Análisis y diseño de algoritmos

Árboles de recubrimiento mínimo

```c
// Arbol de recubrimiento minimo
#include <stdio.h>
#include <stdlib.h>

// Minimo de dos enteros
int min(int x, int y) { return (x < y)? x: y; }


struct nodo
{
    int data;
    struct nodo *izquierda, *derecha;
};

// La función devuelve el tamaño  mínimo de vértices.
int vCover(struct nodo *root)
{
// El tamaño de la cobertura mínima de vértices es cero si el árbol está vacío o allí
// es solo un nodo
    if (root == NULL)
        return 0;
    if (root->izquierda == NULL && root->derecha == NULL)
        return 0;

    // Calcula el tamaño de la cobertura del vértice cuando la raíz forma parte de ella
    int size_incl = 1 + vCover(root->izquierda) + vCover(root->derecha);

    // Calcula el tamaño de la cobertura del vértice cuando la raíz no forma parte de ella
    int size_excl = 0;
    if (root->izquierda)
      size_excl += 1 + vCover(root->izquierda->izquierda) + vCover(root->izquierda->derecha);
    if (root->derecha)
      size_excl += 1 + vCover(root->derecha->izquierda) + vCover(root->derecha->derecha);

    // Devuelve el mínimo de dos tamaños
    return min(size_incl, size_excl);
}

// Crear un nodo
struct nodo* nuevoNodo( int data )
{
    struct nodo* temp = (struct nodo *) malloc( sizeof(struct nodo) );
    temp->data = data;
    temp->izquierda = temp->derecha = NULL;
    return temp;
}

// MAIN DEL PROGRAMA
int main()
{
    // Arbol de prueba
    struct nodo *root         = nuevoNodo(20);
    root->izquierda                = nuevoNodo(8);
    root->izquierda->izquierda          = nuevoNodo(4);
    root->izquierda->derecha         = nuevoNodo(12);
    root->izquierda->derecha->izquierda   = nuevoNodo(10);
    root->izquierda->derecha->derecha  = nuevoNodo(14);
    root->derecha               = nuevoNodo(22);
    root->derecha->derecha        = nuevoNodo(25);

    printf ("El tamaño de la cubierta de vértice más pequeña es %d ", vCover(root));

    return 0;
```

Codigo

main.c

```
31          size_excl += 1 + vCover(root->izquierda->izquierda) + vCover(root
32      if (root->derecha)
33          size_excl += 1 + vCover(root->derecha->izquierda) + vCover(root->
34
35      // Devuelve el mínimo de dos tamaños
36      return min(size_incl, size_excl);
37  }
38
39  // Crear un nodo
40  struct nodo* nuevoNodo( int data )
41  {
42      struct nodo* temp = (struct nodo *) malloc( sizeof(struct nodo) );
43      temp->data = data;
44      temp->izquierda = temp->derecha = NULL;
45      return temp;
46  }
47
48  // MAIN DEL PROGRAMA
49  int main()
50  {
51      // Arbol de prueba
52      struct nodo *root          = nuevoNodo(20);
53      root->izquierda                = nuevoNodo(8);
54      root->izquierda->izquierda          = nuevoNodo(4);
55      root->izquierda->derecha         = nuevoNodo(12);
56      root->izquierda->derecha->izquierda    = nuevoNodo(10);
57      root->izquierda->derecha->derecha   = nuevoNodo(14);
58      root->derecha               = nuevoNodo(22);
59      root->derecha->derecha        = nuevoNodo(25);
60
61      printf ("El tamaño de la cubierta de vértice más pequeña es %d ", v
62
63      return 0;
64  }
```

```
El tamaño de la cubierta de vértice más pequeña es 3

...Program finished with exit code 0
Press ENTER to exit console.
```

Output