A Series of Choices

- divide-and-conquer works by dividing the task into independent subproblems which are solved separately
- an alternative is to build up a solution incrementally by making a series of choices

Given a collection of events with start time s(i) and finish time f(i) $(0 \le s(i) \le f(i))$, find the largest set of non-overlapping events.

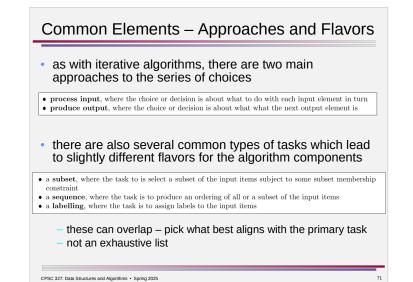
initialize an empty set of events repeatedly select a non-overlapping event until no non-overlapping events remain

Paradigms

CPSC 327: Data Structures and Algorithms . Spring 2025

CPSC 327: Data Structures and Algorithms . Spring 2025

- how many alternatives need to be considered for each decision leads to fundamentally different algorithmic paradigms
- If only one alternative needs to be considered, the formulation can be iterative. The key focus for the algorithm is determining how to pick that right alternative for each decision, and showing that the series of choices made leads to a correct solution. Greedy algorithms are of this type and will be considered in chapter 7.
- If more than one alternative needs to be considered, the formulation is typically recursive. The key
 focus for the algorithm is how to avoid an exponential blowup in running time. Backtracking, branchand-bound, and dynamic programming algorithms are of this type and will be considered in chapters 8
 to 10.



Greedy Algorithms

- iterative
- always make a local decision
 - each choice is made without consideration of future possibilities
- often, but not exclusively, applied to optimization problems
 - goal is to find the best solution among (generally) many legal solutions
 for non-optimization problems, goal is to find a legal solution among (generally) many invalid (non-)solutions
- don't work for everything requires
 - greedy choice property that a globally optimal/legal solution can be found by making local choices
 - optimal substructure property that an optimal/legal solution can be constructed from optimal/legal solutions of subproblems
- a correctness proof is essential!
 - finding counterexamples is an important technique for identifying incorrect greedy choices

CPSC 327: Data Structures and Algorithms • Spring 2025

Proof Techniques – Incorrectness

One counterexample is all that is needed to prove an algorithm incorrect.

Properties of a good counterexample.

- simple, which often means small
- verifiable need to be able to compute the algorithm's output and give a better answer

Strategies.

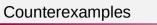
- think exhaustively can often enumerate all possible inputs of a small size
- hunt for weakness look for a case where the algorithm's choice is the wrong thing to do

'4

76

- try inputs with duplicates or ties, as that neutralizes the algorithm's choice
- seek extremes rather than uniformity

Counterexamples					
1-2. [3] Sl	[3] Show that $a \times b$ can be less than $\min(a, b)$.				
	[5] Design/draw a road network with two points a and b such that the fastest route between a and b is not the shortest route.				
			k with two points a and b ne route with the fewest to		
and a ample but n Find	a targe e, ther ot T = counte	t number T , find a e exists a subset wi = 23. rexamples to each o	ollows: given a set of integ subset of S that adds up thin $S = \{1, 2, 5, 9, 10\}$ th f the following algorithms where the algorithm does a) exactly to T. For exhat adds up to $T = 22$ for the subset sum problem	
	1 11070 1	s, grie un o und i '	, even though a	solution exists.	
(a)	Pick	elements of S	in left to right	nt order if they fit,	
(b)	Pick	elements of S	from smalles	t to largest, that is,	
(c)	Pick	elements of S	from largest	to smallest.	
PSC 327: Data Struc	tures and Alq	gorithms • Spring 2025			



Find a counterexample to prove the following statement false:

a + b >= min(a,b)

both numbers negative e.g. -5, -2
 5, -2
 5, -2
 5, -2
 5, -2

--5 + -2 = -7 ≥ min(-5, -2) = -5 → false

How to Design Greedy Algorithms

establish the problem

CPSC 327: Data Structures and Algorithms . Spring 2025

for optimization problems, identify "legal solution" separate from "optimal solution"

- identify avenues of attack
 - patterns iterative patterns + series-of-choices interpretation
 what the decision is about next input item (process input) or next output item (produce output)
 - flavors
 - type of decision (picking a subset, ordering, labeling, $\ldots)$
 - greedy choice by what criteria can we pick an alternative?
 - counterexamples rule out incorrect greedy choices
- define the algorithm
 - iterative algorithm steps
- show termination and correctness
 - loop invariant patterns
 - for optimization problems staying ahead
 - in general we haven't gone wrong yet
 - commonly use proof by contradiction for the "maintain the invariant" step
- determine efficiency

Proof Techniques – Contradiction

- assume that what you want to prove is false
- develop logical consequences from this assumption, until you get to one that is demonstrably false
- since there were no flaws in the deduction, the assumption that what you want to prove is false must have been faulty and thus what you want to prove is true

CPSC 327: Data Structures and Algorithms • Spring 2025