Lecture 6
Combinatorics

Euiseong Seo
(euiseong@skku.edu)
Combinatorics

- A branch of mathematics concerning the study of finite or countable discrete structures
- Notorious for their reliance on cleverness and insight
- Once you look at the problem in the right way, the answer suddenly becomes obvious
Counting Techniques

- **Product rule**
  - If there are $|A|$ possibilities from set $A$ and $|B|$ possibilities from set $B$, then there are $|A| \times |B|$ ways to combine one from $A$ and one from $B$

- **Sum rule**
  - If there are $|A|$ possibilities from set $A$ and $|B|$ possibilities from set $B$, then there are $|A| + |B|$ ways for either $A$ or $B$ to occur – assuming the elements of $A$ and $B$ are distinct

- **Inclusion-Exclusion formula**
  - $|A \cup B| = |A| + |B| - |A \cap B|$
  - $|A \cup B \cup C| = |A| + |B| + |C| - |A \cap B| - |B \cap C| - |A \cap B \cap C| + |A \cap B \cap C|$
Counting Techniques

- **Bijection**
  - One-to-one mapping between the elements of one set and the elements of another
  - Counting the size of one of the sets automatically gives you the size of the other
Permutation

- An arrangement of n items
- What if items are reusable?
- Every item appears exactly once
- \( n! = \prod_{i=1}^{n} i \)
- What if items are reusable?
Subsets

- A selection of elements from n possible items
- \(2^n\) distinct subsets of n things
- Never forget the empty subset
Strings

- A sequence of items which are drawn with repetition
- Number of binary strings of length \( n \) is identical to the number of subsets of \( n \) items (why?)
Recurrence Relations

- An equation which is defined in terms of itself
- Recursively defined structures
- Example: Fibonacci numbers
  - \( F_n = F_{n-1} + F_{n-2} \)
Why Recurrence Relations?

- Many natural functions are easily expressed
- Polynomials
  - \( a_n = a_{n-1} + 1, \ a_1 = 1 \rightarrow a_n = n \)
- Exponentials
  - \( a_n = 2a_{n-1}, \ a_1 = 2 \rightarrow a_n = 2^n \)
- Factorials
  - \( a_n = na_{n-1}, \ a_1 = 1 \rightarrow a_n = n! \)
- It is often easy to find a recurrence as the answer to a counting problem
- Closed form solutions of recurrences would look beautiful, but not always necessary
Binomial Coefficients

- Number of ways to choose \( k \) things out of \( n \) possibilities

\[
\binom{n}{k} = \frac{n!}{(n-k)!k!}
\]

- Coefficient of \((a+b)^n\)
  - What is the coeff of \( a^k b^{n-k} \)?
  - How many ways to choose \( k \) a-terms out of \( n \)
  - \((a + b)^3 = 1a^3 + 3a^2b + 3ab^2 + 1b^3\)
  - \((a + b)^3 = aaa + 3aab + 3abb + bbb\)
Binomial Coefficients

- Pascal’s Triangle

\[
\begin{array}{cccccc}
 & & & & & \\
 & & & & 1 & \\
 & & & 1 & 1 & \\
 & & 1 & 2 & 1 & \\
 & 1 & 3 & 3 & 1 & \\
1 & 4 & 6 & 4 & 1 & \\
1 & 5 & 10 & 10 & 5 & 1 \\
\end{array}
\]

- (n+1)st row of the table gives the value of \( \binom{n}{k} \)
  for \( 0 \leq k \leq n \)

- Can be used to calculate binomial coefficients
  - To prevent overflow errors from factorials
Using Pascal Tree for Binomial coefficients

- Dynamic programming

```c
#define MAXN 100 /* largest n or m */

long binomial_coefficient(n,m)
int n,m; /* computer n choose m */
{
    int i,j; /* counters */
    long bc[MAXN][MAXN]; /* table of binomial coefficients */

    for (i=0; i<=n; i++) bc[i][0] = 1;
    for (j=0; j<=n; j++) bc[j][j] = 1;
    for (i=1; i<=n; i++)
        for (j=1; j<i; j++)
            bc[i][j] = bc[i-1][j-1] + bc[i-1][j];

    return( bc[n][m] );
}
```
Fibonacci Numbers

- Defined by the recurrence
  - $F_n = F_{n-1} + F_{n-2}$
  - $F_0 = 0$, $F_1 = 1$

- Closed form

$$F_n = \frac{1}{\sqrt{5}} \left( \left( \frac{1 + \sqrt{5}}{2} \right)^n - \left( \frac{1 - \sqrt{5}}{2} \right)^n \right)$$
Catalan Numbers

- Defined by a recurrence form

\[ C_n = \sum_{k=0}^{n-1} C_k C_{n-1-k} = \frac{1}{n+1} \binom{2n}{n} \]

\[ C_0 = 1 \]

- Occur in a surprising number of problems in combinatorics

- How many ways to build a balanced formula from \( n \) sets of left and right parentheses?
Eulerian Numbers

- The Eulerian numbers $\langle \frac{n}{k} \rangle$ count the number of permutations of length $n$ with exactly $k$ ascending sequences of runs.

- $\langle \frac{n}{k} \rangle = k \langle \frac{n-1}{k} \rangle + (n - k + 1) \langle \frac{n-1}{k-1} \rangle$, Why?
Integer Partitions

- An integer partition of \( n \) is an unordered set of positive integers which add up to \( n \)
  - Seven partitions of 5: (5), (4,1), (3,2), (3,1,1), (2,2,1), (2,1,1,1), (1,1,1,1,1)
  - \( f(n,k) \): the number of integer partitions of \( n \) with the largest part at most \( k \)
  - \( f(n,k) = f(n-k, k) + f(n,k-1) \), why?
  - \( f(1,1) = 1 \)
  - \( f(n,k) = 0 \) where \( k > n \)
How many pieces of land?

You are given an elliptical-shaped land and you are asked to choose \( n \) arbitrary points on its boundary. Then you connect each point with every other point using straight lines, forming \( n(n - 1)/2 \) connections. What is the maximum number of pieces of land you will get by choosing the points on the boundary carefully?

Dividing the land when \( n = 6 \).
How many pieces of land?

Input

The first line of the input file contains one integer \( s \) \((0 < s < 3,500)\), which indicates how many input instances there are. The next \( s \) lines describe \( s \) input instances, each consisting of exactly one integer \( n \) \((0 \leq n < 2^{31})\).

Output

For each input instance output the maximum possible number pieces of land defined by \( n \) points, each printed on its own line.

Sample Input

\[
\begin{array}{c}
4 \\
1 \\
2 \\
3 \\
4
\end{array}
\]

Sample Output

\[
\begin{array}{c}
1 \\
2 \\
4 \\
8
\end{array}
\]
Steps

Consider the process of stepping from integer $x$ to integer $y$ along integer points of the straight line. The length of each step must be non-negative and can be one bigger than, equal to, or one smaller than the length of the previous step.

What is the minimum number of steps in order to get from $x$ to $y$? The length of both the first and the last step must be 1.

Input

The input begins with a line containing $n$, the number of test cases. Each test case that follows consists of a line with two integers: $0 \leq x \leq y < 2^{31}$.

Output

For each test case, print a line giving the minimum number of steps to get from $x$ to $y$.

Sample Input          Sample Output

3                     3
45 48                  3
45 49                  4
45 50
How Many Trees

A binary search tree is a binary tree with root \( k \) such that any node \( v \) reachable from its left has \( \text{label}(v) < \text{label}(k) \) and any node \( w \) reachable from its right has \( \text{label}(v) > \text{label}(k) \). It is a search structure which can find a node with label \( x \) in \( O(n \log n) \) average time, where \( n \) is the size of the tree (number of vertices).

Given a number \( n \), can you tell how many different binary search trees may be constructed with a set of numbers of size \( n \) such that each element of the set will be associated to the label of exactly one node in a binary search tree?

Input and Output

The input will contain a number \( 1 \leq i \leq 300 \) per line representing the number of elements of the set. You have to print a line in the output for each entry with the answer to the previous question.
## How Many Trees

### Problem D

How Many Trees?

**Input:**
standard input

**Output:**
standard output

**Time Limit:** 100 seconds

**Memory Limit:** 32 MB

A binary search tree is a binary tree with root $k$ such that any node $v$ reachable from its left has $\text{label}(v) < \text{label}(k)$ and any node $w$ reachable from its right has $\text{label}(v) > \text{label}(k)$. It is a search structure which can find a node with label $x$ in $O(n \log n)$ average time, where $n$ is the size of the tree (number of vertices).

Given a number $n$, can you tell how many different binary search trees may be constructed with a set of numbers of size $n$ such that each element of the set will be associated to the label of exactly one node in a binary search tree?

### Input and Output

The input will contain a number $1 \leq i \leq 300$ per line representing the number of elements of the set. You have to print a line in the output for each entry with the answer to the previous question.

### Sample Input

```
1
2
3
```

### Sample Output

```
1
2
5
```