4. Greedy Algorithms

Optimization problems

- For the next four worksheets, we will consider optimization problems:
  - We have a problem with several valid solutions.
  - There is an objective function that tells us how good or bad each valid 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.
    - The objective function is the total lateness.
  - Given a weighted connected graph, find a spanning tree with the smallest total edge weight.
    - The 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 solution.
- We can think of a greedy algorithm as making a sequence of choices.
- There is no precise definition of greedy.

Defining greedy algorithms

- Examples of choice:
  - interval with the earliest finishing time.
  - job with the earliest deadline.
  - item needed further in the future.
  - 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.

Proving a greedy algorithm correct

- Method 1: “the greedy algorithm stays ahead”
  - It’s basically a proof by induction.
  - You compare
    - The list of choices made by the greedy algorithm, to
    - A similar list for an optimal solution
  - You show that at each stage, the greedy choice is *at least as good* as the choice in the optimal solution.
  - Examples:
    - The algorithm for the interval scheduling problem (4.1).

Proving a greedy algorithm correct

- Method 2: exchange arguments
  - Prove that if S is an *arbitrary* solution, and G is the greedy solution, then you can modify S slightly to get S' such that:
    - S' is more similar to G than S.
    - S' is at least as good a solution as S.
  - Examples of what "more similar to" might mean:
    - Has more edges in common with.
    - Has a longer initial sequence that's the same as.

Proving a greedy algorithm correct

- Method 2: exchange arguments
  - Why this works:
    - You start from any arbitrary solution S_0.
    - You get S_1 which is at least as good as S_0 and closer to G.
    - You get S_2 which is at least as good as S_1 and closer to G.
    - You get S_3 which is at least as good as S_2 and closer to G.
    - ...
    - You get G which is at least as good as S_t and closer to G.

Proving a greedy algorithm correct

- Method 2: exchange arguments
  - By induction (or transitivity):
    - G is "at least as good" a solution as S_0.
  - This works no matter what S_0 is
    - Even if S_0 is an optimal solution.
  - So G is at least as good as an optimal solution.
  - Therefore G is optimal.