowledge
- **Boolean variable:** $x = 1$ (true) or $x = 0$ (false)
- **Literal:** $x$ (positive literal), $\bar{x}$ (negative literal)
- **Clause:** a clause consists of literals connected by
  $\wedge$ (AND), $\vee$ (OR), $\implies$ (implies)
  Examples: $x \wedge \bar{y}, \quad (x \wedge y) \implies z$

# Horn formulas

In a Horn formula, there are only two types of clauses (**Horn clauses**):

- **Implication:** $(x_1 \wedge x_2 \wedge \cdots \wedge x_n) \implies y$
  LHS: $\mathrm{AND}$ of any number of positive literals
  RHS: single positive literal
  - $(x \wedge \bar{y}) \implies z$ ✗
  - $(x \vee y) \implies z$ ✗
  - $\implies z$ ✓
- **Pure negative clauses** $\bar{x}_1 \vee \bar{x}_2 \vee \cdots \vee \bar{x}_n$
  $\mathrm{OR}$ of any number of negative literals

# Horn formula example

Consider the puzzle:

- If Alice has a dog, then Bob has a cat
- If Charlie and Bob both have pets of the same species, then Alice has a cat
- Charlie and Alice don't share a pet of the same species

Define variables:

- $a$: Alice has a dog
- $b$: Bob has a dog
- $c$: Charlie has a dog
- $x$: Alice has a cat
- $y$: Bob has a cat
- $z$: Charlie has a cat

Modelled by a set of Horn clauses:

$a \implies y$

$(b \wedge c) \implies x$

$(y \wedge z) \implies x$

$\bar{a} \vee \bar{c}$

$\bar{x} \vee \bar{z}$

Question: satisfying assignment?

# Greedy approach for Horn formulas

**Problem (Horn Satisfiability)**

*Given a set of Horn clauses, determine whether or not there is a consistent explanation, i.e., an assignment of 0/1 to variables that satisfy all clauses*

Example: $(x \wedge y) \implies z$, $\bar{x} \vee \bar{w}$ can be satisfied by $x = 0, y = 0, z = 0, w = 0$

**Greedy heuristic:** start with all 0. Only set a variable to 1 if you need to, i.e., when an implication says you need to

Recall: $p \implies q \iff \bar{p} \vee q$

# Pseudocode

**def** GREEDY_HORN(*set of Horn clauses*)**:**

> Set all variables to 0;
>
> **while** *there exists an "$\implies$" that is not satisfied***:**
>
> > Set its RHS to 1;
>
> **if** *all pure negative clauses are 1***:**
>
> > **return** the assignment;
>
> **else:**
>
> > **return** "unsatisfiable";

Example: $\implies x$, $x \implies y$, $(\bar{x} \vee \bar{y})$

| $x$ | $y$ | |
|---|---|---|
| 0 | 0 | $\implies x$ ✗ |
| 1 | 0 | $x \implies y$ ✗ |
| 1 | 1 | $\implies x$ ✓, $x \implies y$ ✓, $(\bar{x} \vee \bar{y})$ ✗ |

**Unsatisfiable**

## Correctness and running time

**Correctness:** If GREEDY_HORN finds an assignment, then the problem has a satisfying assignment

If it returns "unsatisfiable", is it really unsatisfiable?

**Theorem**

*The variables set to 1 by* GREEDY_HORN *must be 1 in* ***any*** *satisfying assignment*

**Exercise:** Prove this by induction

How does this theorem help?

If all the pure negative clauses cannot be satisfied after the while loop, then there's no such assignment satisfying them

**Running time:** Let $n$ be the size of the Horn formula, i.e., the number occurrences of literals.

Total running time: $O(n^2)$. Can be improved to $O(n)$ (**exercise**)