ABB Óptimo, Flujo Máximo

Pedro O. Pérez M., PhD.

Análisis y diseño de algoritmos avanzados
Tecnológico de Monterrey

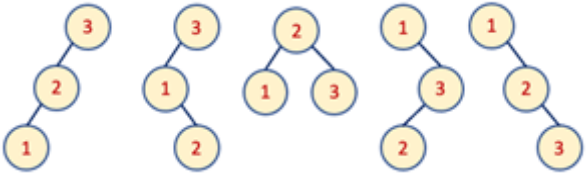
pperezm@tec.mx

12-2021

Contenido

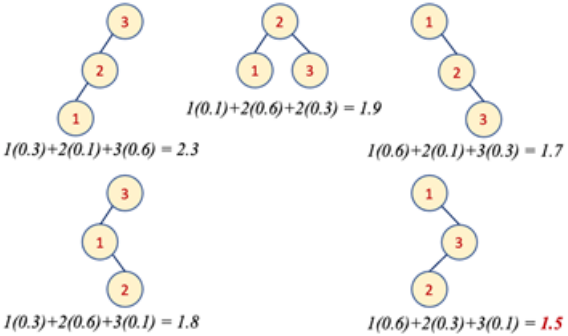
Definición

Como recordarán, un árbol binario de búsqueda (Binary Search Tree, BST), es un árbol binario que almacena valores comparables únicos. Los valores son almacenados de tal forma que para cada nodo, los valores de los nodos de su subárbol izquierdo son menores y todos los valores del subárbol derecho son mayores a él. La altura del árbol es la cantidad máxima de comparaciones que se tienen que hacer como máximo para buscar un dato.



El problema del BST óptimo es dadas n llaves, cada llave con su probabilidad de ser buscadas, encontrar la estructura del árbol que minimice el tiempo promedio de búsqueda. El tiempo promedio de búsqueda de un BST se calcula sumado la cantidad de comparaciones que se requieren para llegar a cada nodo multiplicado por su probabilidad.

Por ejemplo, si se tuvieran 3 datos con las siguientes probabilidades, el 1 con probabilidad de 0.6, el 2 con probabilidad de 0.1 y el 3 con probabilidad de 0.3.



Algoritmo de Gilbert & Moore

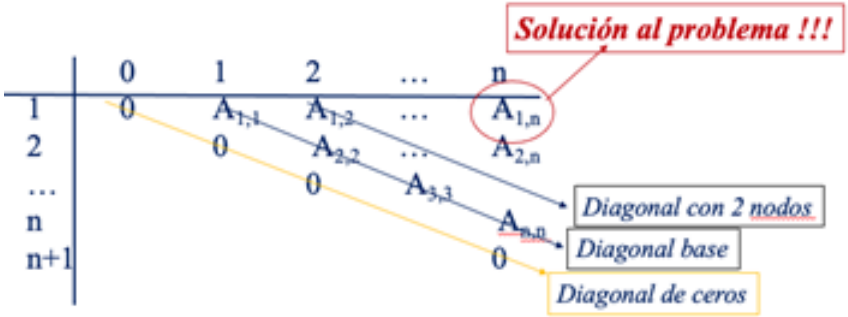
El algortimo propuesto por Gilbert & Moore en 1959, utiliza la técnica de programación dinámica para resolver este problema. El algoritmo utiliza una matriz A de dimensiones $(n+1)x(n+1)$ donde n es la cantidad de datos a almacenar, teniendo los reglones 1 hasta $n+1$, y las columnas 0 hasta n. En cade celda, $A[i][j]$, se almacena el mínimo promedio de búsqueda de los nodos i hasta j. Lo que se desea es encontrar el tiempo mínimo promedio de búsquedas de todo el árbol, el cual quedará almacenado en la celda $A[i][n]$.

1. Al inicio, se coloca en la celda $A[i][i]$ la probabilidad, p_i, en la búsqueda del i-ésimo elemento. Este se considera el caso base, un árbol de un solo nodo.
2. Partiendo de esto, hay que encontrar la solución general. Asumiendo que k es el nodo raíz del árbol óptimo desde i hasta j, esto es, k tiene que ser un valor entre i y j inclusive. Los subárboles del nodo raíz k serán BST óptimos que tienen un tiempo mínimo promedio previamente calculado y almacenados en:
   - El subárbol izquierdo tiene las llaves desde i hasta $k-1$, y su valor del tiempo mínimo promedio estará almacenado en $A[i][k-1]$.
   - El subárbol derecho tiene las llaves desde $k+1$ hasta j, y su valor del tiempo mínimo promedios estará almacenado en $A[k+1][j]$.

Ahora bien, el cálculo del subárbol desde i hasta j, considerando a k como la raíz, es:

- La probabilidad de la raíz, p_k.
- La probabilidad de los nodos desde i hasta $k-1$, para bajarlos de 1 nivel, ya que ahora serán el subárbol izquierdo de k.
- La probabilidad de los nodos desde $k+1$ hasta j, para bajarlos 1 nivel, ya que ahora serán el subárbol derecho de k.

$$A[i][j] = \min_{i \leq k \leq j} (A[i][k-1] + A[k+1][j]) + \sum_{a=i}^{j} p_a$$

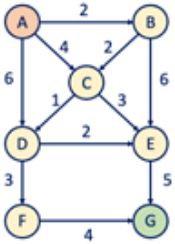

Solución al problema !!!
0 1 2 … n
1 0 A1,1 A1,2 … A1,n
2 0 A2,2 … A2,n
… 0 A3,3
n An,n
n+1 0
Diagonal con 2 nodos
Diagonal base
Diagonal de ceros

Procedure 1 GILBERT_MOORE

Input: $N : Integer, Pi : Vector$
  Generate a $(N+1)x(N+1)$ matrix called A
  for $i \leftarrow 1$ **to** N **do**
    $A[i][i-1] \leftarrow 0$, $A[i][i] \leftarrow P[i]$
  end for
  for $diag \leftarrow 1$ **to** $N-1$ **do**
    for $i \leftarrow |$ **to** $N - diag$ **do**
      $j \leftarrow i + diag$, $minimum \leftarrow \infty$
      for $k \leftarrow i$ **to** j **do**
        $minimum \leftarrow \min(A[i][k-1] + A[k+1][j], minimum)$
      end for
      $minimum \leftarrow minimum + \sum_{a=i}^{j} p_a$
    end for
  end for
  return $A[1][N]$

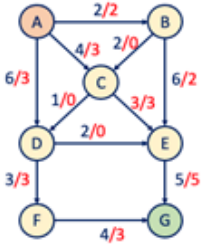
Definición

El problema del flujo máximo es que, dado un grado dirigido, debemos determinar el máximo flujo que puede ser movido de un punto de partida a un punto de llegada. Este problema se puede emplear para determinar el flujo de agua de un punto a otra de la ciudad, la transmisión de datos entre dos equipos, o flujo de vehículos. Por ejemplo, para el grafo que está a la izquierda el flujo máximo de A a G es 8.



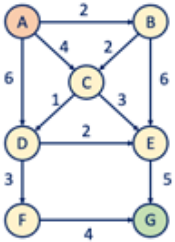
La solución nos indica que:

- Del arco A-B se utilizan las 2 unidades que tiene el arco.
- Del arco A-C solo utilizan 3 unidades de las 4 que tiene el arco.
- Del arco A-D solo utilizan 3 unidades de las 6 que tiene el arco.
- Del arco B-C no se utilizan nada de las unidades del arco.
- Del arco B-E solo utilizan 2 unidades de las 6 que tiene el arco.
- Del arco C-D no se utilizan nada de las unidades del arco.
- Del arco C-E se utilizan las 3 unidades que tiene el arco.
- Del arco D-E no se utilizan nada de las unidades del arco.
- Del arco D-F se utilizan las 3 unidades que tiene el arco.
- Del arco E-G se utilizan las 5 unidades que tiene el arco.
- Del arco F-G solo se utilizan 3 de las 4 unidades del arco.



Algoritmo

El algoritmo de Dinic, nombrado así por su autor Yefim A. Dinitz (1970), va iterando en posibles trayectorias del punto de origen al punto destino, etiquetando a los nodos intermedios por su nivel dentro de la trayectoria y obteniendo un costo residual en cada iteración.



Procedure 2 DINIC

Input: $G : Graph, start, end : Vertex$

```
maxFlow ← 0
while TRUE do
    path ← BFS(start, end)
    min ← AUGMENT(start)
    if min = 0 then
        break
    end if
    maxFlow ← maxFlow + min
end while
return maxFlow
```