Subject introduction

Table of Contents

Algorithms

- Sequence of *unambiguous* instructions for solving a problem to obtain required output for *legitimate input* in a *finite* amount of time
- multiple valid solutions with different efficiency

Greatest common divisor

Euclid's algorithm `gcd(m, n)= gcd(n, m mod n)`

For example

```
gcd(24, 60) = gcd(60, 24)
            = gcd(24, 12)
            = gcd(12, 0)
            = 12
```

Since `gcd(m, 0)= m`

Sieve of Eratosthenes

- algorithm to generate consecutive primes not exceeding a given integer `n > 1`
- procedure:
  - generate a list of prime candidates from 2 to n
  - loop over the list, each time eliminating candidates that are multiples of 2, 3, …
  - no pass for 4 is necessary as all multiples of 4 have already been eliminated
  - algorithm continues until no more numbers can be eliminated; remaining numbers are prime

- what is largest `p` whose multiples can still remain on the list to make further iterations of the algorithm necessary?
  - if `p` is a number whose multiples are being eliminated on the current pass, first multiple we should consider is `p.p` because all smaller multiples `2p, ..., (p-1)p` have been eliminated on earlier passes
  - `p.p` should be less than `n` otherwise it isn't a candidate, i.e.

```
p \leq \lfloor\sqrt{n}\rfloor
```

Algorithmic Problem Solving

- understand the problem
- understand the capabilities of the hardware
- decide between exact/approximate solution
- choose design techniques
- design algorithm and data structure
- prove correctness: prove that algorithm yields required result for every legitimate input in finite time
  - often uses mathematical induction
  - for approximation algorithms you need to show error does not exceed defined limit
- analysis
  - time efficiency: run time
  - space efficiency: memory
  - generality
- implement the algorithm

Important problem types

- sorting: rearrange list items in non-decreasing order
  - stable: preserves relative order of equal elements
  - typically algorithms that switch keys far apart are not stable but are faster
  - in-place: doesn't require extra memory to run
- searching: find a given value (*search key*) in a given set
- string processing

  - e.g. string matching

- graph problems
  - graph is a collection of vertices, connected by edges
  - e.g. graph traversal, shortest path
  - graph-coloring: assign smallest number of colors to vertices of a graph such that no two adjacent vertices are the same color (event scheduling)
  - travelling salesman problem: shortest tour through n cities that visits each city only once

- combinatorial problems
  - ask to find a combinatorial object satisfying constraints (e.g. permutation, combination, subset)
  - typically most difficult class of problems: number of objects grows extremely fast with problem size

- geometric problems: points, lines and polygons
  - e.g. computer graphics, robotics, tomography
  - closest-pair problem: given n points in the plane, find the closest pair among them
  - convex-hull problem: smallest convex polygon that contains all points of a set

- numerical problems: mathematical objects of continuous nature
  - solving systems of equations, computing integrals, evaluating functions

Fundamental Data Structures