• **Homework 1**  
  • Due Saturday at 2am (so, basically Friday night)
• **Homework 2**  
  • Will be posted tomorrow afternoon-ish
• Build a Fenwick tree (table) for $L = [1, 2, 1, 3, -1, 1]$
• Find the sum of the range [3, 6]

<table>
<thead>
<tr>
<th>Index$_{10}$</th>
<th>Index$_2$</th>
<th>Interval</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>[1, 1]</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>[1, 2]</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>11</td>
<td>[3, 3]</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>[1, 4]</td>
<td>7</td>
</tr>
<tr>
<td>5</td>
<td>101</td>
<td>[5, 5]</td>
<td>-1</td>
</tr>
<tr>
<td>6</td>
<td>110</td>
<td>[5, 6]</td>
<td>0</td>
</tr>
</tbody>
</table>
• What happens to the table when L[1] is updated from 1 to 4?
  • All entries corresponding to ranges including affected index need to be updated

<table>
<thead>
<tr>
<th>Index_{10}</th>
<th>Index_2</th>
<th>Interval</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>[1, 1]</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>[1, 2]</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>11</td>
<td>[3, 3]</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>[1, 4]</td>
<td>7</td>
</tr>
<tr>
<td>5</td>
<td>101</td>
<td>[5, 5]</td>
<td>-1</td>
</tr>
<tr>
<td>6</td>
<td>110</td>
<td>[5, 6]</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Index_{10}</th>
<th>Index_2</th>
<th>Interval</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>[1, 1]</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>[1, 2]</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>11</td>
<td>[3, 3]</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>[1, 4]</td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>101</td>
<td>[5, 5]</td>
<td>-1</td>
</tr>
<tr>
<td>6</td>
<td>110</td>
<td>[5, 6]</td>
<td>0</td>
</tr>
</tbody>
</table>
• If you don’t store the original list but have its Fenwick tree, how can you rebuild the original list?
• Can you use a Fenwick tree for min/max queries?
Suppose you have a set of $n = 10^6$ of the numbers $\{1, 2, \ldots, n\}$. You are given a list of instructions each taking one of the following forms:

(a) Remove the number $k$.
(b) Output the $j$th largest remaining number.

Explain how to use a Fenwick tree to deal with each instruction in $\log(n)$ time.
Suppose you are given a list of instructions regarding the multiset $S$ each taking one of the following forms:

(a) Insert the value $k$, where $k$ is a 32-bit signed integer.
(b) Remove the value $k$, where $k$ is a 32-bit signed integer.
(c) Give the percentage of the list smaller than $x$, where $x$ is a 32-bit signed integer.
(d) Give the $j$th largest element in the set.

Assume further that you are guaranteed only $10^6$ distinct values will occur (but each may occur many times). Explain how to a Fenwick tree to efficiently handle these instructions.
Exercise  Given an integer $x$, write a function $\text{reverseBytes}(x)$ that returns an integer whose bytes are reversed. For example, $\text{reverseBytes}(3628) = 739115008$ because $3628 = 0x00000E2C$ and $739115008 = 0x2C0E0000$. 
reverseBytes(x)
return (x & 0x000000FF) << 24
          | (x & 0x0000FF00) << 8
          | (x & 0x00FF0000) >> 8
          | (x & 0xFF000000) >>> 24
...or Integer.reverseBytes(x) in Java
Exercise   Write the following functions using bitmasks: \( \text{isEven}(x) \), \( \text{evenBitsOnly}(x) \), \( \text{setNthBit}(x, N) \). For example:

- \( \text{isEven}(5) \) returns false
- \( \text{evenBitsOnly}(14) \) returns 10
- \( \text{setNthBit}(11, 2) \) returns 15
• $2^{10} = 1024 \approx 1K$
• $2^{20} = 1048576 \approx 1M$
• $10! = 3628800 \neq 3M$
• Max signed integer value: $2147483647 \approx 2B$
• Power set of a set of size N: $2^N$
• Today we’ll look at: iterative loops, permutations, subsets, recursive backtracking
• For the following exercises, describe your approach to solving the problem and give its runtime
Exercise  (Perfect square problem)  Given an integer $N$, determine if $N$ is a perfect square, i.e., if $\exists x$ such that $x^2 = N$ and $x$ is an integer. Constraint: $0 < N < 10^6$
• Math solution: take the square root, determine if it’s an integer
• Search solution: compute the square of numbers up to $\sqrt{N}$
  • Binary search
Exercise  (Use all digits problem) Given a number $N$, find all pairs of 5-digit numbers that between them use the digits 0 through 9 exactly once and such that $abcde/fghij = N$ (each letter represents a unique digit [0–9]). Constraint: $2 < N < 79$
• **Solution 1: Rewrite $N \times abcde = fghij$**
  - Check all $abcde$ (100k values) and check that $abcde$ and $fghij$ are different digits
  - Don’t forget to pad with 0

• **Solution 2: Check all permutations of $abcde$**
  - You could even run all permutations of 0123456789 and divide
Exercise  (Movie seating problem)  $N$ friends go to a movie and sit in a row with $N$ consecutive open seats. There are $M$ seating constraints, i.e., two people $a$ and $b$ must sit at least (or at most) $c$ seats apart. How many possible seating configurations are there? Constraints: $0 < N \leq 8$ and $0 \leq M \leq 20$

*Input:* Each case begins with a line containing two integers $N$ and $M$, followed by $M$ lines containing a seating constraint, $a, b, c$. If $c$ is positive, $a$ and $b$ must be at most $c$ seats apart. Otherwise, if $c$ is negative, $a$ and $b$ must be at least $-c$ seats apart.

<table>
<thead>
<tr>
<th>Sample input</th>
<th>Sample output</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 0</td>
<td>2</td>
</tr>
<tr>
<td>3 1</td>
<td>4</td>
</tr>
<tr>
<td>0 1 1</td>
<td>8</td>
</tr>
<tr>
<td>4 2</td>
<td></td>
</tr>
<tr>
<td>0 1 1</td>
<td></td>
</tr>
<tr>
<td>0 2 −2</td>
<td></td>
</tr>
</tbody>
</table>
• **Important constraints:**
  • Up to 8 friends
  • Up to 20 constraints
  • $8! \times 20$ checks --- brute force is possible!

• **Solution:**
  • Try all $8!$ permutations and check constraints
  • In C++, start with the lexicographically smallest (i.e., sorted) permutation and use `next_permutation`
  • In Java, you need to implement this from scratch

• **How do you loop through all permutations?**
  • Recursively
  • Iteratively
int permutation[] = new int[8];
bool seen[] = new bool[8];

public void findPermutation(int depth) {
    if (depth == N) {
        // found a full permutation
        return;
    }
    for (int i = 0; i < N; i++) {
        if (permutation[i] == -1 && !seen[i]) {
            permutation[i] = depth;
            seen[i] = true;
            findPermutation(depth+1);
            permutation[i] = -1;
            seen[i] = false;
        }
    }
}
• Intuitively:
  • We want to find the next lexicographic ordering
    • If we think of a list as a number, we want the permutation of its digits which form the
      smallest number larger than the original

• Algorithm: Given list \( L = [0, 1, 2, 5, 3, 3, 0] \)
  1. Let \( i = |L| - 1 \) (i.e., index of last element). Loop:
  2. If \( L[i-1] < L[i] \):
     1. Let \( j = i - 1 \)
     2. Find largest index \( k \) such that \( L[j] \geq L[k] \) and swap \( L[j] \) and \( L[k] \)
     3. Reverse \( L[i..\text{end}] \)
  Else if no more permutations when \( L \) is sorted in descending order
• Running time: each shuffle is $O(n)$, and there are $n!$ shuffles to go through all permutations.
  • However, performing all $n!$ shuffles runs in $O(n!)$ time
• Benefits:
  • Iterative
  • Lexicographic ordering
  • Allows repeated items in the list
• Reference and code
Exercise (Water gates problem) A dam has \( N \) water gates to release water from the dam’s reservoir if the water level is getting too high. You are given each gate’s flow rate and cost of use. Determine the minimum total cost of use for a total flow rate of at least \( F \).

Constraint: \( 2 \leq N \leq 20 \)

Input: Each case begins with a line containing two integers \( N \) and \( F \), the number of gates and the total flow rate to achieve. Following are \( N \) lines describing a gate which contains a pair of integers \( r \) and \( c \), the flow rate and cost of use for the gate. The sum of \( r \) and the sum of \( c \) will be less than \( 10^9 \).

<table>
<thead>
<tr>
<th>Sample input</th>
<th>Sample output</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 5</td>
<td>5</td>
</tr>
<tr>
<td>5 5</td>
<td>11</td>
</tr>
<tr>
<td>5 8</td>
<td></td>
</tr>
<tr>
<td>3 5</td>
<td></td>
</tr>
<tr>
<td>2 4</td>
<td></td>
</tr>
<tr>
<td>3 7</td>
<td></td>
</tr>
<tr>
<td>5 12</td>
<td></td>
</tr>
</tbody>
</table>
“Water gates” solution:

- Generate all subsets of the water gates
  - If the flow rate of the subset is more than $F$, consider it as a solution
- How do you generate all subsets?
  - Bitmasks!
- How many subsets are there?
  - N choose M
// calculates the next integer with the same number of 1-bits
unsigned int snoob(unsigned int x)
{
    unsigned int smallest, ripple, ones;
    // x = xxx0 1111 0000
    smallest = x & -x; // 0000 0001 0000
    ripple = x + smallest; // xxx1 0000 0000
    ones = x ^ ripple; // 0001 1111 0000
    ones = (ones >> 2) / smallest; // 0000 0000 0111
    return ripple | ones; // xxx1 0000 0111
}
• Place 8 queens on an 8x8 chessboard and count the number of solutions
• No queens are allowed to attack each other
Naïve solution:

• Enumerate all possible positions on the chessboard and simulate.
• Problem? 64 choose 8 \approx 4B

Can you prune the search space?
• Two queens cannot be in the same column
  • Place one queen in each column.
  • Represent this as an array of digits 0-7
  • The index of the digit is the column, the digit is the row

• Two queens cannot be in the same row
  • So each value in the array is unique
  • Now we have reduced the complete search to be over permutations of digits 0 to 7
    • $8! = 40320$

• Two queens cannot be in the same diagonal
  • If you permute using recursive backtracking, you can preemptively cut out
Competitive Programming 3.1, 3.2

Next class: greedy, divide and conquer, binary search