TORRE DE CUBOS

Sara Chica, Rodrigo Gualtero

1 de Octubre, 2012

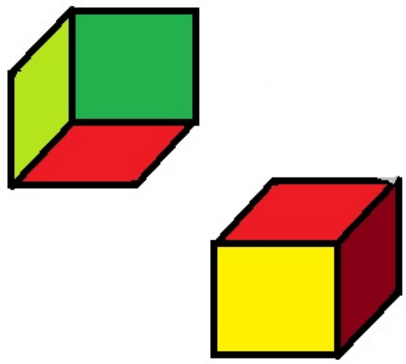
Índice

1. Introducción

Este es un problema de la UVA, identificado con el código *10051*, El cual consiste en armar la torre de cubos más alta posible a partir de N cubos teniendo en cuenta los colores de cada cubo, su respectivo tamaño y algunas restricciones que se mostraran más adelante.

Un cubo es *apilable* si el color de la cara superior del cubo que se encuentra en la parte inferior de la torre tambien existe en una de las caras del cubo que se desea *apilar*; esto se cumple siempre y cuando el cubo que se desee apilar tenga un tamaño más pequeño que que cubo base.

A continuación se presenta un ejemplo en donde se puede observar más claramente lo dicho anteriormente:



El color de la cara inferior de un cubo corresponde al color de la cara superior del otro, nótese que el cubo inferior es mas grande que el que se encuentra en la parte superior.

2. Definición del problema

Para este problema cada cubo puede modelarse como un nodo de un grafo. Este grafo cumple varias caracteristicas al momento de apilar los cubos:

1. Un nodo no puede apilarse consigo mismo.
2. El grafo es no dirigido, es decir siendo a y b cubos si a está conectado con b entonces b estará conectado con a.
3. El grafo es fuertemente conexo, porque necesariamente podemos encontrar un camino del cubo a hasta el cubo b.

2.1. Entrada

En un principio recibimos un entero N de $(0 \leq N \leq 500)$ que representa el numero de cubos con los cuales se desea construir la torre mas alta; es importante saber que los cubos se encuentran ordenados por tamaños. Seguido a esto entran 5 números $(1 \leq N \leq 100)$ que representan cada uno de los colores de las caras de los cubos; estos números se encuentran ordenados uno detras de otro de la siguiente forma: *front, back, left, right, top, bottom.*

2.2. Salida

Se debe imprimir la torre mas alta de cubos; cada cubo se imprime en una linea independiente por orden de tamaños mostrando su ubicación de la siguiente manera:

número del Cubo seguido de su cara inferior bien sea: front, back, left, right, top, bottom.

Ejemplo:
1front
2top
5back

3. Modelamiento matemático

Sea C_i color de un cubo donde $(1 \leq i \leq 6)$
Donde C es un número tal que $(1 \leq C \leq 100)$
Sea G un grafo, $G(V, L)$, donde

V es el conjunto de nodos(Cubos) del grafo,

$$V = \{a_1, a_2, ..., a_n\}$$

y L es el conjunto de todas las conexiones,

$$L = \{(a_1, a_2), (a_2, a_3), ..., (a_{n-1}, a_n)\}$$

donde,

$$a_i = \{C_1, C_2, C_3, C_4, C_5, C_6\}$$

El grafo es conexo si existe una sucesión de arcos tal que es posible encontrar un camino para llegar a cualquier elemento del Conjunto.
El grafo es no dirigido si $(a_i, a_{i+1}) \Rightarrow (a_{i+1}, (a_i)$.
No existe una conexión tal que (a, a); es decir $(a, a) \notin L$
Los nodos $(a_i, a_{i+1}) \in L$ si y solo si $(\exists C | C \in a_i \wedge C \in a_{i+1})$
Una matriz de adyacencia es una matriz cuadrada que se utiliza para representar las conexiones de un grafo, en donde 0 implica que no hay ninguna

conexión, y 1 implica que existe una conexión. La matriz para este ejercicio cumple con las siguientes propiedades:

1. Todos los elementos de su diagonal principal son 0, ya que ningún nodo de la matriz tiene conexión consigo mismo.

$$\forall i, j | 0 \leq i \leq N \wedge 0 \leq j \leq N \wedge i = j : M_{ij} = 0$$

4. Planteamiento de la Solución

Para determinar la solución del problema primero se debe saber como se pueden apilar los cubos como se dijo anteriormente dos cubos son apilables si $(\exists C | C \in a_i \wedge C \in a_{i+1})$ para esto se puede utilizar una matriz de adyacencias que permita identificar las conexiones entre todos los nodos. La matriz se muestra a continuación.

$$\begin{bmatrix} 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Donde las filas y las columnas son los N cubos

Como todos los cubos estan en orden notese que la matriz es triangular superior con la diagonal principal igual a cero

De la misma forma se hace una matriz que permita identificar los colores que se encuentran relacionados con todos los cubos es decir una matriz donde los colores son las filas y los cubos las columnas.
Teniendo estas dos matrices encontramos el color que tiene mas conexiones con los cubos y verificamos si el primer cubo ingresado posee este color, de no ser así se elimina; una vez ya tengamos el cubo con este color guardamos el color inverso de la siguiente manera.
Sea *Opuesto* una función y C un color entonces:

$$Opuesto(C) = \begin{cases} C_i & si \quad C = C_{i+1} \\ C_{i+1} & si \quad C = C_i \end{cases}$$

El color obtenido por la funcion anterior será el color que permitirá empezar a armar la torre porque este color se le enviará al siguiente cubo y este a su vez retornará otro color para enviarle al siguiente cubo hasta terminar con todos los cubos enviados por el usuario; si uno de los cubos a los que se le envia el color no lo tiene entonces este cubo debe ser eliminado.
A medida que un cubo retorna su color opuesto debe imprimirse.

5. Conclusiones

1. La mejor forma para almacenar las conexiones bien sean de los colores con los nodos o los nodos con los nodos es una matriz de adyacencia, ya que teniendolas almacenadas de esa forma se logra de manera más eficaz conocer información y evitar recorridos.

2. Este problema tiene varias formas de solución; aunque es un problema que se puede solucionar por grafos una forma más eficiente de solución es a partir de derivación de programas ya que se tiene una precondición y una poscondición lo cual implica reducción de código y reducción de tiempo de ejecución.