Estructuras de datos

Wilmer Emiro Castrillón Calderón

10 de marzo de 2020

1. Tablas aditivas

Son estructuras de datos utilizadas para realizar operaciones acumuladas sobre un conjunto de datos estáticos en un rango específico, es decir, ejecutar una misma operación(como por ejemplo la suma) sobre un intervalo de datos, se asume que los datos en la estructura no van a cambiar. Estas estructuras también son conocidas como *Cumulative Frequency Table* o de manera mas informal *Tablas aditivas*, aunque no necesariamente son exclusivas para operaciones de suma, pues la idea general es aplicable a otras operaciones.

Durante el cálculo de una misma operación sobre diferentes rangos se presenta superposición de problemas, las tablas aditivas son utilizadas para reducir la complejidad computacional aprovechando estas superposiciones utilizando tablas de memorización.

Ejemplo inicial.

Dado un vector V = {5,2,8,2,4,3,1} encontrar para múltiples consultas la suma de todos los elementos en un rango [i,j], indexando desde 1, por ejemplo con el rango [1, 3] la suma es [5+2+8] = 15.

La solución trivial es hacer un ciclo recorriendo el vector entre el intervalo [i,j], en el peor de los casos se debe recorrer todo el vector, esto tiene una complejidad $O(n)$ puede que para una consulta sea aceptable, pero en casos grandes como por ejemplo un vector de tamaño 10^5 y una cantidad igualmente de consultas el tiempo de ejecución se hace muy alto, por lo tanto se hace necesario encontrar una mejor solución.

Como un aspecto clave se puede observar que la solución para la consulta [1,4] y la consulta [2,5] ambas tienen un intervalo en común, es decir, existe una superposición de problemas en el rango [2,4], otro aspecto clave es que el problema se puede reescribir de la siguiente manera: sea $suma(x) = \sum_{k=1}^{x} V_k$ entonces: $\sum_{k=i}^{j} V_k = suma(j) - suma(i-1)$ cuando $i \neq 1$. Ahora pre-calculando $suma(x)$ se puede dar una solución inmediata a cada consulta, esto se puede resolver utilizando un enfoque básico de programación dinámica

Para encontrar $suma(x)$ se puede reescribir como: $suma(x) = V_x + suma(x-1)$ con caso base $suma(1) = V_1$, de esta manera se tendría la siguiente solución en C++

```
int V[] = {5,2,8,2,4,3,1}, memo[6];

void precalcular(){
    memo[0] = 0;
    for(int i = 0; i < 7; i++)
        memo[i] = V[i] + memo[i-1];
}

int query(int i, int j){ return memo[j] - memo[i-1]; }
```

Suma de subconjuntos:

Ejemplo de subtitulo

2. Bibliografia

http://trainingcamp.org.ar/anteriores/2017/clases.shtml.
http://programacioncompetitivaufps.github.io/
https://www.geeksforgeeks.org/dynamic-programming-subset-sum-problem/
libro: competitive programming 3.