1. Think Graphically

\[ a^2 + b^2 = c^2 \]
Proof by Picture

\[
\text{sum of first } n \text{ odd numbers} = n^2
\]

\[
\frac{1}{4} + \left(\frac{1}{4}\right)^2 + \left(\frac{1}{4}\right)^3 + \ldots = \frac{1}{3}
\]

“Parameter” Space

- You are given \(N\) intervals \([a_1, b_1] \ldots [a_N, b_N]\).
- For each one, compute the number of other intervals contained within it.
You are given N intervals \([a_1, b_1] \ldots [a_N, b_N]\).

For each one, compute the number of other intervals contained within it.

Note that each item above is characterized by two parameters: \(a\) and \(b\). Hence, we can represent each one with a point \((a, b)\) in the plane. This makes the problem easier to visualize...

A line in the plane is also characterized by two parameters: \(ax + by = 1\) or \(y = ax + b\).

So we could also represent our n intervals by n lines (say, \([a,b] \rightarrow y = ax + b\)). Does this make things easier to visualize?
For many problems on intervals, we can transform them into meaningful (and sometimes simpler) problems about points or lines.

Likewise, problems about points and lines can often be viewed differently as problems on lines, points, or sometimes intervals.

In computational geometry, we often exploit geometric duality – exchanging the roles of points and lines.

- Slope-intercept duality: point (a,b) ↔ line y = ax - b.
- Polar duality: point (a,b) ↔ line ax + by = 1.

The preceding two problems are equivalent due to point-line duality in the 2d plane!

The roles of points and lines reverse when we dualize:

- Recall that two points determine a unique line, and likewise that two lines determine a unique point.
- If L1 and L2 are two lines intersecting at point P, then dual(L1) and dual(L2) are two points that lie on the common line dual(P).
- If P1 and P2 are two points determining line L, then dual(L) is the point of intersection between the lines dual(P1) and dual(P2).
Geometric Duality

• Duality allows us to solve twice as many problems with the same number of techniques!
  – Caveat: only works for “incidence”-related problems; no distances allowed.

• Example: half-space intersection (containing the origin) and convex hulls, via polar duality.

  ![Diagram]

• Example: given n points, find the two of them that determine a line of min (or max) slope.

Parameter Space: Robot Arms

• Suppose a robot arm has 4 movable joints, each with one degree of freedom.

• You want to move the arm from its current state to a new “goal” state (say, with the end of the arm at a certain position).

• Certain joint configurations are not valid (i.e., the arm can’t fold back and cross through itself!)

• Moving the arm becomes an easier problem, conceptually, if we view it as finding a path through a 4d “parameter space” in which obstacles correspond to invalid configurations.
Another Example

• Given an array $A[1...N]$ of integers, please preprocess it so that we can quickly answer queries of the form: “tell me the number of distinct values present in the subarray $A[i...j]$”.

Another Example

• Given an array $A[1...N]$ of integers, please preprocess it so that we can quickly answer queries of the form: “tell me the number of distinct values present in the subarray $A[i...j]$”.

• Preprocessing step: compute $B[1...N]$, where $B[j]$ tells us the index of the most recent occurrence of $A[j]$ in the array (-1 if none). For example:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>(3)</td>
<td>(3)</td>
<td>(1)</td>
<td>(4)</td>
<td>(1)</td>
<td>(3)</td>
<td>(6)</td>
<td>(1)</td>
<td>(4)</td>
<td>(1)</td>
</tr>
<tr>
<td>B</td>
<td>(-1)</td>
<td>1</td>
<td>(-1)</td>
<td>(-1)</td>
<td>3</td>
<td>2</td>
<td>(-1)</td>
<td>5</td>
<td>4</td>
<td>8</td>
</tr>
</tbody>
</table>

• How quickly can we compute $B[1...N]$?
Now visualize each element $i$ as the point $(i, B[i])$.
A query now involves counting points in a rectangular region of the plane (for which we have good data structures, like range trees).

### Counting Distinct Elements

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>3</td>
<td>6</td>
<td>1</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>3</td>
<td>2</td>
<td></td>
<td>-1</td>
<td>5</td>
<td>4</td>
<td>8</td>
</tr>
</tbody>
</table>

### 2. Think Symbolically
Hanging a Picture the Fun Way...

1 CW 2 CW 1 CCW 2 CCW

“1 CW 2 CW 1 CCW 2 CCW”
Hanging a Picture the Fun Way...

“1 CW 2 CW 1 CCW 2 CCW”

Hanging a Picture the Fun Way...

“1 CW 2 CW 1 CCW 2 CCW 3 CW 2 CW 1 CW 2 CCW 1 CCW 3 CCW”
3. Equivalent Objects

- Given an array $A[1…N]$ of integers, please count the number of inversions – pairs of elements that are out of order with respect to their current locations in $A$.
  
  (this gives a good measurement of how “unsorted” $A$ is)

$$A[1…4] = 4, 1, 2, 3$$

N lines in the plane
These are all “Equivalent”…

- Strings of balanced parentheses.
- Nested collections of intervals.
- “Horizons”.
- Binary rooted trees.
- Non-binary rooted trees.
- Triangulations of convex polygons.
- Laminar families of sets.

- On his website, MIT mathematician Richard Stanley maintains a list of ~200 total objects!

4. Divide & Conquer

- Divide problem into (typically 2) pieces
- Solve those pieces recursively
- “Merge” these solutions to somehow obtain a solution of the original problem.
- Prototypical example: **merge sort**.

  ![Diagram](image)

- “If only I could merge two half-sized solutions, I’d be finished!”
Counting Inversions

• By modifying merge sort, we can easily count inversions in a length-n array in $O(n \log n)$ time.
• Idea: count inversions as they are corrected.
  – Inversions solely in first or second halves are automatically counted by recursion!
  – Inversions crossing between first and second halves are counted during merge (whenever we take an element from the second half).

Example: The Skyline Problem
(A Classic Contest Problem)

• Give N rectangular buildings sharing a common base, find the area of the “skyline” they form.

• “All you need is merge” (the lesser-known hit Beatles song)
Partitioning Versus Merging

- Spend some work “partitioning” a large problem into independent sub-problems (usually two of them), which are then solved recursively.
- Prototypical example: quicksort

Bi-Chromatic Matching

- Given N red points and N blue points in the plane, match them up with non-crossing line segments…
• It’s always possible to find a line that equally subdivides both the ham and the cheese!
Bi-Chromatic Matching

- Given $N$ red points and $N$ blue points in the plane, match them up with non-crossing line segments…

- Easy to solve recursively after partitioning with a ham sandwich cut!

The Firing Squad Problem

- $N$ parallel processors hooked together in a line:

- Each processor doesn’t know $N$, and only has a constant number of bits of memory (so it can’t even count to $N$).
- Processors synchronized to a global clock. In each time step, a processor can:
  - Perform some simple calculation.
  - Exchange messages with its neighbors.
- At some point in time, we give the leftmost processor a “ready!” message.
- Sometime in the future, we want all the processors to enter the same state “fire!” all in the same time step.
Longest Line of Sight

- Given the heights of $N$ individuals standing in a line.
- **Goal**: find the length of the longest line of sight (difference in indices between two people between whom everyone else is strictly shorter).
- Divide and conquer gives us an $O(n \log n)$ solution; there’s even an easy $O(n)$ solution using Cartesian trees.

Points – IOI’06

- Given $N$ red and green points in the plane (top two are green, bottom two red).
- Connect with non-crossing segments:
Points – IOI’06

• First decompose into two triangles by adding a diagonal from v1 to v3.
• Then recursively subdivide triangles, carefully…

Finding a Local Maximum

• Given an N x N grid of elevations, each distinct.
• Find some grid point whose height is larger than all 4 neighbors (or fewer neighbors, if on boundary).
• N = 1,000,000
• Goal: minimize the number of queries required to find a local maximum.
Separators in Trees and Polygons

- Every tree has a “separator” node whose removal breaks the tree into pieces each of size at most \( n/2 \) nodes.
- Every polygon has a separator “diagonal” whose addition breaks the polygon into two polygons of size at most \( 2n/3 \).

5. Binary Search on Answer

- Problem A: Find the answer to problem P.
- Problem B: Given \( X \), is this answer \( \leq X \).
- Oftentimes B is much easier than A, so we can solve A by binary searching on the answer and applying a solution of B.

- Example: Given an integer \( N \), compute the integer part of its square root.
Example: Bottleneck Shortest Paths And Spanning Trees

• Given a graph with edge weights, find:
  – A path from x to y whose maximum weight edge is as small as possible
  – A path from x to y whose minimum weight edge is as large as possible (i.e., if weight indicates capacity of a pipe, a path along which we can pump the most fluid).
  – A spanning tree of our graph whose maximum weight edge is as small as possible.
  – A spanning tree whose minimum weight edge is as large as possible.
• We can solve “Problem B” for all of these just using depth-first search…

Optimizing a Ratio Objective

• Given an array A[1…100,000] as input, which problem below sounds harder?
  – Find a subarray A[i…j] whose length is between 100 and 200 having maximum sum.
  – Find a subarray A[i…j] whose length is between 100 and 200 having maximum average.
Optimizing a Ratio Objective

• Given an array $A[1…100,000]$ as input, which problem below sounds harder?
  – Find a subarray $A[i…j]$ whose length is between 100 and 200 having maximum sum.
  – Find a subarray $A[i…j]$ whose length is between 100 and 200 having maximum average.
• The first can be solved in linear time with a few sliding window tricks, and the second can then be solved by binary search, treating the first problem as “problem B”!

Another Nice Example: Optimal Weighted Subsets

• You have just taken 1000 quizzes (ouch!)
• You earned $P_1$ … $P_{1000}$ points on each of them, out of a total of $T_1$ … $T_{1000}$ possible points. (so your scores are $100P_1 / T_1$, $100P_2 / T_2$, etc.)
• You get to drop 50 quizzes, leaving you with a set $S$ of 950 remaining. What is the maximum possible score you can achieve in this manner:

$$\max_{|S|=950} \sum_{i \in S} \frac{P_i}{\sum_{i \in S} T_i}$$

• There’s a nice geometric outlook on this problem!
6. Guess Partial Solution

- Knowing even part of a solution often helps a lot!
- **The classic “lights out” puzzle:**
  - Given an 8 x 8 binary grid as input.
  - If you select a square, you flip it plus its 4 neighbors.
  - Goal is to transform the grid into the grid of all ones.

```
0 0 1 1 0 1 0 1
0 1 1 0 1 1 1 0
0 1 1 1 0 0 0 1
1 1 1 0 1 0 1 0
1 0 0 1 0 0 1 1
1 1 0 1 0 1 1 0
0 1 0 0 1 0 0 0
```

Knowing Even Part of the Solution Helps...

- What about “lights out” on a torus – i.e., a grid that wraps around from top to bottom and from left to right?
- How would you find a maximum-sum submatrix (induced by a subset of rows and columns) of an 20 x 20 matrix?

```
0 0 1 1 0 1 0 1
0 1 1 0 1 1 1 0
0 1 1 1 0 0 0 1
1 1 1 0 1 0 1 0
1 0 0 1 0 0 1 1
1 1 0 1 0 1 1 0
0 1 0 0 1 0 0 0
-3 1 -5 7 -2 -1 0 2
-8 -3 -1 4 2 3 -1 -1
-1 5 1 0 0 2 -4 -8
-5 -2 0 -1 4 1 -2 -1
-1 -1 3 6 -5 -3 -2 -1
-3 4 8 2 -5 -2 -7 4
0 0 0 -1 2 -3 1 3
```
Can you strike the 8 ball, directly in its center, with at most 2 bounces?
• **Pattern matching:** given a long text $T$ and a short pattern $P$, find all occurrences of $P$ in $T$.
  - E.g., find all occurrences of ABC in BABCBCABCCA.
• This is a well-studied problem, and it can be solved in $O(\text{length}(T))$ time.
• However, now suppose you are given two strings $A$ and $B$, where $A$ is obtained from $B$ by a left cyclic shift of length $x$. How can you compute $x$?
  - E.g., $A = \text{“ABCCDEF”}$. $B = \text{“CDEFABC”}$. $x = 3$. 
Pattern matching: given a long text $T$ and a short pattern $P$, find all occurrences of $P$ in $T$.
- E.g., find all occurrences of ABC in BABCBCBABCCA.

This is a well-studied problem, and it can be solved in $O(\text{length}(T))$ time.

However, now suppose you are given two strings $A$ and $B$, where $A$ is obtained from $B$ by a left cyclic shift of length $x$. How can you compute $x$?
- E.g., $A = \text{“ABCCDEF”}$. $B = \text{“CDEFABC”}$. $x = 3$.
- Search for $B$ in $AA$!... ABCCDEFABCCDEF.

Cyclic Shift Detection

Ants on a Stick

100 ants are placed at various locations on a one-meter stick. Some start facing left, others facing right.
- Each ant moves at 1 cm / sec.
- When ants collide, they both reverse direction.
- How long until all the ants fall of the ends of the stick?
Birthday Cake

- Bessie’s age, x, is an integer in the range 000000…999999 (in cow years, of course)
- Bessie notices that if she flips her birthday cake around and reads her age backwards by mistake, it appears that she is younger.
- How many values of x are there satisfying this property?

Saving Time While Counting…

- How many solutions are there for the “8 queens” problem?
Exhaustive Enumeration...

• For each location of the first queen (say, in the top row), fix a queen there and recursively count the number of ways to complete the solution...

Exhaustive Enumeration...

• For each location of the first queen (say, in the top row), fix a queen there and recursively count the number of ways to complete the solution...

Unnecessary to check!
Rafting

• What is the radius of the largest circular raft we can fit through all the circular rocks in this river?

8. Sort + Scan

• Often, it’s much easier to solve a problem after first sorting the input as a preprocessing step.

• **Trivial examples**: Given an unordered array,
  – Find the median element
  – Determine if all elements are distinct
  – Count the number of distinct elements
  – Find the most-frequently-occurring element
  – Find the smallest gap $|X - Y|$, where $X$ and $Y$ belong to the array.
Sort First, Ask Questions Later

- **Example:** You are given N points in the plane. A point is “surrounded” if there are other points directly above, below, left, and right of it. Count the number of surrounded points.

- **Example:** Given N points in the plane, find the area of the largest rectangle you can form using four of them (not necessarily axially aligned!)

Example: The Skyline Problem

- Give N rectangular buildings sharing a common base, find the area of the “skyline” they form.

- Here we use a heap, a simple data structure supporting **insert**, **delete**, and **find-max** all in O(log N) time.
Generalization: Area of a Union of Rectangles

- Give N rectangles, compute the total area of the plane that they cover.

- Same general idea, except here we use a fancier augmented binary search tree, for example a segment tree.

Counting Intersection Points

- Initially sort all endpoints by x coordinate.
- Sweep a vertical line from left to right, stopping at each endpoint.
- Maintain a balanced BST, T, with set of y coordinates for horizontal segments currently intersecting the sweep line.
- When we encounter the left/right endpoint of a horizontal segment, perform an insert/delete in T.
- When we encounter a vertical segment, do a range query in T.
Paranoid Cows

• N cows are standing at various points in the 2D plane.
• Due to the way the sun is shining, all cows can only see other cows in their upper-right quadrants.
• A cow is paranoid if she doesn’t see any other cows.
• Please compute the number of paranoid cows.

• What about:
  – The 3D case?
  – Alternative approaches, such as divide & conquer?

Finding the Median…

• X is a median of a set of N numbers if \( \leq N/2 \) of our numbers are smaller than X, and \( \leq N/2 \) of our numbers are bigger than X.
• One easy way to find the median of N numbers is to sort them and take the middle number. (although there actually are faster methods!)
• Note that we can easily test if a candidate value of X is “too high” or “too low”…
• **Challenging problem:** You are given as input N numbers, which define \((N^2/2)\) pairwise differences. What is the median of these differences?
9. Knowing the Future Helps

- Most common solution approaches (at least on contests): divide and conquer, sweep lines.
- Good way to “dress up” other problems like graph problems.
- Fancy data structures often involved.
- Beware floating point & special cases!
- Useful trick in 2D: sign of cross product.
- Useful duality results:
  - Point-line duality
  - Planar graph duality (e.g., between cuts & paths, polygon triangulations & trees).

10. Geometry Tricks
N cows live at lattice points in a big city.
Since cows can only walk along the grid of city streets, the distance between two cows at \((a, b)\) and \((c, d)\) is given by \(|a - c| + |b - d|\).
Please compute the distance between the farthest pair of cows.

Trick: L₁ Norm → L∞ Norm
Trick: $L_1$ Norm $\rightarrow L_\infty$ Norm

- Consider two points $(a, b)$ and $(c, d)$.
  - $L_1$ distance between the two: $|a - c| + |b - d|$.
  - $L_\infty$ distance: $\max(|a - c|, |b - d|)$.

- Computing the diameter of a set of $N$ points using the $L_\infty$ distance is easy in $O(N)$ time!
- By applying the transformation $(x, y) \rightarrow (x+y, x-y)$, we can transform an $L_1$ problem into an $L_\infty$ problem!

Cow Clusters [Camp ’04]

- $N$ cows in the 2D plane.
- Partition into two groups, so as to minimize the $L_1$ diameter of the largest group.

- This problem is much easier after converting to an $L_\infty$ problem as a preprocessing step!
11. Parity

- You are given a sequence of N numbers.
  4 1 2 7 5 0 8 6 3 2
- Players A and B take turns removing numbers from either endpoint.
- The game stops when all the numbers are taken, and the winner is the player with the largest sum.
- If N is even, player A can always win!

A Simple Game

- You are given a sequence of N numbers.
  4 1 2 7 5 0 8 6 3 2
- Players A and B take turns removing numbers from either endpoint.
- The game stops when all the numbers are taken, and the winner is the player with the largest sum.
- If N is even, player A can always win!
  - Choose the color with maximum sum, and always take the endpoint of that color.
The Game of Nim

- Start with N piles containing $x_1 \ldots x_N$ stones.
- Players A and B take turns.
- Each term, you can remove any number of stones from a single pile of your choice.
- Whoever removes the last stone wins.
- **Winning strategy if you move first:** ensure that $\text{xor}(x_1 \ldots x_N) = 0$ after each move.

XOR Tricks

- XOR is a wonderful operation, since applying a second time cancels out the first application:
  $$(a \text{ XOR } b) \text{ XOR } b = a.$$ 
- Swap two integers without using a temporary variable:
  $$a = a \text{^} b; \quad b = a \text{^} b; \quad a = a \text{^} b;$$
- **Simple problem:** given an integer array $A[1 \ldots N]$, all the numbers in $A$ except one occur an even number of times. Find the number appearing an odd number of times.
XOR Tricks

• XOR is a wonderful operation, since applying a second time cancels out the first application:
  \[(a \text{ XOR } b) \text{ XOR } b = a.\]
• Swap two integers without using a temporary variable:
  \[a = a^\oplus b; \quad b = a^\oplus b; \quad a = a^\oplus b;\]
• Simple problem: given an integer array \([1…N]\), all the numbers in \(A\) except one occur an even number of times. Find the number appearing an odd number of times.

Point-In-Polygon Testing

• What is the easiest way to test if a point lies inside a non-convex polygon?
Point-In-Polygon Testing

• What is the easiest way to test if a point lies inside a non-convex polygon?

  ![Point-In-Polygon Diagram]

• Cast a ray in any direction, counting the parity of the number of crossings with the polygon!

Snack Snatching

• Suppose you are a graduate student looking at a pile of N free snacks.
• You and your opponent (also a graduate student) take turns removing snacks.
• The first player to move cannot take all the snacks (that would just be greedy!)
• Whoever removes the last snack wins.
• You cannot remove more snacks than your opponent did on his last move.
• For what values of N can you guarantee winning if you move first?

- Sometimes the simplest solution is just to use brute force – do the math to figure out if this is feasible.
- **Example:** Given 20 integers, can you partition them into two sets with equal sums?
  (by the way, how do you easily enumerate all possible subsets of 20 integers?)

---

DFS + Iterative Deepening

- We search essentially be performing DFS or BFS on our search tree of solutions.
- **Common issue:** we don’t know how deep we need to search to obtain a good solution (example: sliding puzzle).
  - DFS might search too deeply
  - BFS finds a shallow solution but might take too much memory, due to large branching factor.
  - So we sometimes use DFS with “iterative deepening…” to simulate BFS with DFS.
Search Smartly: Tree Pruning

- **Big Idea:** Prune away bad branches of the search tree as soon as we realize they aren’t going to lead to a solution better than the best one we’ve found so far.
  - E.g., if our current depth + the “manhattan distance to destination” for all squares exceeds the depth of our current best solution, prune the search immediately and back up.
  - See also “branch & bound” and “A* search”.

- **Big Ideas:**
  - Most constrained choices first
  - Greedy choices first
  - Exploit symmetry

13. Neighborhood Search

- Another general heuristic approach for obtaining good solutions to hard problems.
  - Example: “2-swap” neighborhoods for TSP:

```
1 2 3 4 5 6 7
```

- Multiple starting points; take best solution.
Problem: Given an array $A[1…100,000]$, find a subarray $A[i…j]$ whose average value is as large as possible.

Solution Structure Matters

Problem: Given an array $A[1…100,000]$, find a subarray $A[i…j]$ of length at least 100 whose average value is as large as possible.
Solution Structure Matters

- **Problem**: Given an array $A[1...100,000]$, find a subarray $A[i...j]$ of length at least 100 whose average value is as large as possible.

- Useful result: Let $a_1$, $a_2$ and $b_1$, $b_2$ be positive numbers such that $a_1 / b_1 < a_2 / b_2$. Then,
  \[ a_1 / b_1 < (a_1+a_2) / (b_1+b_2) < a_2 / b_2. \]

Best Case: No Searching Required (Closed-Form Math Solution...)

- Professor Snarf lives on a long street on which the houses are numbered sequentially 1…N.
- From his house (number $X$), he notices a curious phenomenon: if he adds up the house numbers on his left and those on his right, he gets the same sum!
- Please determine (as quickly as possible) all values of $N$ and $X$ for which this is possible...
- We can answer this question mathematically by solving a Pell Equation.
Simplify # of Cases to Check (Know Your Solutions!)

• Jane has just bought 3 rectangular Persian carpets, whose dimensions are given to you as input.
• She would like to arrange them in a non-overlapping fashion in a square room.
• Please compute the minimum side length of the room that can possibly accommodate all three carpets…

15. Build a Graph

• N = 16 cities
• Each city has a red outgoing road and a blue outgoing road (all roads are 1-way!)
  – You can therefore give someone directions by specifying a string like RBRRBB.
• Find the shortest string of directions you can tell someone that will bring them to city #1, irrespective of where they start.
DAGs are Good Ways to Visualize Dynamic Programs

- Maximum Length/Value Interval Packing
- Longest Increasing Subsequence

16. \( \sqrt{N} \)

**Problem:** maintain an array \( A[1..N] \) so that we can support two operations:
- Modify \( A[i] \)
- Compute the sum of a range \( A[i..j] \).
16. $\sqrt{N}$

- **Problem**: maintain an array $A[1..N]$ so that we can support two operations:
  - Modify $A[i]$
  - Compute the sum of a range $A[i..j]$.
- Can achieve both in $O(\log n)$ time very easily with a binary index tree (discussed later).
- For now, observe that another simple solution is just to do a 2-level hierarchical decomposition of the problem.

“Mo’s Algorithm”

- Goal: answer a number of queries (given in advance), each over some range $i…j$.
  - Further, suppose that if we move $i$ or $j$ by one position, we can update the answer to the query in only $O(1)$ time…
- How to order the queries so we can answer them all quickly?
  - Goal: $O((n + \#queries)\sqrt{n})$ total time.
“Mo’s Algorithm”

• Bucket queries into $\sqrt{n}$ buckets by left endpoint; process queries in each bucket in sorted order by right endpoint.

17. Prefixes

• Quick: preprocess an array $A[1…n]$ so that one can compute any range sum $A[i…j]$ instantly…
Prefixes and Suffixes

- Window = Prefix – Prefix
  - Example: Prefix hashes → window hashes
  - Example: Count the number of “special” integers in some large range [a, b].
- Two-Sided Answer = Prefix + Suffix
  - Example: Find the longest bitonic (increasing then decreasing) subsequence.

Prefixes

- Quick: preprocess an array A[1…n] so that one can compute any range sum A[i…j] instantly…
- Given an length-n binary string, find the longest substring that is “balanced” (same number of 0s as 1s).
• Quick: preprocess an array $A[1...n]$ so that one can compute any range sum $A[i...j]$ instantly...

• Given an length-$n$ binary string, find the longest substring that is “balanced” (same number of 0s as 1s).

• Given an array $A[1...n]$, quickly count the number of subarrays $A[i...j]$ that have nonnegative sums.

• Given stock prices on $N$ days, what is the max profit you can make by buying one share on some day $i$ and selling it on some later day $j > i$?
“Dual” Representations of Sequences

Prefix sums
$A_1, A_1+A_2, A_1+A_2+A_3, \ldots$

Array of numbers
$A_1, A_2, A_3, \ldots$

Successive Differences
$A_1, A_2-A_1, A_3-A_2, \ldots$

- Prefix sums allow for fast range sum queries
- Successive differences allow for fast range updates (e.g., add 5 to all elements in range).

- Prefix sums allow for fast range sum queries
- Successive differences allow for fast range updates (e.g., add 5 to all elements in range).
- Given stock prices on N days, what is the max profit you can make by buying one share on some day $i$ and selling it on some later day $j > i$?
Prefix Practice

• Count the number of squares in the grid that are part of some $K \times K$ square of all 1s...

```
0 0 1 1 0 1 0 1
0 1 1 1 1 1 0 0
0 1 1 1 0 0 0 1
1 1 1 1 1 0 1 0
1 0 1 1 1 0 1 1
1 1 0 1 0 1 0 0
0 1 0 0 1 0 0 0
```

Fixed-Length / Sliding Window Range Minimum Queries

• Quick: preprocess an array $A[1…n]$ so that one can compute the minimum of any length-$L$ subarray...
18. Compute NOT(Answer)

- In an N-vertex convex polygon (with N at most 1000), there is a line segment joining every pair of vertices that is colored either orange or purple.
- Please count the number of monochromatic triangles.

Simplicial Depth

- Given N points in the 2D plane (no 3 collinear), count the number of triangles induced by the point set that contain a specified point p.
Perhaps My Favorite Problem: Apple Catching

- You are told when and where $N$ apples will fall on the number line.
- Assuming everyone can move at a speed of 1 unit per second, how many individuals do you need to recruit to catch all the apples?

Today’s Lab Exercise: Slope Selection

- $N$ points in the plane (no 3 collinear) determine $\binom{N}{2}$ lines, and hence $\binom{N}{2}$ slopes.
- Find the $k^{th}$ largest such slope (e.g., for $k = 1$, we want the minimum slope).

- Solution involves:
  - Transform points into lines with slope-intercept duality
  - Binary search by guessing answer
  - Counting inversions