Design and Analysis of Algorithms

Textbooks

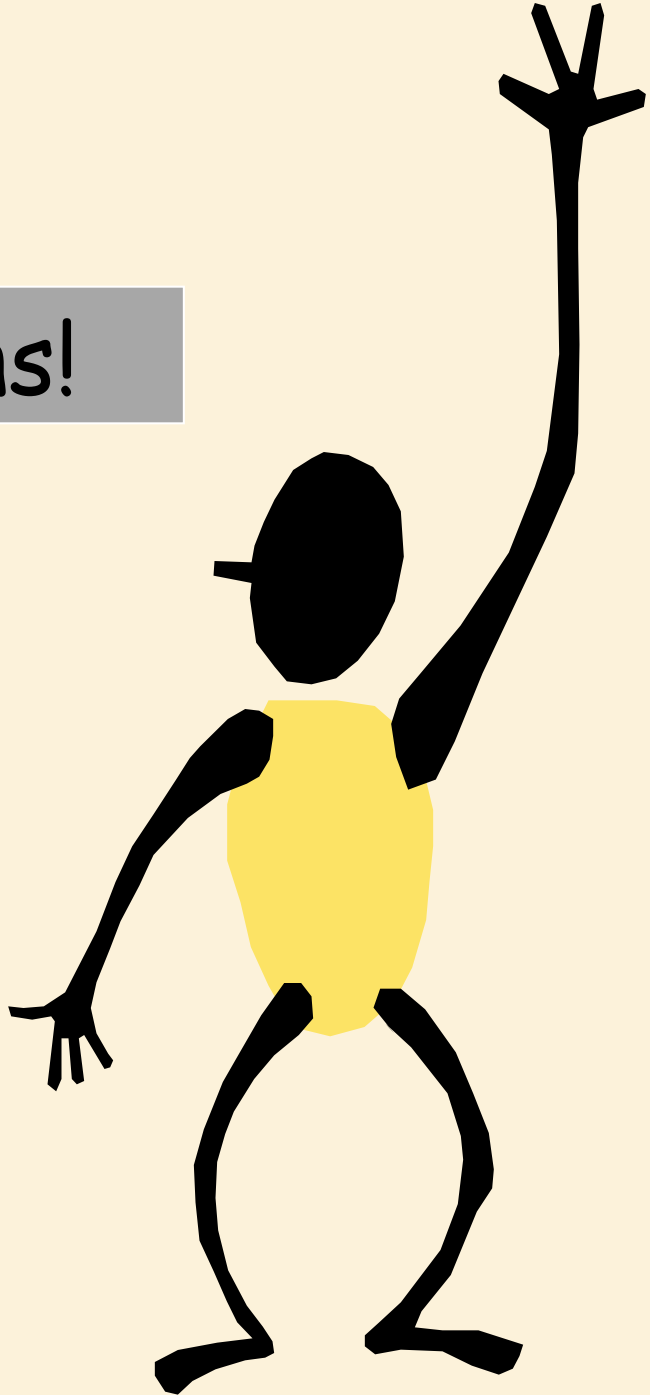
- Required Text:
  - Cormen, T.H., Leiserson, C.E., Rivest, R.L. & Stein, C. (2009). *Introduction to Algorithms*, 3rd Ed. Cambridge, Mass: MIT Press.

Assignments

- You will learn best if you try to tackle each problem by yourself initially.
- You are encouraged to discuss the problems and work in groups to overcome roadblocks.
- You are encouraged to team up with a partner and write up a single assignment report (maximum 2 per group).
- Make the reports as concise and organized as possible. Marks may be taken off for excess verbosity or lack of clarity.
- Late assignments are not excepted (except for medical emergencies – please see syllabus).

Please ask questions!

Help me know what people are not understanding!



Lecture 1. What is this course about?

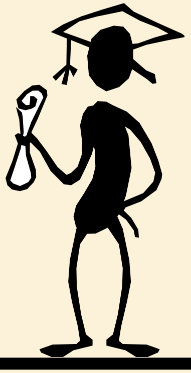
Course Content

- ~~A list of algorithms.~~
  - ~~Learn their code.~~
  - ~~Trace them until you are convinced that they work.~~
  - ~~Implement them.~~
  - ~~Worry about details.~~

```
 class InsertionSortAlgorithm extends SortAlgorithm
{
  void sort(int a[]) throws Exception {
        for (int i = 1; i < a.length; i++) {
          int j = i;
          int B = a[i];
          while ((j > 0) && (a[j-1] > B)) {
            a[j] = a[j-1];
            j--;  }
          a[j] = B;
      }}
```

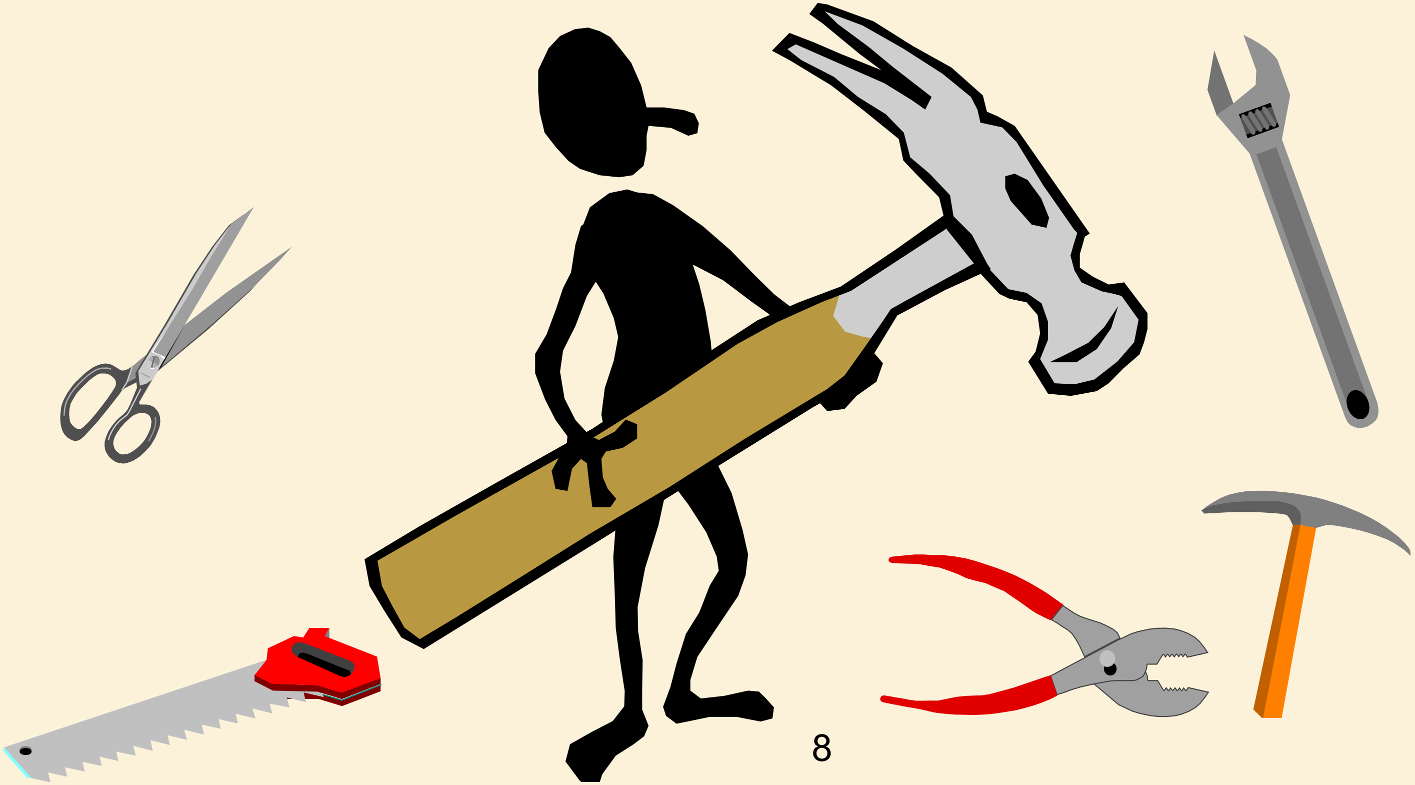
The future belongs to the computer scientist/engineer who has

- Knowledge: An up to date grasp of fundamental problems and solutions
- Ability: Principles and techniques that can be adapted to solve new problems



Course Content

- A survey of algorithmic design techniques.
- Abstract thinking.
- How to develop new algorithms for any problem that may arise.



A survey of fundamental ideas and algorithmic design techniques

For example . . .

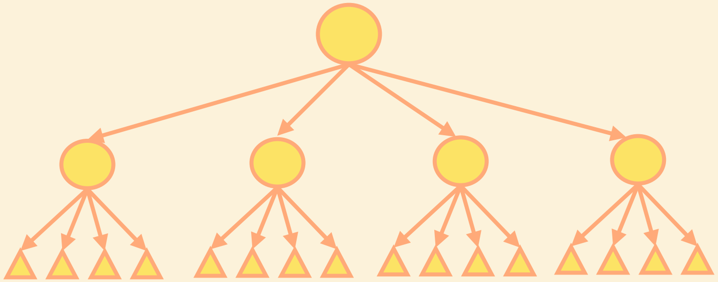
Mathematical Tools

Summations

$\sum_{i=1} f(i).$

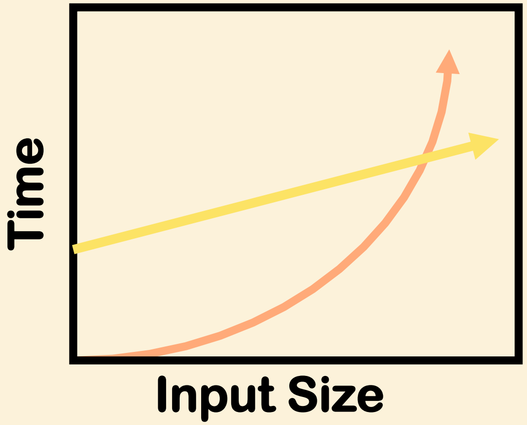
Recurrence Relations

$T(n) = a\,T(n/b) + f(n)$



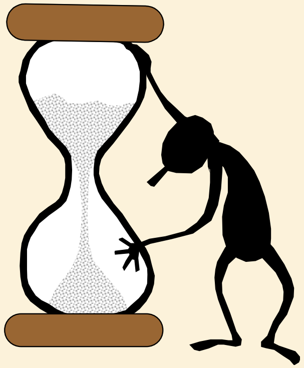
Classifying Functions

$f(i) = n^{\Theta(n)}$



Time Complexity

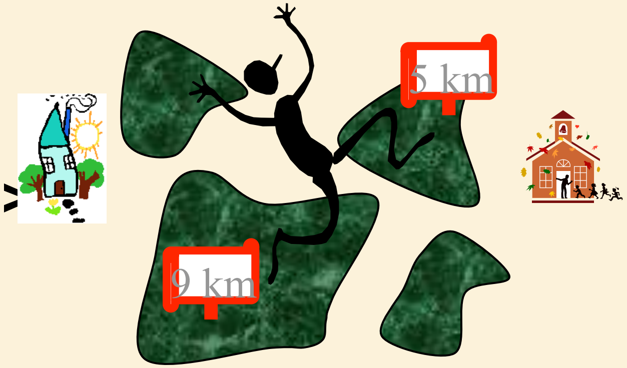
$t(n) = \Theta(n^2)$



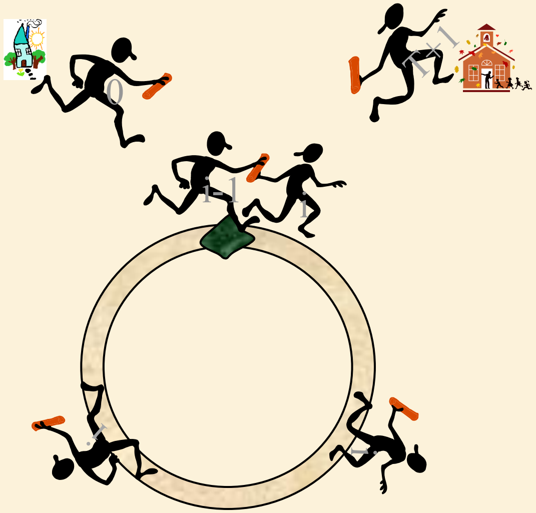
Iterative Algorithms
Loop Invariants

```
<preCond>
codeA
loop
    <loop-invariant>
    exit when <exit Cond>
    codeB
codeC
<postCond>
```

Code

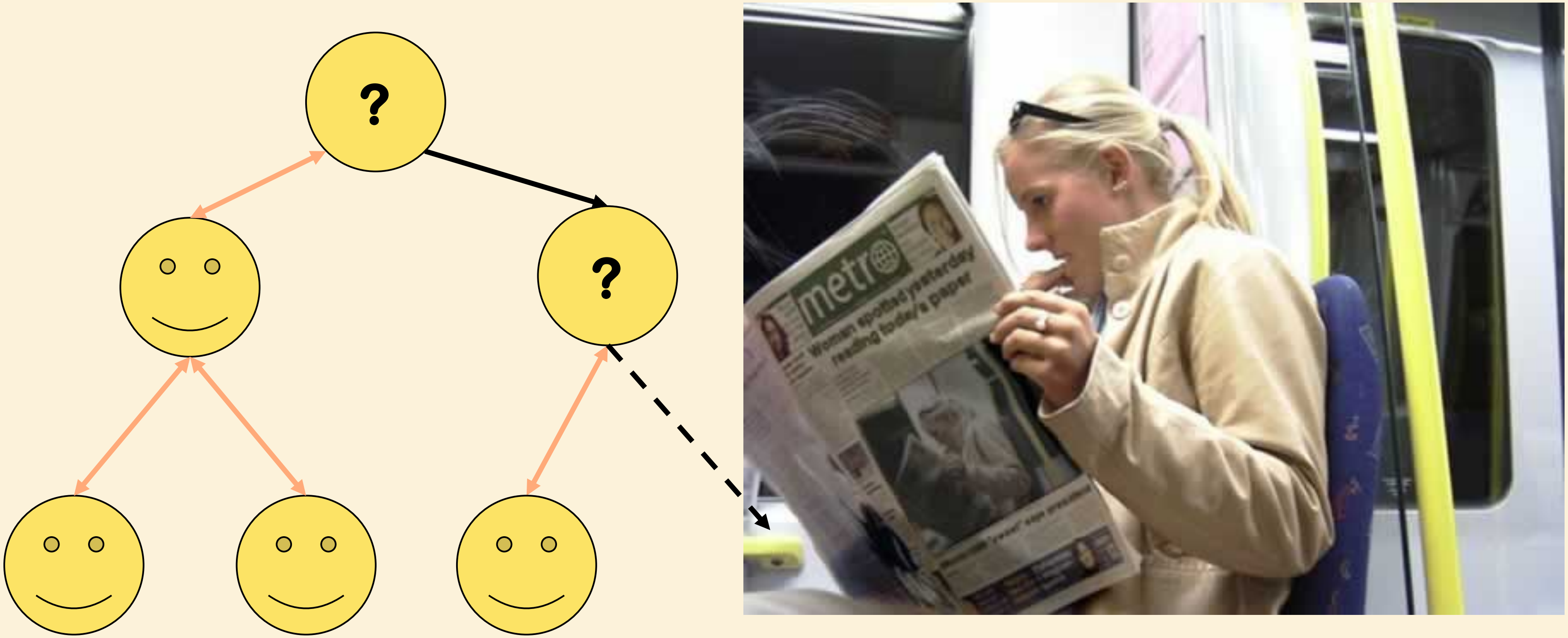


One step
at a time

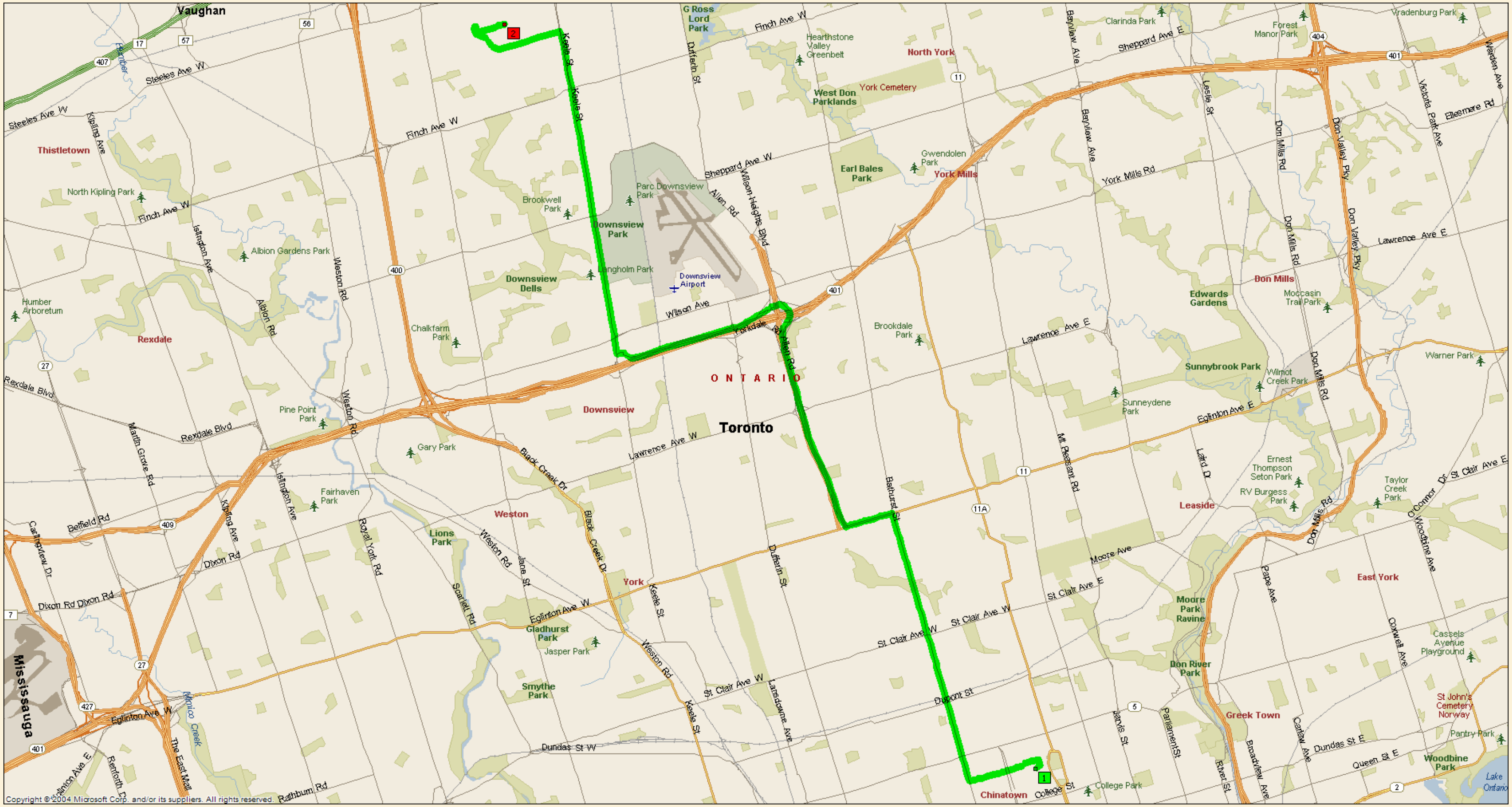


Relay Race

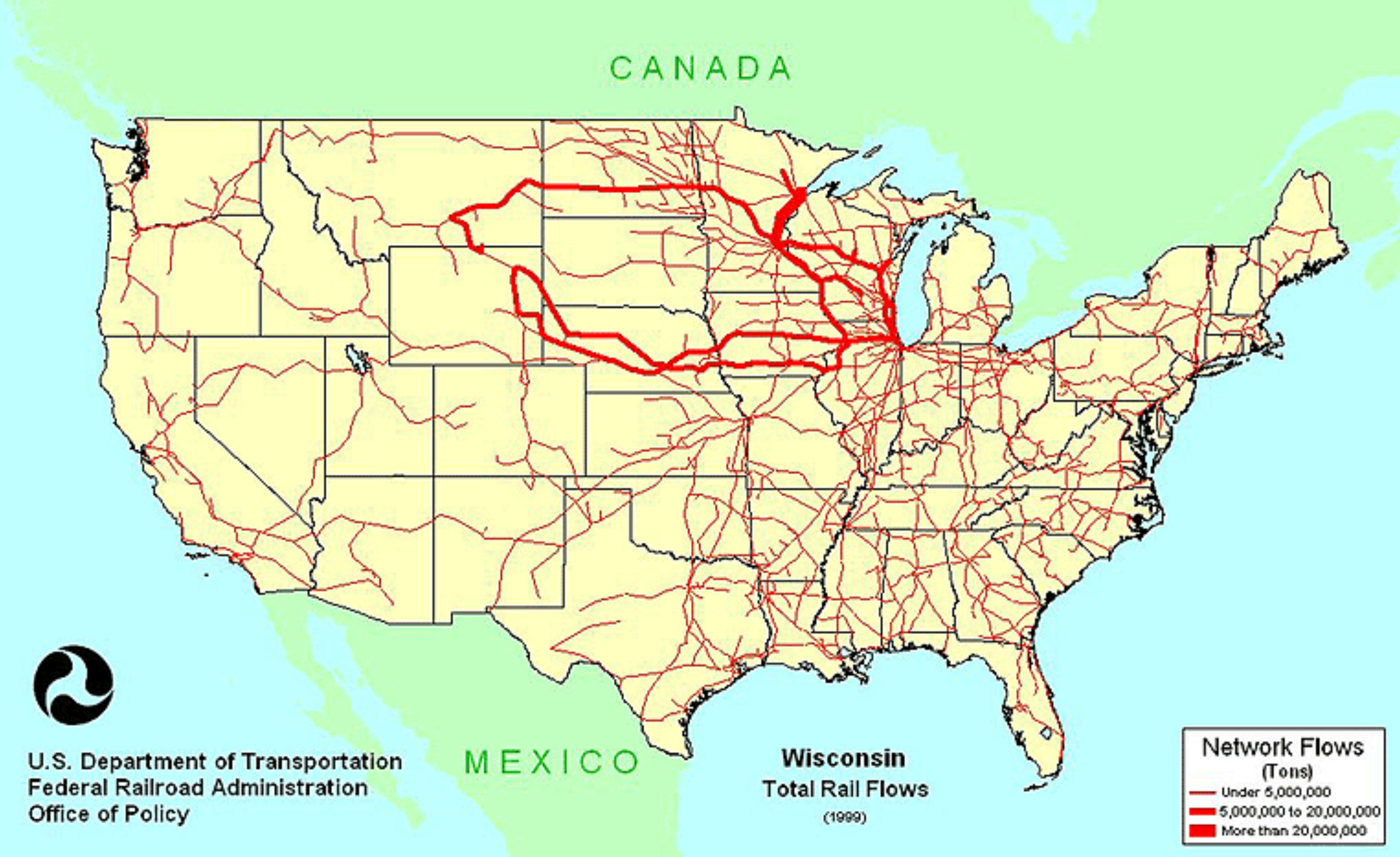
Recursive Algorithms



Graph Search Algorithms



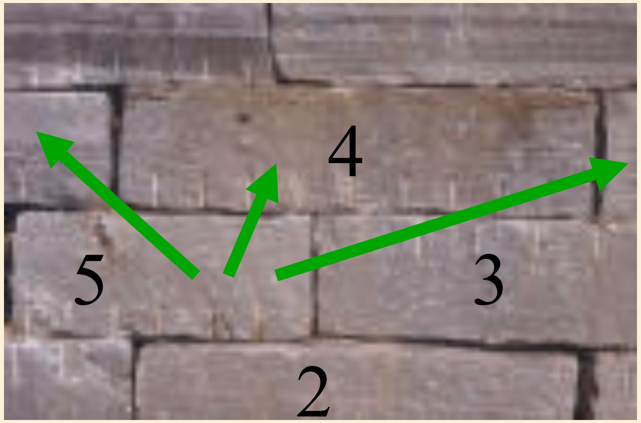
Network Flows



Greedy Algorithms

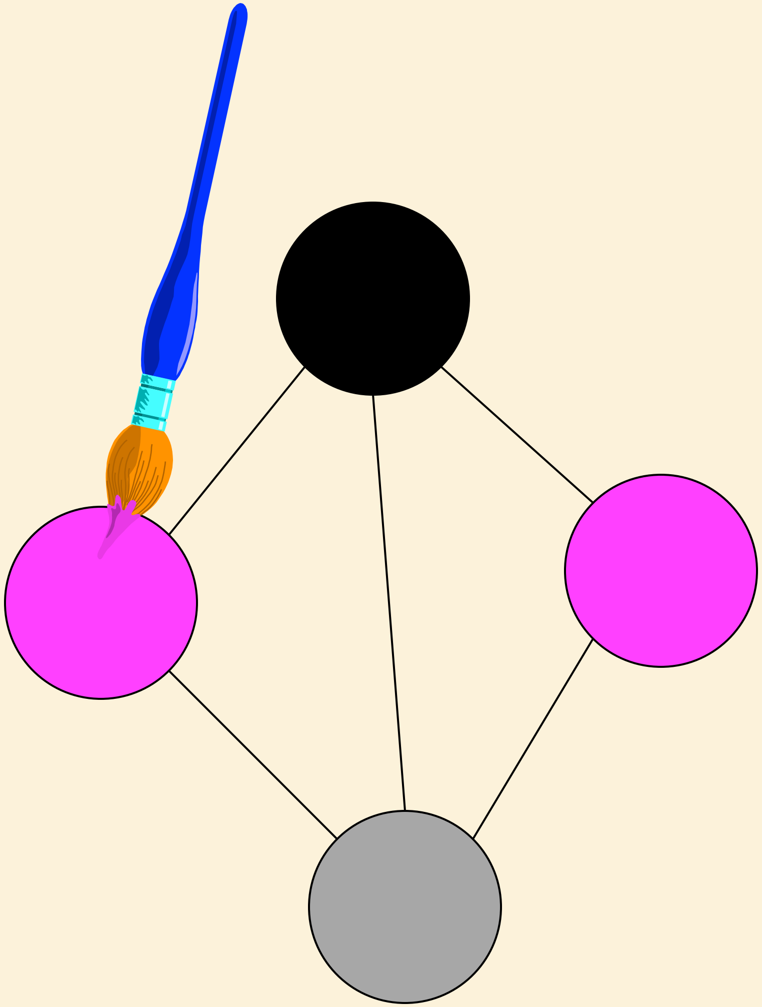
Example: Making Change

Dynamic Programing

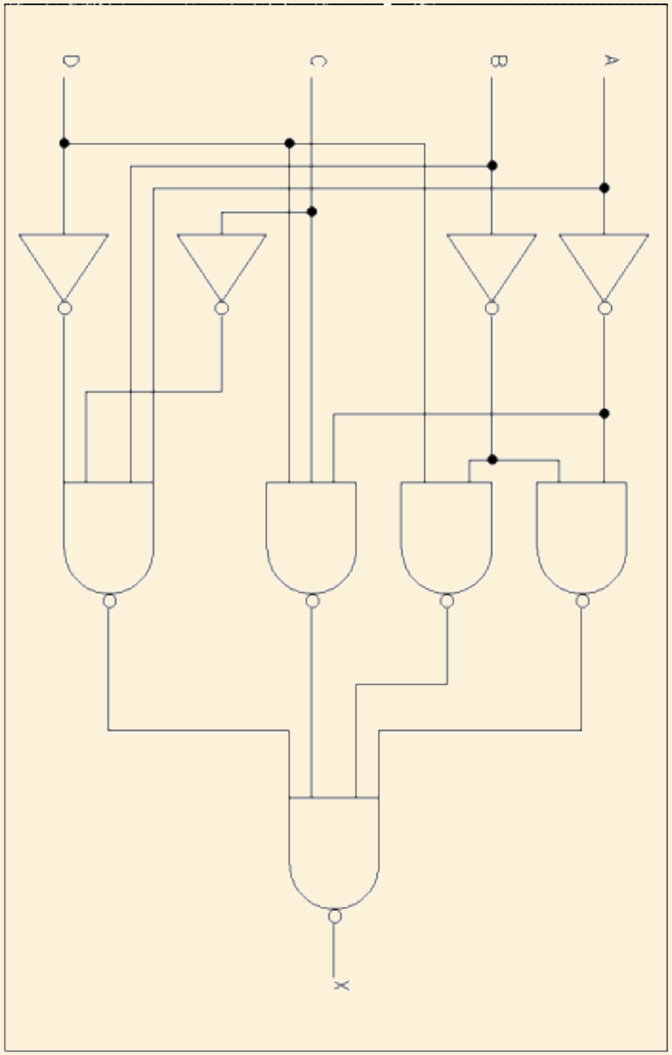


2	8	9	5	8
4	4	6	2	3
5	7	5	6	1
3	2	5	4	8

Reduction



=



NP-Completeness

Useful Learning Techniques

Read Ahead

You are expected to read the lecture notes **before** the lecture.

This will facilitate more productive discussion during class.

Explaining

- We are going to test you on your ability to explain the material.
- One good way to study is to explain the material over and over again to yourself or to each other.

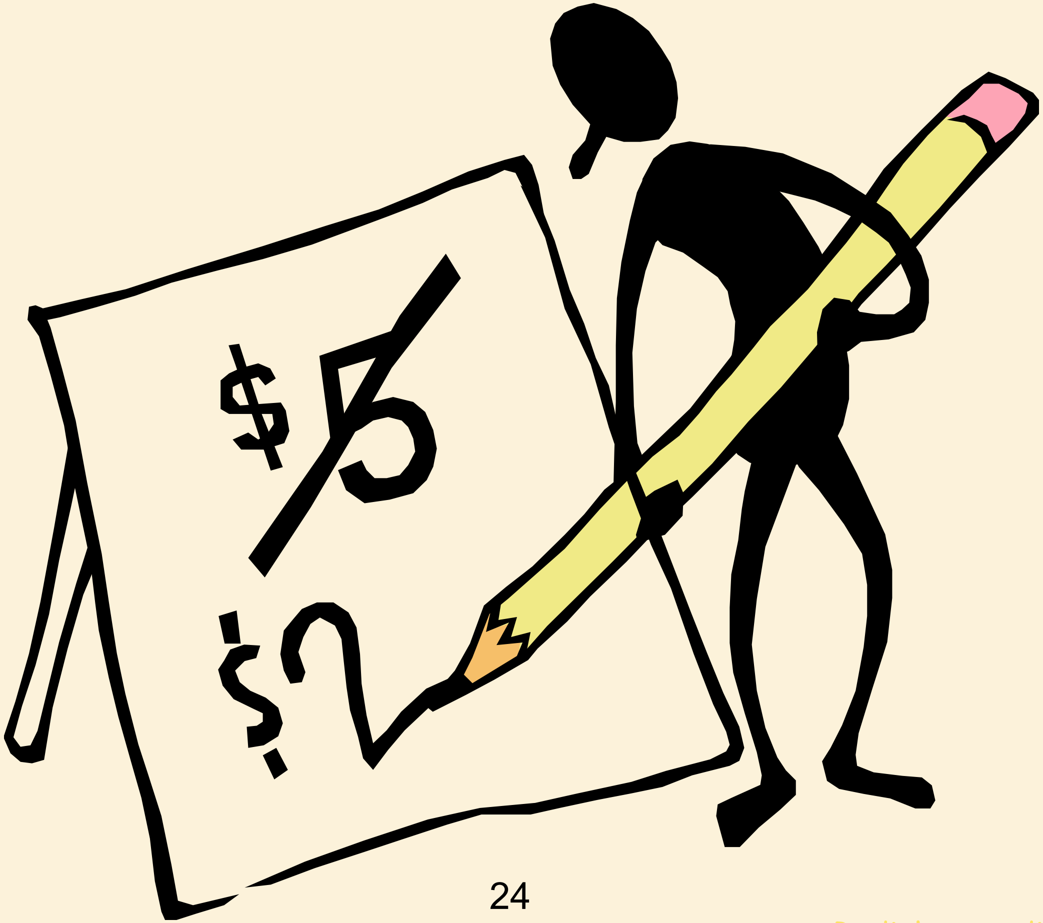
Be Creative

- Ask questions.
- Why is it done this way and not that way?

Guesses and Counter Examples

- Guess at potential algorithms for solving a problem.
- Look for input instances for which your algorithm gives the wrong answer.
- Treat it as a game between these two players.

Refinement:
The best solution comes from a process of repeatedly refining and inventing alternative solutions



End