Greedy Algorithms

CPSC 320 2023W2

Optimization problems

- The next few worksheets will cover **optimization problems**:
  - We have a problem with several valid solutions
  - There is an **objective function** that tells us how good (or bad) each solution is
  - We are looking for the valid solution that minimizes or maximizes the value of the objective function
- We will look at two algorithm design paradigms:
  - Greedy algorithms
  - Dynamic programming

Optimization problems

Examples:

- Given a set of intervals, find the largest set of intervals that don’t overlap.
  - The objective function is the cardinality of the set
- Given a set of jobs with deadlines, order the jobs so as to minimize their total lateness
  - Objective function is the total lateness
- Given a weighted connected graph, find a spanning tree with the smallest total edge weight
  - Objective function is the sum of the weights of the edges in the spanning tree

Defining greedy algorithms

- A greedy algorithm proceeds by:
  - Making a choice based on a simple, local criterion
  - Solving the subproblem that results from that choice
  - Combining the choice and the subproblem choice
- We can think of a greedy algorithm as making a **sequence of** (locally “good”) **choices.**
- There is no precise definition of “greedy.”

Defining greedy algorithms

Examples of choice:

- Choosing the interval with the earliest finish time.
- Choosing the job with the earliest deadline.
- Choosing the item needed further in the future.
- Choosing the smallest weight edge that does not create a cycle.

Defining greedy algorithms

Does a greedy algorithm always give the correct solution?

- Sometimes yes, sometimes no.
- There are some classes of problems (e.g., matroids) for which there exists a greedy algorithm that always returns the correct solution.
- There are other problems where no one knows any greedy algorithm with this property (e.g., weighted interval scheduling).
- There are some problems where greedy doesn't give an optimal solution, but it's provably *close* to optimal in some sense (e.g., traveling salesperson).

Proving a greedy algorithm correct

Method 1: “the greedy algorithm stays ahead”

- Essentially a proof by induction
- You compare the list of choices made by the greedy algorithm, to a similar list of choices made by an optimal solution
- Show that at each stage, the greedy choice is *at least as good* as the choice in the optimal solution
- Example: algorithm for interval scheduling problem (4.1)

Proving a greedy algorithm correct

Method 2: exchange arguments

- Prove that if **O** is an optimal solution, and **G** is the greedy solution, then you can modify **O** slightly to get **O'** such that:
  - **O'** is more similar to **G** than **O** was
  - **O'** is at least as good a solution as **O**
- Then describe how you can repeatedly modify **O** until it is *the same solution* as **G**, without ever decreasing the solution quality
- Examples of what "more similar to" might mean:
  - Has more edges in common with
  - Selects more of the same jobs
  - Has fewer elements out of order compared with the greedy solution