Temario de programación competitiva

José Leonidas García Gonzales

2018

Descripción

Algunos temas nivel básico-intermedio de programación competitiva. Los temas señalados están diseñados para ser llevados en 1 o 2 semanas dependiendo de la dificultad y extensión del tema.

Índice

1. Familiarizandose con C++ **3**

2. Standard Template Library I **3**

3. Standard Template Library II **4**

4. Búsqueda Completa (a.k.a Fuerza bruta) **4**

5. Repaso y contest por equipos **4**

6. Manipulación de bits **4**

7. Divide y vencerás **5**

8. Recursividad **5**

9. Backtracking I **6**

10.Repaso y contest por equipos **6**

11.Grafos I **6**

12.Grafos II **6**

13.Algoritmos golosos (a.k.a Greedy) **7**

14.Programación Dinámica I (a.k.a *DP*) **7**

15.Programación Dinámica II **8**

16.Repaso y contest por equipos **8**

17.Teoría de números I **8**

18.Teoría de números II **9**

19.Backtracking II **9**

20.Teoría de juegos combinatorios I **9**

21.Teoría de juegos combinatorios II **10**

22.Repaso y contest por equipos **10**

23.Combinatoria I **10**

24.Disjoint Set Union (a.k.a DSU) **11**

25.Fenwick Tree (a.k.a BIT) **11**

26.Segment Tree **11**

27.Repaso y contest por equipos **11**

1. Familiarizandose con C++

(I) Introducción a la programación competitiva

(II) Hola mundo

(III) Tipos de datos primitivos

(IV) Operaciones aritméticas entre datos primitivos

(V) Estructuras de control y anidamiento de condicionales

(VI) Estructuras de repetición

(VII) Funciones y *variable scopes*

(VIII) Namespace

(IX) Funciones de entrada y salida estándar

(X) Vectores

(XI) String

(XII) Funciones built-ins importantes

(XIII) Problemas Ad Hoc

2. Standard Template Library I

(I) Complejidades

(II) Vector

(III) Queue

(IV) Stack

(V) Deque

(VI) List

(VII) Set

(VIII) Map

(IX) Priority queue

(X) Unordered set

(XI) Unordered map

(XII) Multiset

(XIII) Multimap

3. Standard Template Library II

(I) Struct

(II) Class

(III) Templates

(IV) Paso de valores por asignación y referencia

(V) Const

(VI) Operadores

(VII) Built-in sort

(VIII) Repasando funciones de entrada y salida

a) cin y cout
b) scanf, sscanf, getchar, gets
c) print, sprintf, puts, putchar
d) getline
e) String Streams
f) strtok
g) Fast I/O

4. Búsqueda Completa (a.k.a Fuerza bruta)

(I) Problemas de escaneo lineal

(II) Problemas de dos loops anidados

(III) Técnicas de fijar parámetros

(IV) Propiedades matemáticas más usadas

(V) Problemas de más de dos loops anidados

(VI) Análisis de complejidades según el espacio de búsqueda

5. Repaso y contest por equipos

Repasar todos los temas anteriores. Despejar dudas de los problemas. Formar equipos de tres de manera aleatoria y dar un contest.

6. Manipulación de bits

(I) Representación binaria de números

(II) & (and)

(III) | (or)

(IV) ! (not)

(V) ^ (xor)

(VI) Corrimiento de bits

(VII) Chequear el estado del k-simo bit

(VIII) Prender, apagar y alternar el estado del k-simo bit

(IX) *Two's complement*

(X) Representación en bits de números negativos

(XI) Bitset

(XII) Usando máscara de bits para recorrer estados

(XIII) Funciones built-ins importantes

a) __builtin_popcount
b) __builtin_clz
c) __builtin_ctz

(XIV) Usando máscara de bits para recorrer todos los subjuntos de un conjunto

7. Divide y vencerás

(I) El teorema maestro

(II) Idea general del paradigma de divide y vencerás

(III) Merge sort

(IV) Búsqueda binaria

(V) Busqueda ternaria

(VI) *Golden section search*

8. Recursividad

(I) Principio de inducción matemática

(II) Principio de buen orden

(III) Funciones recursivas

(IV) Problemas conocidos (Fibonacci, Torres de Hanoi, ...)

(V) Encontrando formas cerradas de algunas funciones recursivas

(VI) Estrategias de planteamiento recursivo de soluciones

(VII) Fractales

(VIII) Recursión en Python

9. Backtracking I

(I) El problema de las 8 reinas

(II) Repaso de variables globales y variables locales

(III) El problema de hallar todas las permutaciones de un array

(IV) Usando *next_permutation* y *prev_permutation* en C++

(V) Usando el módulo *itertools* en Python

(VI) Entendiendo el árbol de expansión implícito generado al usar backtracking

(VII) *Heavy pruning*

10. Repaso y contest por equipos

Repasar todos los temas anteriores. Despejar dudas de los problemas. Formar equipos de tres por afinidad y dar un contest.

11. Grafos I

(I) Definiciones y propiedades importantes

(II) Formas de representar grafos

(III) BFS

a) Idea intuitiva

b) Coloración de nodos

c) Ejemplos y propiedades

(IV) DFS

a) Idea intuitiva

b) Coloración de nodos

c) Ejemplos y propiedades

12. Grafos II

1. *Single Source Shortest Paths (SSSP)*

   a) BFS

      1) Demostración de que el BFS resuelve el SSSP con ciertas restricciones
      2) Encontrando mínimas distancias desde un nodo en un grafo sin pesos o con peso constante
      3) Reconstrucción de caminos

   b) Dijkstra

      1) Descripción y restricciones del algoritmo de Dijkstra
      2) Demostración de que el algoritmo de Dijkstra resuelve el *SSSP* con las restricciones dadas
      3) Implementación de Dijkstra usando *priority_queue*
      4) Implementación de Dijkstra usando *set*

   c) Bellman Ford

1) Descripción y restricciones del algoritm de Bellman Ford
2) Demostración de que el algoritmo de Bellman Ford resuelve el *SSSP* con las restricciones dadas
3) Implementacion del algoritmo de Bellman Ford
4) Encontrando ciclos negativos

d) *Shortest Path Faster Algorithm (SPFA)*

1) Descripción y restricciones del algoritmo del *SPFA*
2) Desmostración de que el *SPFA* resuelve el *SSSP* con las restricciones dadas
3) Implementación del *SPFA*

2. Usando estados no triviales en el *SSSP*

13. Algoritmos golosos (a.k.a Greedy)

1. Definición y propiedades de los algoritmos *greedy*
2. Problemas *greedy* conocidos

a) *Interval covering*
b) *Load balancing*
c) *Huffman Codes*
d) *Fractional Knapsack*
e) *Job Scheduling problems*

3. *Sort* y *greedy*
4. Ejemplos de problemas resueltos con algoritmos *greedy* no comunes
5. Técnicas comunes de demostración de algoritmos *greedy*

14. Programación Dinámica I (a.k.a *DP*)

1. Definición y propiedades de los algoritmos DP
2. Calculando la secuencia de Fibonnaci

a) Analizando su árbol de expansión
b) Ilustrando la superposición de problemas
c) Ilustrando la *memoization*
d) Resolviendo el problema con un enfoque *Top-Down*
e) Resolviendo el problema con un enfoque *Bottom-Up*

3. *Top-Down vs Bottom-Up*
4. Más ejemplos
5. Problemas conocidos de DP

a) Máxima suma en rango
1) 1 dimensión
2) 2 dimensiones
3) 3 dimensiones
4) Algoritmo de Kadane

6. Subsequencia creciente más larga (a.k.a *LIS*)
    a) Resolviéndolo en $O(n^2)$
    b) Resolviéndolo en $O(nlogn)$
    c) Recontrucción de soluciones
    d) Demostración de las soluciones
7. El problema de la mochila
    a) Enfoque Bottom-Up
    b) Enfoque Top-Down
8. *Coin Change*
9. *Traveling Salesman Problem (a.k.a TSP)*

15. Programación Dinámica II

1. *DP + Bitmask*
2. Técnicas comunes de representación de estados
3. *Offset technique*
4. DP sobre dígitos

16. Repaso y contest por equipos

Repasar todos los temas anteriores. Despejar dudas de los problemas. Formar equipos de tres por afinidad y dar un contest.

17. Teoría de números I

1. Divisibilidad y números primos
2. Teorema fundamental de la aritmética
3. Test de primabilidad determinista
    a) En $O(n)$ para un número
    b) En $O(\sqrt{n})$ para un número
    c) En $O(nlogn)$ para los n primeros números usando Criba de Eratóstenes
4. Usando la Criba de Eratóstenes para generar números primos $< n$
5. Hallando los factores primos de un número
6. Calculando la suma de factores primos de un número
7. Calculando el número divisores de un número
8. Calculando la suma de divisores de un número
9. Calculando la función φ de Euler de un número
10. Funciones aritméticas

18. Teoría de números II

1. *Square-free numbers*
2. Función de Mobioüs (μ)
3. Construyendo Cribas de Eratóstenes modificadas
4. Máximo comun divisor (gcd) y Mínimo comun múltiplo (lcd)
5. Aritmética modular
6. Ecuaciones diofánticas lineales
7. Teorema chino del resto
8. Test de primabilidad probabilístico
    a) Test de Fermat
    b) Test de Miller-Rabin
9. Algoritmo de Pollard's rho

19. Backtracking II

1. *Bitmask + Backtracking*
2. Técnicas más optimas de representación de estados
    a) El problema de los 8 reinas
    b) Sudoku
3. Más problemas clásicos (*Graph coloring*, *Lights out*, ...)
4. Transición de *Backtracking* a *DP*
5. Más sobre Heavy prunning
6. Meet in the Middle
7. A*
8. IDA*

20. Teoría de juegos combinatorios I

1. Clasificación de los juegos combinatorios
2. *Take-away game*
    a) Hallando los conjuntos de *winning positions (N-set)* y *lossing positions (P-set)*
    b) Resolviendo la versión misère del problema
3. Propiedad caracteristicas de los conjuntos *N-set* y *P-set*
4. Ejemplos ilustrativos para hallar los conjuntos *N-set* y *P-set*
5. El árbol de decisiones
6. Minimax

7. *Alpha-beta prunning*
8. El problema del tic-tac-toe
    a) Usando solo Minimax
    b) Usando *Alpha-beta prunning*

21. Teoría de juegos combinatorios II

1. Definiendo el $nim - sum$ de dos números
2. Resolviendo el juego de Nim
3. Teorema de Bouton
    a) Nim normal
    b) Versión misère
4. *Staircase Nim*
5. Usando grafos dirigidos para generalizar la teoría
6. La función de Sprague-Grundy
7. Suma de juegos combinatorios
8. El teorema de Sprague-Grundy

22. Repaso y contest por equipos

Repasar todos los temas anteriores. Despejar dudas de los problemas. Formar equipos de tres por afinidad y dar un contest.

23. Combinatoria I

1. Técnicas de conteo
2. El teorema de Lucas
3. Aplicaciones más conocidas de los números de Fibonacci
4. Particiones
    a) Los números de Bell
    b) Los números de Stirling
5. Permutaciones
    a) Descomposición en ciclos
    b) Los números de Stirling
6. Los números de Catalan
7. Conteo y DP
8. Principio de inclusión y exclusión (a.k.a PIE)

24. Disjoint Set Union (a.k.a DSU)

1. Descripcion del problema que queremos resolver
2. Operaciones de conjuntos disjuntos
3. Usando *linked-list* para la representación de conjuntos disjuntos
4. La función Ackermann
5. Implementación y analisis de complejidad del DSU usando heurísticas
6. Usando DSU para hallar el número de *connected components* de un grafo
7. Problemas ilustrativos
8. DSU en árboles

25. Fenwick Tree (a.k.a BIT)

1. Descripción del problema que queremos resolver
2. Mostrando soluciones con temas anteriores y analizando la complejidad del tiempo de ejecución y memoria
3. Descripción y análisis de las operaciones del BIT
4. Implementación y análisis de complejidad del BIT
5. Usando el BIT para hallar el número de inversiones de una secuencia
6. Usando el BIT + búsqueda binaria para hallar el k^{th} mayor de una secuencia
7. Problemas ilustrativos
8. BIT multidimensional

26. Segment Tree

1. Descripción del tipo de problema que queremos resolver
2. Descripción y análisis de las operaciones del *segment tree*
3. Problema de suma en rangos con actualizaciones usando *segment tree*
4. Más problemas ilustrativos
5. *Lazy propagation*
6. Más problemas ilustrativos
7. *Segment tree* y árboles

27. Repaso y contest por equipos

Repasar todos los temas anteriores. Despejar dudas de los problemas. Formar equipos de tres por afinidad y dar un contest.

(*) Sujeto a cambios de acuerdo al avance