# Solution Ideas

* Dynamic Programming

– Drop a parameter, recover from others

– Swap answer and a parameter

– Parsing CFGs: CYK Algorithm

– Optimizations

∗ Convex hull optimization

· dp[*i*] = minj < i {dp[*j*] + b[*j*]×a[*i*]}

· b[*j*] ≥ b[*j* + 1]

· optionally a[*i*] ≤ a[*i* + 1]

· O(n^2 ) to O(n)

∗ Divide and conquer optimization

· dp[*i*][*j*] = mink < j {dp[*i* − 1][k] + C[k][*j*]}

· A[*i*][*j*] ≤ A[*i*][*j* + 1]

· O(kn^2 ) to O(knlogn)

· sufficient:C[a][c]+C[b][d]≤C[a][d]+C[b][c],

a ≤ b ≤ c ≤ d (QI)

∗ Knuth optimization

· dp[*i*][*j*] = mini<k<j {dp[*i*][k]+dp[k][*j*]+C[*i*][*j*]}

· A[*i*][*j* − 1] ≤ A[*i*][*j*] ≤ A[*i* + 1][*j*]

· O(n^3 ) to O(n^2 )

· sufficient:QI and C[b][c] ≤ C[a][d],a≤b≤c≤d

• Greedy

• Randomized

• Optimizations

– Use bitset (/64)

– Switch order of loops (cache locality)

• Process queries offline

– Mo’s algorithm

• Square-root decomposition

• Precomputation

• Efficient simulation

– Mo’s algorithm

– Sqrt decomposition

– Store 2^k jump pointers

• Data structure techniques

– Sqrt buckets

– Store 2^k jump pointers

– 2^k merging trick

• Counting

– Inclusion-exclusion principle

– Generating functions

• Graphs

– Can we model the problem as a graph?

– Can we use any properties of the graph?

– Strongly connected components

– Cycles (or odd cycles)

– Bipartite (no odd cycles)

∗ Bipartite matching

∗ Hall’s marriage theorem

∗ Stable Marriage

– Cut vertex/bridge

– Biconnected components

– Degrees of vertices (odd/even)

– Trees

∗ Heavy-light decomposition

∗ Centroid decomposition

∗ Least common ancestor

∗ Centers of the tree

– Eulerian path/circuit

– Chinese postman problem

– Topological sort

– (Min-Cost) Max Flow

– Min Cut

∗ Maximum Density Subgraph

– Huffman Coding

– Min-Cost Arborescence

– Steiner Tree

– Kirchoff’s matrix tree theorem

– Prüfer sequences

– Lovász Toggle

– Look at the DFS tree (which has no cross-edges)

• Mathematics

– Is the function multiplicative?

– Look for a pattern

– Permutations

∗ Consider the cycles of the permutation

– Functions

∗ Sum of piecewise-linear functions is a

piecewise-linear function

∗ Sum of convex(concave)functions is convex (concave)

– Modular arithmetic

∗ Chinese Remainder Theorem

∗ Linear Congruence

– Sieve

– System of linear equations

– Values to big to represent?

∗ Compute using the logarithm

∗ Divide everything by some large value

– Linear programming

∗ Is the dual problem easier to solve?

• Logic

– 2-SAT

– XOR-SAT (Gauss elimination or Bipartite matching)

• Meet in the middle

• Only work with the smaller half (log(n))

• Strings

– Trie (maybe over something weird, like bits)

– Suffix array

– Suffix automaton (+DP?)

– Aho-Corasick

– eerTree

– Work with S + S

• Hashing

• Euler tour, tree to array

• Segment trees

– Lazy propagation

– Persistent

– Implicit

– Segment tree of X

• Geometry

– Minkowski sum (of convex sets)

– Rotating calipers

– Sweep line (horizontally or vertically?)

– Sweep angle

– Convex hull

• Fix a parameter (possibly the answer).

• Are there few distinct values?

• Binary search

• Sliding Window (+ Monotonic Queue)

• Fast Fourier Transform

• Exact Cover (+ Algorithm X)

• Cycle-Finding

• What is the smallest set of values that identify the solution?

The cycle structure of the permutation? The powers of primes

in the factorization?

• Look at the complement problem

– Minimize something instead of maximizing

• Immediately enforce necessary conditions. (All values greater

than 0? Initialize them all to 1)

• Add large constant to negative numbers to make them positive

• Counting/Bucket sort

• Buscar cota extrema.

• Iterar por todas las máscaras de N. {j=n, where n > 0: j=n&(j-1)}

# Debugging Tips

• Stack overflow? Recursive DFS on tree that is actually a long

path?

• Rounding negative numbers?

• Double

• Wrong Answer?

– Quitar el freopen,

– no mezclar cin con scanf

– Ver si hay que imprimir fin de linea

– Leer nuevamente el problema.

– Ver si es multiples casos, repetir el mismo caso varias

veces.

– long long

– Posibles Casos:

∗ n = 0,n = −1,n = 1,n = 2^31 − 1 or n = −2^31

∗ La lista esta vacía o con un solo elemento

∗ n is even, n is odd

∗ El Grafo esta vacio o contiene un solo vertice

∗ El Grafo es un multigrafo (lazo o multiple aristas)

∗ El Polygono es convexo o no

–Hay condición inicial para los casos pequeños

–Estas utilizando el algoritmo correcto

– Explique su solución a alguien

– ¿Usa usted algunas funciones que usted completamente no

comprende? ¿Puede que STL funcione?

– ¿Puede que usted (o alguien más) debiera reescribir la

solución?

• Run-Time Error?

–Verificar el tamaño de los arreglos

–División por 0