Competitive Programming UTEC - Summer 2020

Inicio: 6 de enero
Fin: 21 de febrero
Horarios:

- Lunes 17:00-19:00
- Miércoles 17:00-19:00
- Viernes 14:00-16:00

Información del taller

Este taller de 7 semanas busca introducir a los interesados en la programación competitiva a este deporte presentandoles las bases y los fundamentos básicos necesarios. Se enseñaran técnicas para resolver problemas usando los paradigmas de búsqueda completa (*fuerza bruta*) y divide y vencerás. Además, se trabajarán algunos conceptos y algoritmos de grafos como formas de recorrido, cálculo de caminos mínimos y encontrar mínimos árboles de expansión.

Tópicos del taller

- Fuerza bruta
- Divide y vencerás
- Recorrido de grafos (BFS & DFS)
- Single Source Shortest Path (SSSP)
- All Pair Shortest Path (APSP)
- Mínimo árbol de expansión (MST)

Plan semanal

Semana 1 (6, 8, 10 enero)

- **Sesión 1: Introducción**
  - ¿Qué es programación competitiva?
  - ¿Qué veremos en el curso?
  - Plataformas a usar
  - Estructura de los problemas
  - Algunos problemas que podremos resolver al finalizar el curso
  - **Lecturas recomendadas**
    - Training Camp Argentina 2019. Nivel Inicial. E/S + Intro
    - Competitive programming 3, capítulo 1 [1]
    - Errichto - How to test your solutions in Competitive Programming, on Linux

        - An awesome list for competitive programming

- **Sesión 2: Análisis asintótico**
    - Notación Big O
    - Complejidades comunes: constante, logarítmica, lineal, cuadrática, ...
    - Distintas medidas de performance: tiempo y memoria
    - Notación Omega y Theta
    - Algoritmos online y offline
    - Problemas
    - **Lecturas recomendadas**
        - Competitive Programmer's Handbook, capítulo 2 [2]
        - Learn Data Structures and Algorithms, sección Asymptotic analysis [3]

- **Sesión 3: Standard Template Library (STL)**
    - Input & Output (I/O)
    - Static Array
    - Estructuras de datos (vector, stack, queue, set, map)
    - Cadenas y texto (std::string)
    - Ordeneamiento (std::sort)
    - Objetos (struct)
    - Trucos de C++ comúnmente usados en programación competitiva
    - **Lecturas recomendadas**
        - Competitive Programmer's Handbook, capítulo 1, 3 y 4 [2]
        - Codeforces - Competitive C++ Manifesto: A Style Guide y sus referencias
        - Topcoder - Power up C++ with the Standard Template Library Part 1

Semana 2 (13, 15, 17 enero)

- **Sesión 4: Fuerza Bruta I**
    - 4 paradigmas de problem solving: fuerza bruta, divide y vencerás, greedy y programación dinámica
    - Ternas pitagóricas
    - Test de primalidad en $O(\sqrt{n})$
    - Criba de Eratóstenes
    - Los divisores de un número: calculo en $O(\sqrt{n})$ con fuerza bruta y en $O(n \log n)$ con criba
    - La cantidad de divisores de un número es $O(\sqrt[3]{n})$: problemas
    - **Lecturas recomendadas**
        - Competitive Programming 3, sección 3.2 y 5.5.1 [1]
        - Codeforces - How to come up with the solutions: techniques
        - Codeforces - Amount of divisors of N
        - Codeforces - Counting Divisors of a Number in $O(\sqrt[3]{n})$ [Tutorial]

- **Sesión 5: Repaso I**
    - Más problemas de los temas vistos en las sesiones 1-4

- **Sesión 6: Fuerza Bruta II**
    - Ejemplos de estrategias comúnes con fuerza bruta
        - Simulación por fuerza bruta
        - Aprovecha un constrain bajo

        * Fija la respuesta
        * Reduce las variables
    - Problemas
    - **Lecturas recomendadas**
        * Notas de fuerza bruta
        * PCUNI-2019 clase 05
        * Topcoder - Representation of integers and reals section 1
        * Topcoder - Representation of integers and reals section 2

Semana 3 (20, 22, 24 enero)

- **Sesión 7: Fuerza Bruta III**
    - Invariantes y monovariantes
    - Técnica de dos punteros
    - Recursión
    - Aritmética modular
    - Exponenciación binaria
    - Problemas
    - **Lecturas recomendadas**
        * [Tutorial] Invariants and Monovariants
        * Competitive Programmer's Handbook, capítulo 5, 8 y 21 [2]
        * Principles of Algorithmic Problem Solving, sección 15.6 y 15.7 [4]
        * Learn Data Structures and Algorithms, sección Basic Recursion [3]
        * E-maxx Binary Exponentiation [5]
- **Sesión 8: Fuerza Bruta IV**
    - Búsqueda por todas las permutaciones
    - Representación binaria de números
    - Operaciones sobre bits
    - Fuerza bruta usando máscara de bits
    - Problemas
    - **Lecturas recomendadas**
        * Competitive Programming 3, sección 2.1 y 2.2 [1]
        * Competitive Programmer's Handbook, capítulo 10 [2]
        * GPC-UNI Clase 4 [6]
        * GPC-UNI Clase 5 [6]
        * PCUNI-2019 clase 07
        * Principles of Algorithmic Problem Solving, capítulo 7 [4]
- **Sesión 9: Fuerza Bruta V**
    - El problema de las 8 reinas
    - Sudoku de 9×9
    - Buscando palabras en un pupiletras
    - Prunning
    - Problemas
    - **Lecturas recomendadas**
        * HackerEarth - Recursion and Backtracking

        * Competitive Programming 3, sección 3.2.2, 8.2.1 y 8.2.2 [1]
        * GeekForGeeks - Backtracking Algorithms

Semana 4 (27, 29, 31 enero)

- **Sesión 10: Repaso II**
    - Más problemas de los temas vistos en las sesiones 6-9
- **Sesión 11: Divide y vencerás I**
    - Binary search
        * Encontrar un elemento en un array ordenado
        * Lower bound y upper bound
        * Búsqueda binaria sobre la respuesta
        * Búsqueda binaria sobre números reales
        * Búsqueda binaria generalizada
        * Problemas
    - Ternary search
        * Encontrar un mínimo o máximo en una función unimodal
        * Problemas
    - Errores comunes de implementación
    - **Lecturas recomendadas**
        * Errichto - Binary Search lecture (C++ and Python)
        * Topcoder - Binary search
        * Principles of Algorithmic Problem Solving, sección 10.3 [4]
        * Competitive Programmer's Handbook, sección 3.3 [2]
        * HackerEarth - Searching [Tutorial]
        * E-maxx Ternary search [5]
        * Topcoder - Binary stride a variant on binary search
- **Sesión 12: Divide y vencerás II**
    - El teorema maestro
    - Merge sort y quick sort
    - Construcciones inductivas
    - Problemas
    - **Lecturas recomendadas**
        * Principles of Algorithmic Problem Solving, sección 10.1 y 10.2 [4]
        * PCUNI-2019 clase 16

Semana 5 (3, 5, 7 febrero)

- **Sesión 13: Grafos I**
    - Terminología
    - Representación de grafos
    - DFS
        * Clasificación de aristas en un grafo
        * Floodfill: componentes conexas
        * Determinar si un grafo es acíclico
        * Diámetro de un árbol

    - Problemas
  - **Lecturas recomendadas**
    - Competitive Programmer's Handbook, capítulo 11, 12 y 14 [2]
    - E-maxx DFS [5]
    - E-maxx Checking a graph for acyclicity and finding a cycle in $O(M)$ [5]
    - VisuaAlgo - DFS, BFS

- **Sesión 14: Grafos II**
  - BFS
    - Mínimo camino en grafos de peso 0 o 1: cálculo y reconstrucción de caminos
    - Determinar si un grafo es bipartito
    - Problemas
  - Dijkstra
    - Mínimo camino en grafos de peso positivo: cálculo y reconstrucción de caminos
    - Problemas
  - **Lecturas recomendadas**
    - Competitive Programmer's Handbook, capítulo 13 [2]
    - E-maxx BFS [5]
    - E-maxx Dijkstra [5]
    - Principles of Algorithmic Problem Solving, capítulo 12 [4]

- **Sesión 15: Repaso III**
  - Más problemas de los temas vistos en las sesiones 11-14

Semana 6 (10, 12, 14 febrero)

- **Sesión 16: DSU**
  - Problema motivacional de DSU
  - Implementación ingenua
  - Optimización de compresión de caminos
  - Optimización de unión por tamaño o rango
  - Encontrando componentes conexas con DSU
  - Almacenando información extra en un DSU
  - Problemas
  - **Lecturas recomendadas**
    - E-maxx Disjoint Set Union [5]
    - Codeforces - Understanding Disjoint Set Structures
    - Habr - Disjoint sets and the mysterious function of Ackermann. Traducción: Yandex Translate

- **Sesión 17: Grafos III**
  - Mínimo árbol de expansión
    - Algoritmo de Prim
    - Algoritmo de Kruskal
  - Máximo árbol de expansión
  - Segundo mejor mínimo árbol de expansión
  - Problemas
  - **Lecturas recomendadas**
    - Competitive Programmer's Handbook, capítulo 15 [2]

        * E-maxx Minimum spanning tree - Kruskal with Disjoint Set Union [5]
        * E-maxx Minimum spanning tree - Prim's algorithm [5]
        * E-maxx Second Best Minimum Spanning Tree [5]
        * Learn Data Structures and Algorithms, sección Minimum Spanning Tree [3]

- **Sesión 18: Descanso**

Semana 7 (17, 19, 21 febrero)

- **Sesión 19: Grafos IV**
  - Camino mínimo en un grafo
    * Algoritmo de Bellmand-Ford
    * Algoritmo de Floyd-Warshall
  - Determinar la existencia de ciclos negativos
  - Exponenciación binaria de matrices
  - Número de caminos de tamaño mínimo
  - **Lecturas recomendadas**
    * E-maxx Bellman-Ford Algorithm [5]
    * E-maxx Finding a negative cycle in the graph [5]
    * E-maxx Floyd-Warshall Algorithm [5]
    * E-maxx Number of paths of fixed length / Shortest paths of fixed length [5]
    * Competitive Programmer's Handbook, capítulo 23 [2]
- **Sesión 20: Repaso IV**
  - Más problemas de los temas vistos en las sesiones 16-19
- **Sesión 21: Competencia de Cierre / Compartir**

Referencias

[1] Steven Halim. *Competitive Programming 3. The New Lower Bound of Programming Contests*. Paperback - Lulu, 2013.

[2] Antti Laaksonen. Competitive programmer's handbook. `https://jadi.net/wp-content/uploads/2017/07/competetive-programmers-handbook.pdf`, 2017. [Online; accedido 12-diciembre-2019].

[3] Codechef. Codechef certified data structure & algorithms programme. `https://www.codechef.com/certification/data-structures-and-algorithms/prepare`, 2019. [Online; accedido 12-diciembre-2019].

[4] Johan Sannemo. Principles of algorithmic problem solving. `https://www.csc.kth.se/~jsannemo/slask/main.pdf`, 2018. [Online; accedido 12-diciembre-2019].

[5] Max Ivanov. E-maxx algorithms. `https://cp-algorithms.com/`, 2012. [Online; accedido 12-diciembre-2019].

[6] GPC-UNI. Programación competitiva uni. `https://github.com/GPC-UNI/Programacion-Competitiva`, 2018. [Online; accedido 12-diciembre-2019].