10.1.1
What is an algorithmic problem?

What is an algorithmic problem?

Simplest and robust definition: An algorithmic problem is simply to compute a function $f : \Sigma^* \to \Sigma^*$ over strings of a finite alphabet.

Algorithm $\mathcal{A}$ solves f if for all **input strings** w, $\mathcal{A}$ outputs $f(w)$.

Typically we are interested in functions $f : D \to R$ where $D \subseteq \Sigma^*$ is the domain of f and where $R \subseteq \Sigma^*$ is the range of f.

We say that $w \in D$ is an **instance** of the problem. Implicit assumption is that the algorithm, given an arbitrary string w, can tell whether $w \in D$ or not. Parsing problem! The **size of the input** w is simply the length $|w|$.

The domain D depends on what **representation** is used. Can be lead to formally different algorithmic problems.

Types of Problems

We will broadly see three types of problems.

1. Decision Problem: Is the input a YES or NO input?
   Example: Given graph G, nodes s, t, is there a path from s to t in G?
   Example: Given a CFG grammar G and string w, is $w \in L(G)$?
2. Search Problem: Find a solution if input is a YES input.
   Example: Given graph G, nodes s, t, find an s-t path.
3. Optimization Problem: Find a best solution among all solutions for the input.
   Example: Given graph G, nodes s, t, find a shortest s-t path.

Analysis of Algorithms

Given a problem P and an algorithm $\mathcal{A}$ for P we want to know:

- Does $\mathcal{A}$ **correctly** solve problem P?
- What is the **asymptotic worst-case running time** of $\mathcal{A}$?
- What is the **asymptotic worst-case space** used by $\mathcal{A}$.

Asymptotic running-time analysis: $\mathcal{A}$ runs in $O(f(n))$ time if:

“for all n and for all inputs I of size n, $\mathcal{A}$ on input I terminates after $O(f(n))$ primitive steps.”

Algorithmic Techniques

- Reduction to known problem/algorithm
- Recursion, divide-and-conquer, dynamic programming
- Graph algorithms to use as basic reductions
- Greedy

Some advanced techniques not covered in this class:

- Combinatorial optimization
- Linear and Convex Programming, more generally continuous optimization method
- Advanced data structure
- Randomization
- Many specialized areas

THE END

...

(for now)