• Homework 7 due Friday at 5pm
• Final Exam:
  • Thursday, May 14 from 4pm to 5:50pm in WWH 101
  • ICPC world finals May 16—21 in Casablanca!
  • Must bring a Laptop! (Email me asap if you do not have one!)
Exercise: You are given a DAG of potential jobs for your company to complete, where a job can only be completed once its predecessors in the DAG have been. You also have an integer value $v$ (positive or negative) associated with each job. Determine the subset of jobs that yields the most total value.
• Build the dag where each node points to the nodes it depends on with infinite capacity
• Create a source s and a sink t
  • If a task has positive value, then connect the source to it with that value
  • If a task has negative value, then connect it to the sink with its absolute value
• As each vertex becomes paired with either s or t, the value of the min cut exactly determines the optimal project selection
  • To get the optimal value, compute the sum of all positive project values and subtract the value of the cut
• More detailed explanation starting on page 5 [here](#)
• **BigInteger class**
  • Arbitrarily large integer type
    • But has to fit in memory of the program
  • If you're asked to work with very large numbers (larger than $2^{63}$), use BigInteger!
  • Basic operations: +, -, /, *, %, pow
  • Advanced operations: gcd, modulo arithmetic, base conversion

• **Also, BigDecimal class**

• **3rd party libraries for C++:**
  • GMP, Victor Shoup’s NTL
Number theory is fairly popular in programming contests
  “Recreational mathematics”

Recall: prime numbers
  An integer $p \geq 2$ divisible by 1 and $p$
  First 10 primes: \{2, 3, 5, 7, 11, 13, 17, 19, 23, 29\}
  Prime counts:
    - 1-100: 25 primes
    - 1-1000: 168 primes
    - 1-10,000: 1229 primes
  Prime number theorem: $\pi(x) \sim x / \log(x)$
  Useful for factoring
• **Testing if a number is prime, isPrime(n)**
  • Check if n is divisible by 2, ..., n-1
    • $O(n)$
  • Check if n is divisible by 2, 3, ..., $\sqrt{n}$
    • $O(\sqrt{n})$
  • Check if n is divisible by 3, 5, 7, ..., $\sqrt{n}$
    • $O(\sqrt{n}/2) = O(\sqrt{n})$ – plus check of 2
  • Check if n is divisible by primes $\leq \sqrt{n}$
    • $O(\pi(\sqrt{n})) = O(\sqrt{n} / \log(\sqrt{n}))$
      • $\pi(m)$ = number of primes up to m
• Generating primes from 1 to N
  • Naive algorithm:
    • for (int i = 0; i <= N; i++) {
      • if (isPrime(i)) print(i);
    }
  • Better algorithm:
    • Sieve of Eratosthenes
Sieve of Eratosthenes

- A 1D boolean array $P$ from 0 to $N$
  - Initialized to true except 0 and 1
- Pick $i$, the next True element in the array
  - Mark $P[j] = False$ for $j = i \times 2, i \times 3, \ldots, i \times k$ where $i(k+1) > N$

Primality testing:
- Use sieve to generate primes
- Check divisibility on all primes up to $\sqrt{N}$
### Sieve of Eratosthenes: Example

|   | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | ... | 51 | 52 | 53 | 54 | 55 | ... | 75 | 76 | 77 |
|---|---|---|---|---|---|---|---|---|---|---|     |    |    |    |    |    |     |    |    |    |
| F | F | T | T | T | T | T | T | T | T | T |     | T  | T  | T  | T  | T  |     | T  | T  | T  |
| F | F | I | T | F | T | F | T | F | T | T |     | F  | T  | F  | T  | T  |     | F  | T  | T  |
| F | F | T | I | F | T | F | T | F | F | F |     | F  | F  | T  | F  | T  |     | F  | F  | T  |
| F | F | T | T | F | I | F | T | F | F | F |     | F  | F  | T  | F  | F  |     | F  | F  | T  |
| F | F | T | T | F | T | F | I | F | F | F |     | F  | F  | T  | F  | F  |     | F  | F  | F  |

This table illustrates the process of the Sieve of Eratosthenes, where multiples of each prime number starting from 2 are marked as False (F) in the corresponding positions.
• **GCD, greatest common divisor of two numbers**
  • Euclidean algorithm
    • \( \gcd(a, 0) = a \)
    • \( \gcd(a, b) = \gcd(b, a \mod b) \)
  • In code:
    • ```cpp
      int gcd(int a, int b) { return b == 0 ? a : gcd(b, a % b); }
    ```

• **LCM, least common multiple of two numbers**
  • \( \text{lcm}(a, b) = \frac{a \times b}{\gcd(a, b)} \)
Exercise 1. Pandigital primes - Project Euler Problem 41

We shall say that an $n$-digit number is pandigital if it makes use of all the digits 1 to $n$ exactly once. For example, 2143 is a 4-digit pandigital and is also prime.

What is the largest $n$-digit pandigital prime that exists?
Exercise 2. Consecutive prime sum - Project Euler Problem 50

The prime 41, can be written as the sum of six consecutive primes:

$$41 = 2 + 3 + 5 + 7 + 11 + 13$$

This is the longest sum of consecutive primes that adds to a prime below one-hundred. The longest sum of consecutive primes below one-thousand that adds to a prime, contains 21 terms, and is equal to 953.

Which prime, below one-million, can be written as the sum of the most consecutive primes?
Exercise 3. Prime summations - Project Euler Problem 77

It is possible to write ten as the sum of primes in exactly five different ways:

\[ 7 + 3 \]
\[ 5 + 5 \]
\[ 5 + 3 + 2 \]
\[ 3 + 3 + 2 + 2 \]
\[ 2 + 2 + 2 + 2 + 2 \]

What is the first value which can be written as the sum of primes in over five thousand different ways?
• Many probability problems can be reduced to counting: combinatorics!
• How to compute the number of subsets of \{1, 2, \ldots, n\} of size \(k\)?
  • In code:
    ```
    long ret = 1;
    for (int i = 0; i < k; +i) {
      ret *= n – i;
      ret /= 1 + i;
    }
    ```
• Definitions: Let \( f \) be a function \( f : X \rightarrow Y \)
  • One-to-one (injective):
    \[ \forall a, b \in X, f(a) = f(b) \Rightarrow a = b \]
  • Onto (surjective):
    \[ \forall y \in Y, \exists x \in X \Rightarrow f(x) = y \]
  • Bijection: a function that is both one-to-one and onto

• Exercise: Let \( A, B \) be sets with \( |A| = 7 \) and \( |B| = 3 \). How many functions \( f : A \rightarrow B \) are there? How many are onto? How many are one-to-one?
**Exercise**: How many solutions are there to the equation $a + b + c = 10$ where $a$, $b$, $c$ are positive integers? What if zeros are allowed?

**Exercise**: How many different ways are there for a class of 10 students to pair up?
• Often in CS, we’re talking about probability on discrete sets
  • Assign a non-negative probability to each element of a set such that probabilities sum to one
• Random variables
  • A function that assigns a real number to each element in our probability space
• Expectation
  • Linearity: $E[aX + bY] = aE[X] + bE[Y]$
  • Law of total expectation: $E[E[X|Y]] = E[X]$
    • E.g., suppose 51% of the population is male, average height of male is 5’11 and average height of female is 5’5, what is average height of population?
• Java and C++ each have a function that will give you a pseudo-random number from the uniform distribution [0, 1]
• But how do you sample from an arbitrary distribution?
• tl;dr:
  • Let U be a standard uniform random variable
  • Let X be any continuous random variable whose CDF F is strictly increasing over its support
    • Strictly increasing so that F^{-1} is well defined
  • Then F^{-1}(U) \sim X