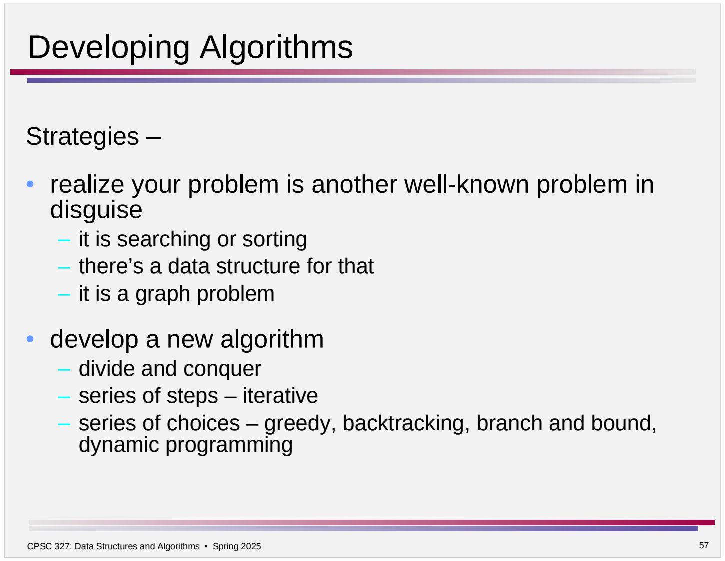
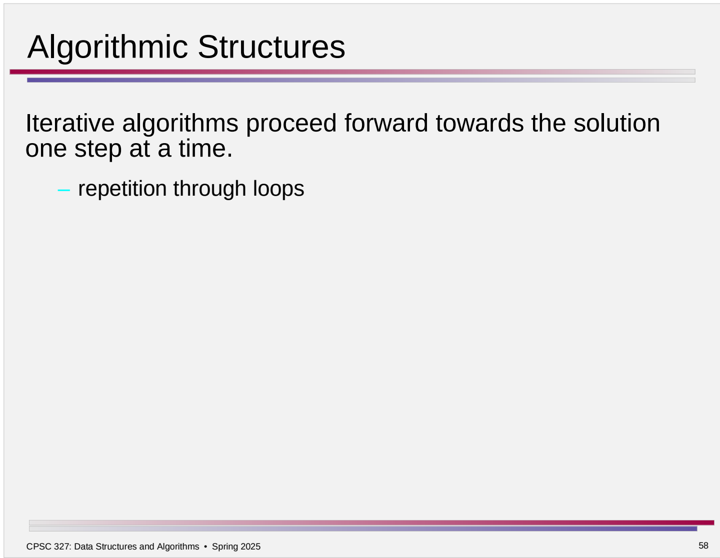


Developing Algorithms

Strategies –

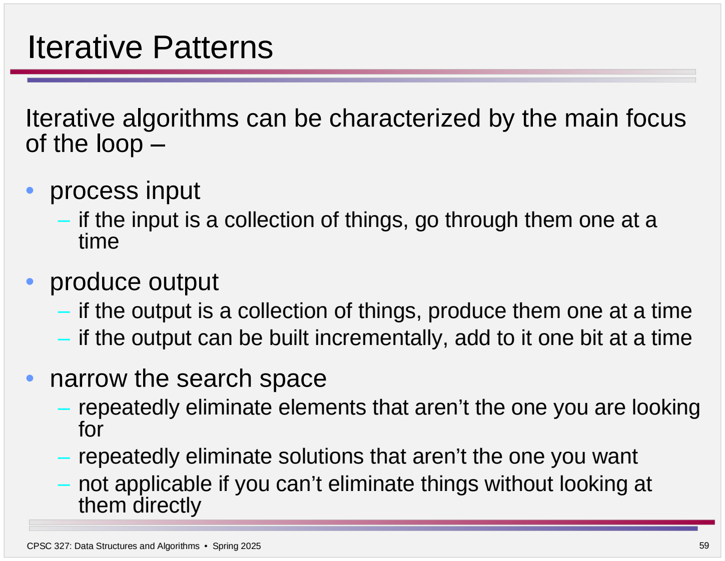
- realize your problem is another well-known problem in disguise
  - it is searching or sorting
  - there’s a data structure for that
  - it is a graph problem
- develop a new algorithm
  - divide and conquer
  - series of steps – iterative
  - series of choices – greedy, backtracking, branch and bound, dynamic programming



Algorithmic Structures

Iterative algorithms proceed forward towards the solution one step at a time.

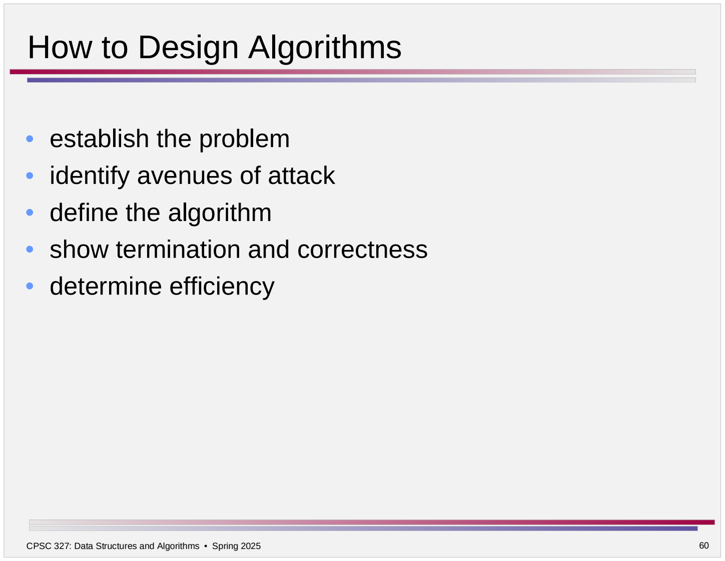
  - repetition through loops



Iterative Patterns

Iterative algorithms can be characterized by the main focus of the loop –

- process input
  - if the input is a collection of things, go through them one at a time
- produce output
  - if the output is a collection of things, produce them one at a time
  - if the output can be built incrementally, add to it one bit at a time
- narrow the search space
  - repeatedly eliminate elements that aren’t the one you are looking for
  - repeatedly eliminate solutions that aren’t the one you want
  - not applicable if you can’t eliminate things without looking at them directly



How to Design Algorithms

- establish the problem
- identify avenues of attack
- define the algorithm
- show termination and correctness
- determine efficiency

How to Design (Iterative) Algorithms

Identify avenues of attack.

- *Paradigms and patterns.*
  Consider the iterative patterns defined in section 3.1.

pattern	loop structure
process input	for each input element, process that element and incorporate it into the solution so far
produce output	repeatedly produce the next output element repeatedly produce the next piece of the solution
narrow the search space	repeatedly eliminate some non-solutions

How to Design (Iterative) Algorithms

Define the algorithm. The core of an iterative algorithm is defining the loop.

- *Main steps.*
  This is the core of the algorithm — the loop body. What's being repeated?
- *Exit condition.*
  When does the loop end?

pattern	loop structure	exit condition
process input	for each input element, process that element and incorporate it into the solution so far	when all of the input elements have been processed
produce output	repeatedly produce the next output element repeatedly produce the next piece of the solution	when all of the output elements have been produced when the solution is complete
narrow the search space	repeatedly eliminate some non-solutions	when the solution has found or there are no solutions left

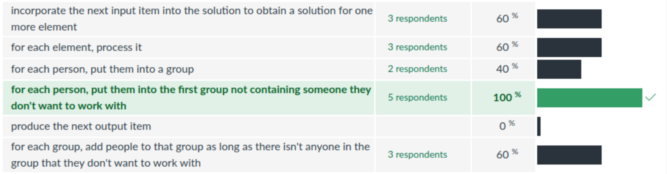
- *Setup.*
  Whatever must happen before the loop begins.
- *Wrapup.*
  Whatever must happen to get the final answer after the loop ends.
- *Special cases.*
  Make sure the algorithm works for all legal inputs — identify the cases that need to be handled and address how that handling is incorporated into the previous steps (if not already accounted for).
- *Algorithm.*
  Assemble the algorithm from the previous steps and state it.

  There shouldn't be new elements here, instead bring together the main steps, exit condition, setup, and wrapup along with any handling needed for special cases and state the whole algorithm.

In a group of people, it is to be expected that some of them may not want to work with each other. Assuming that each person has at most *d* other people that they don't want to work with, divide the people into *d*+1 groups so that everyone is in exactly one group and no one is in a group with someone they don't want to work with.

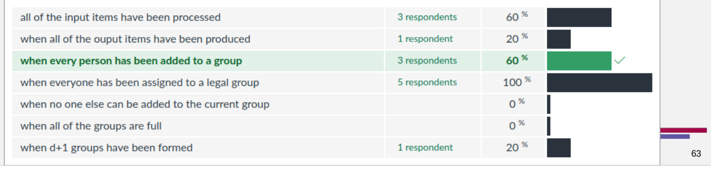
pattern	loop structure	exit condition
process input	for each input element, process that element and incorporate it into the solution so far	when all of the input elements have been processed

Main steps.



incorporate the next input item into the solution to obtain a solution for one more element	3 respondents	60 %	
for each element, process it	3 respondents	60 %	
for each person, put them into a group	2 respondents	40 %	
for each person, put them into the first group not containing someone they don't want to work with	5 respondents	**100 %**	✓
produce the next output item		0 %	
for each group, add people to that group as long as there isn't anyone in the group that they don't want to work with	3 respondents	60 %	

Exit condition. The loop ends when --



all of the input items have been processed	3 respondents	60 %	
when all of the ouput items have been produced	1 respondent	20 %	
when every person has been added to a group	3 respondents	**60 %**	✓
when everyone has been assigned to a legal group	5 respondents	100 %	
when no one else can be added to the current group		0 %	
when all of the groups are full		0 %	
when d+1 groups have been formed	1 respondent	20 %	