Unidad 3

Fundamentos de Programación Competitiva

Logro de sesión

Al finalizar la sesión, el estudiante comprenderá los conceptos de árbol Fenwick

Semana 10

Árbol Fenwick

Contenido:

- Árbol Fenwick - Construcción
- Árbol Fenwick - Update
- Árbol Fenwick – Query

Arbol Binario Indexado (Fenwick)

- El Árbol de Fenwick (también conocido como Árbol Binario Indexado o BIT por sus siglas en ingles) fue propuesto por Peter M Fenwick en 1994 para resolver problemas de compresión de datos.
- Es una estructura de datos muy eficiente para calcular sumas acumulativas.
- El Fenwick Tree es una estructura de datos útil para implementar tablas dinámicas de frecuencia acumulada.

Arbol Binario Indexado (Fenwick)

- Ejemplo, los puntajes de prueba son valores enteros que van desde [1..10]. La siguiente tabla muestra la frecuencia de cada puntaje de prueba individual:

∈ [1..10] y la f frecuencia acumulada de los puntajes de las pruebas que van desde [1..i] denotados por cf[i], es decir, la suma de las frecuencias de los puntajes de las pruebas 1, 2.

Arbol Binario Indexado (Fenwick)

Index/ Score	Frequency f	Cumulative Frequency cf	Short Comment
0	-	-	Index 0 is ignored (as the sentinel value).
1	0	0	cf[1] = f[1] = 0.
2	1	1	cf[2] = f[1] + f[2] = 0 + 1 = 1.
3	0	1	cf[3] = f[1] + f[2] + f[3] = 0 + 1 + 0 = 1.
4	1	2	cf[4] = cf[3] + f[4] = 1 + 1 = 2.
5	2	4	cf[5] = cf[4] + f[5] = 2 + 2 = 4.
6	3	7	cf[6] = cf[5] + f[6] = 4 + 3 = 7.
7	2	9	cf[7] = cf[6] + f[7] = 7 + 2 = 9.
8	1	10	cf[8] = cf[7] + f[8] = 9 + 1 = 10.
9	1	11	cf[9] = cf[8] + f[9] = 10 + 1 = 11.
10	0	11	cf[10] = cf[9] + f[10] = 11 + 0 = 11.

Conceptos

- El BIT permite solucionar este problema de forma mas eficiente. Para solucionarlo, definiremos un árbol donde habrán nodos responsables, y nodos dependientes.

- Cada nodo responsable se encarga de mantener la suma de los valores de todos sus nodos dependientes, los valores en los nodos responsables que estén en el intervalo, y sumarlos, ya que estos comprimen en ellos la información de sus nodos dependientes.

Conceptos

- Se define la función **lowbit(i)**, donde un número “i” cual es el menor bit que tiene valor 1 en la representación binaria de este numero.
- Por ejemplo, el numero 6 es 110 en binario (1×2^2 + 1×2^1 + 0×2^0), y su menor bit es 1×2^1 = 2.

```
1 | public int lowbit(int i) { return (i & -i); }
```

Conceptos

- Cada elemento i sera responsable por los elementos entre las posiciones i-lowbit(i)+1 hasta i, y almacenara el valor de la suma de los elementos en ese intervalo. Por ejemplo, lowbit(6)=2, por tanto, el elemento en la posición 6 es responsable por el elemento 5 y 6

- En el caso de 8, lowbit(8) = 8, y el elemento 8 seria responsable por los valores desde 1 hasta la 8.

Conceptos

Vamos a implementarlo…

Muchas Gracias!!!