**ANALISIS Y DISEÑO DE ALGORITMOS**

Practica Final

Ramificación y poda

Grado en ingeniería informática

Francisco Joaquín Murcia Gómez 48734281H

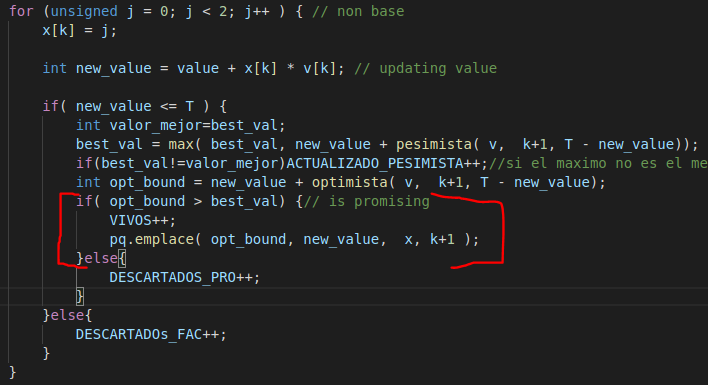
Grupo 1

1. **Estructuras de datos**
2. **Nodo**

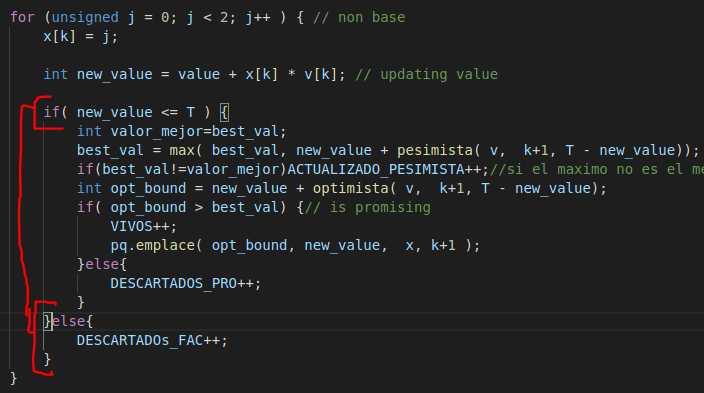
Nodo, está compuesto por la siguiente estructura: <int, int, Sol, int>, siendo Sol un vector de tipo int, que guarda las soluciones factibles, el primer int es la cota optimista, el segundo, es el valor que tiene ese nodo, y el ultimo int es el que controla para no recorrer los nodos recorridos.

1. **Lista de nodos vivos**

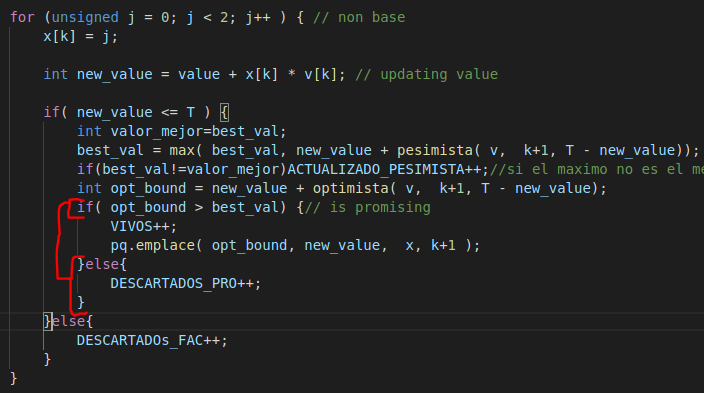
Para listar los nodos vivos usamos la librería “queue” para usar una cola de prioridad. Miramos primero si es factible, si el nuevo valor esta por debajo de la suma máxima, y después si ese valor esta por debajo o igual que la cota optimista.

En el código se ha implementado de la siguiente manera:

1. **Mecanismos de poda**
2. **Poda de nodos no factibles**

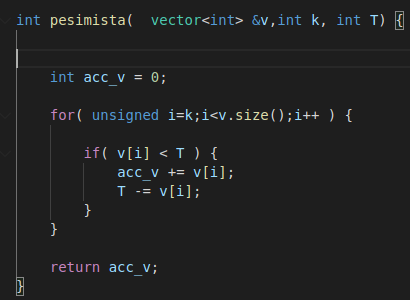
Comprobamos si el valor nuevo es mayor que la suma máxima, en ese caso se descarta, como podemos observar:

1. **Poda de nodos no Prometedores**

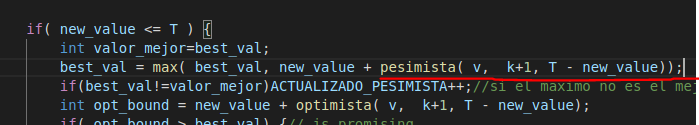
Si no es un nodo prometedor, es decir, en el “else” del “if” que controla los nodos vivos, como se puede ver a continuación:

1. **Cotas pesimistas y optimistas**
2. **Cota pesimista inicial.**

Usamos el algoritmo voraz para la mochila discreta (sin los pesos), de este modo tomara un valor cercano.

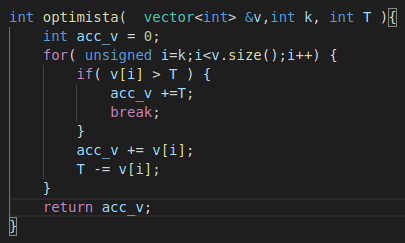
El algoritmo es el siguiente:

1. **Cota pesimista del resto de nodos.**

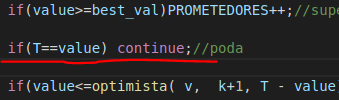
Para el resto de los nodos hacemos lo siguiente:

Hacemos la llamada con los siguientes valores, el vector de valores, añadiendo 1 a la variable que controla para no recorrer los nodos recorridos, y por último, el valor que falta para llegar a la suma máxima.

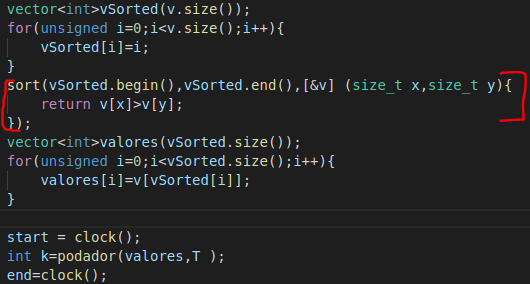
1. **Cota optimista.**

La cota optimista esta implementada a partir del siguiente algoritmo voraz:

1. **Otros medios empleados para acelerar la búsqueda**

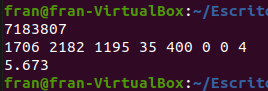
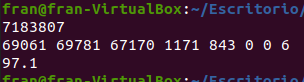
Para acelerar la búsqueda, se ha implementado una poda,

De este modo controlamos si el valor es el máximo dejamos de buscar.

También hemos ordenado el vector de valores antes de iniciar el algoritmo.

1. **Estudio comparativo de distintas estrategias de búsqueda**

Estas pruebas se han realizado con el fichero “Fichero 1k\_bb.problem”

Usando todas las mejoras: Sin las mejoras:

Como resultado tenemos que con mejoras es 17,34 veces más rápido.

Como se puede también observar, no aparecen nodos completados, esto es debido a que encontramos la solución antes de llegar a las hojas.

1. **Tiempos de ejecución**

|  |  |
| --- | --- |
| Fichero X\_bb.problem | Tiempo en ms |
| 1 | 0.015 |
| 10 | 0.012 |
| 20 | 0.698 |
| 30 | 2773.72 |
| 40 | 3409.78 |
| 45 | 3130.84 |
| 50 | 0.047 |
| 100 | 0.547 |
| 200 | 0.45 |
| 1K | 6.231 |
| 2K | 23.491 |
| 3K | 52.677 |