Técnicas comunes en la solución de algoritmos

Pattern: sliding window

En la serie de Patrones de codificación , intentaremos reconocer patrones comunes subyacentes detrás de cada pregunta de algoritmo, utilizando ejemplos reales de Leetcode .

Primero, presentaremos el patrón de Ventana deslizante que es muy útil para resolver problemas en los que se le pide que encuentre la cadena más larga / más corta , la submatriz o el valor deseado que necesita calcular a partir de las submatrices

Por lo tanto, queremos poder identificar los problemas que funciona el patrón de ventana deslizante.

El problema involucra una estructura de datos que está ordenada e iterable como matrices, cadenas, etc.

El problema es pedir encontrar un subrango en una matriz / cadena, el valor contiguo más largo, más corto, promedio u objetivo.

Existe una aparente solución ingenua o de fuerza bruta que se ejecuta en O (N 2), O (2 N) o en alguna otra complejidad de gran tiempo.

Pattern: two pointers

En la serie de Patrones de codificación, intentaremos reconocer patrones comunes subyacentes detrás de cada pregunta de algoritmo, utilizando ejemplos reales de Leetcode.

La publicación anterior fue sobre el patrón de Ventana deslizante y hoy presentaremos el patrón de Dos punteros que es muy útil para resolver problemas con matrices ordenadas (o Listas vinculadas) que involucran un conjunto de elementos de par, o un triplete o incluso una submatriz.

Por lo tanto, queremos poder identificar los problemas con los que funciona el patrón Two Pointers.

El problema involucra arreglos ordenados (o Listas Vinculadas), un conjunto de elementos de par, o un triplete o incluso un subconjunto.

Hay un valor objetivo para coincidir o duplicados para eliminar.

La mayor parte de este tipo de problemas se pueden resolver en O (N) complejidad del tiempo y O (1) o O (N) complejidad espacio.

Pattern: fast & slow pointers

En la serie de Patrones de codificación, intentaremos reconocer patrones comunes subyacentes detrás de cada pregunta de algoritmo, utilizando ejemplos reales de Leetcode Las publicaciones anteriores trataban sobre los patrones de Ventana deslizante y Dos punteros y hoy presentaremos el patrón de Punteros rápidos y lentos (también conocido como Algoritmo de tortuga y liebre de Floyd), que es muy útil cuando se trata de Listas o matrices cíclicas vinculadas.

Al moverse a diferentes velocidades, el algoritmo demuestra que los dos punteros se encontrarán eventualmente. El puntero rápido debería atrapar al puntero lento una vez que ambos punteros estén en un bucle cíclico.

Este enfoque es bastante útil cuando se trata de listas o matrices cíclicas vinculadas.

Cuando el problema involucra algo relacionado con estructuras de datos cíclicos, debe pensar en el patrón de punteros rápidos y lentos.

Pattern: Merge Intervals

En la serie de Patrones de codificación, intentaremos reconocer patrones comunes subyacentes detrás de cada pregunta de algoritmo, utilizando ejemplos reales de Leetcode.

Las publicaciones anteriores trataban sobre los patrones de Ventana deslizante, Dos punteros y Punteros rápidos y lentos, y hoy presentaremos el patrón Intervalos de fusión, que es muy útil para resolver los problemas que implican intervalos, elementos superpuestos que deben fusionarse, etc. Este enfoque es bastante útil cuando se trata de intervalos, elementos superpuestos o intervalos de fusión. Cuando el problema involucra estas palabras clave, debe pensar en el patrón Intervalos de fusión.

Pattern: cyclic sort

En la serie de Patrones de codificación, intentaremos reconocer patrones comunes subyacentes detrás de cada pregunta de algoritmo, utilizando ejemplos reales de Leetcode. Las publicaciones anteriores trataban sobre los patrones de Ventana deslizante, Dos punteros, Punteros rápidos y lentos e Intervalos de fusión y hoy presentaremos el patrón de clasificación cíclica, que es muy útil para resolver los problemas que involucran matrices que contienen números en un rango dado, encontrar los números faltantes o duplicados.

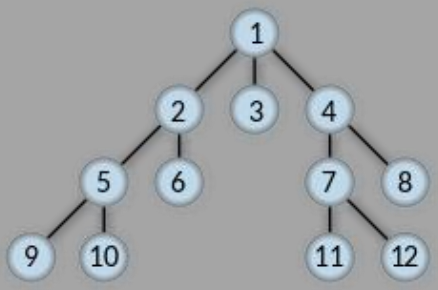
Este enfoque es bastante útil cuando se trata de números en un rango dado y se solicita encontrar los duplicados / los que faltan, etc.

Cuando el problema involucra matrices que contienen números en un rango dado, debe pensar en el patrón de Clasificación Cíclica.

Pattern: In-Place Reversal of a Linkedlist

Es un algoritmo de ordenación simple que crea la matriz ordenada final (o lista) un elemento a la vez. Es mucho menos eficiente en listas grandes que los algoritmos más avanzados, como la clasificación rápida , la clasificación ordenada o la combinación .

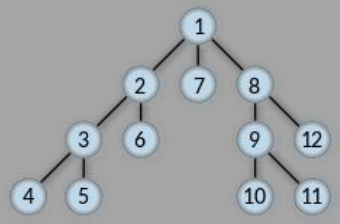
Pattern: Tree Breadth First Search



Búsqueda en anchura (BFS - Breadth First Search) es un algoritmo de búsqueda no informada utilizado para recorrer o buscar elementos en un grafo (usado frecuentemente sobre árboles). Intuitivamente, se comienza en la raíz (eligiendo algún nodo como elemento raíz en el caso de un grafo) y se exploran todos los vecinos de este nodo. A continuación, para cada uno de los vecinos se exploran sus respectivos vecinos adyacentes, y así hasta que se recorra todo el árbol.

Formalmente, BFS es un algoritmo de búsqueda sin información, que expande y examina todos los nodos de un árbol sistemáticamente para buscar una solución. El algoritmo no usa ninguna estrategia heurística.

Pattern: Tree Depth First Search



Una Búsqueda en profundidad (en inglés DFS o Depth First Search) es un algoritmo de búsqueda no informada utilizado para recorrer todos los nodos de un grafo o árbol (teoría de grafos) de manera ordenada, pero no uniforme. Su funcionamiento consiste en ir expandiendo todos y cada uno de los nodos que va localizando, de forma recurrente, en un camino concreto. Cuando ya no quedan más nodos que visitar en dicho camino, regresa (Back tracking), de modo que repite el mismo proceso con cada uno de los hermanos del nodo ya procesado.

Pattern: Two Heaps

Es un algoritmo de ordenamiento no recursivo, no estable, con complejidad computacional 0(n log(n) . Este algoritmo consiste en almacenar todos los elementos del vector a ordenar en un montículo (heap), y luego extraer el nodo que queda como nodo raíz del montículo (cima) en sucesivas iteraciones obteniendo el conjunto ordenado. Basa su funcionamiento en una propiedad de los montículos, por la cual, la cima contiene siempre el menor elemento (o el mayor, según se haya definido el montículo) de todos los almacenados en él. El algoritmo, después de cada extracción, recoloca en el nodo raíz o cima, la última hoja por la derecha del último nivel. Lo cual destruye la propiedad heap del árbol.

Pattern: Subsets

Un subconjunto es un conjunto contenido en otro conjunto, es como si pudieras elegir un helado de los siguientes sabores: {plátano, chocolate, vainilla}

Puede elegir cualquier sabor {banana} , {chocolate} o {vainilla} , o dos sabores: {banana, chocolate} , {banana, vainilla} o {chocolate, vainilla} , o los tres sabores: {plátano, chocolate, vainilla}, o podría decir "ninguno en absoluto gracias", que es el "conjunto vacío": {}.

	List	Number of subsets
zero elements	{}	1
one element	{apple}, {banana}, {cherry}	3
two elements	{apple, banana}, {apple, cherry}, {banana, cherry}	3
three elements	{apple, banana, cherry}	1
	Total:	**8**

Modified Binary Search

La búsqueda binaria modificada , también conocida como búsqueda de medio intervalo , búsqueda logarítmica , o corte binario , es un algoritmo de búsqueda que encuentra la posición de un valor objetivo dentro de una matriz ordenada . La búsqueda binaria compara el valor objetivo con el elemento central de la matriz. Si no son iguales, la mitad en la que el objetivo no puede mentir se elimina y la búsqueda continúa en la mitad restante, nuevamente tomando el elemento del medio para compararlo con el valor objetivo, y repitiendo esto hasta que se encuentre el valor objetivo. Si la búsqueda termina con la mitad restante vacía, el objetivo no está en la matriz.

La búsqueda binaria se ejecuta en tiempo logarítmico en el peor de los casos, haciendo{\ displaystyle O (\ log n)}O (\ log n) comparaciones, donde {\ displaystyle n}norte es el número de elementos en la matriz, el {\ displaystyle O}Oes la notación

Big O , y{\ displaystyle \ log}\Iniciar sesión es el logaritmo. La búsqueda binaria es más rápida que la búsqueda lineal, excepto en matrices pequeñas. Sin embargo, la matriz debe ordenarse primero para poder aplicar la búsqueda binaria. Existen estructuras de datos especializadas diseñadas para la búsqueda rápida, como las tablas hash, que se pueden buscar de manera más eficiente que la búsqueda binaria. Sin embargo, la búsqueda binaria se puede utilizar para resolver una gama más amplia de problemas, como encontrar el siguiente elemento más pequeño o el más grande en la matriz en relación con el objetivo, incluso si está ausente de la matriz.

Bitwise XOR

Los operadores bit a bit realizan una operación en la representación bit a bit (0,1) de sus argumentos, en lugar de como números decimales, hexadecimales u octales. Por ejemplo, el número decimal ocho tiene una representación binaria de 1000. Los operadores bit a bit realizan sus operaciones en dicha representación binaria (por ejemplo 1000) pero devuelven valores numéricos estándar de javascript

Top K Element

Un paradigma popular para hacer frente a este problema es la parte superior - k de consulta, es decir, la clasificación de los resultados y la devolución de los k resultados con las más altas puntuaciones. Numerosas variantes del problema de recuperación top - k y varios algoritmos se han introducido en los últimos años.

K-Way Merge

Una matriz ordenada k es una matriz donde cada elemento está a una distancia máxima de k de su posición de destino en la matriz ordenada . Por ejemplo, consideremos que k es 2, un elemento en el índice 7 en la matriz ordenada , puede estar en los índices 5, 6, 7, 8, 9 en la matriz dada .

0/1 knapsack

0-1 Problema de mochila | DP-10. Dados los pesos y valores de n elementos, coloque estos elementos en una mochila de capacidad W para obtener el valor total máximo en la mochila . En otras palabras, dados dos conjuntos enteros val [0..n-1] y wt [0..n-1] que representan valores y pesos asociados con n elementos respectivamente.

Topological Sort

Tecnología Front End Desarrollo web Javascript . Un ordenamiento topológico u ordenamiento topológico de un gráfico dirigido es un ordenamiento lineal de sus vértices de tal manera que por cada borde dirigido UV desde el vértice u al vértice v, u viene antes de v en el ordenamiento . Esto solo tiene sentido en gráficos dirigidos