Cuestión 1 (1,5 puntos). ¿Para qué tipo de operaciones el montículo es una estructura adecuada? ¿Qué coste tienen asociado? Escribe el procedimiento flotar que flote el nodo *i* en un montículo T[1..*n*].

Solución:

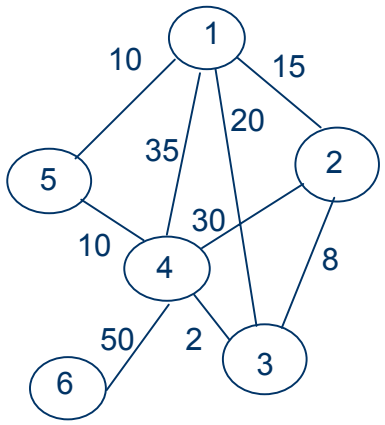
Para las operaciones de listas de prioridad dinámica: hallar el elemento prioritario de un conjunto (O(1)), eliminarlo (O(logn)), añadir un elemento en orden de prioridad (O(logn)). Libro página 188.

Si el montículo es de máximos:

```
PROCEDURE Flotar (i: INTEGER; VAR T: TipoMonticulo)
VAR
  j,k: INTEGER;

BEGIN
        k ← i;
        REPETIR
                j ← k;
                IF (j > 1) AND T[j DIV 2] < T[k]) THEN
                        k ← j DIV 2;
                        Intercambia(T[j],T[k]);
    HASTA j=k;
END Flotar;
```

Cuestión 2 (1,5 puntos). Dado el siguiente grafo, rellene una tabla similar a la adjunta indicando paso a paso cómo el algoritmo de Prim halla un árbol de recubrimiento mínimo (ARM) asociado a dicho grafo, a partir del nodo 1.



Solución:

Paso	Arista Seleccio-nada	Nodos	Aristas ARM
Inicial		1	
1	{1,5}	1, 5	{1,5}
2	{5,4}	1, 4, 5	{1,5}, {5,4}
3	{4,3}	1, 3, 4, 5	{1,5}, {5,4}, {4,3}
4	{3,2}	1, 2, 3, 4, 5	{1,5}, {5,4}, {4,3}, {3,2}
5	{4,6}	1, 2, 3, 4, 5, 6	{1,5}, {5,4}, {4,3}, {3,2}, {4,6}

Cuestión 2 (3 puntos).
a) ¿Qué sucede con la eficiencia de los algoritmos de divide y vencerás si en lugar de utilizar un umbral para decidir cuando hay que volver al subalgoritmo básico, empleamos la recurrencia un máximo de *r* veces, para alguna constante *r*, y utilizamos después el subalgoritmo básico?
b) Se dispone de una bolsa de *n* monedas de oro y una de ellas es falsa. Lo único que distingue a la moneda falsa de las legales es su peso, aunque no hay seguridad de que éste sea mayor o menor que el de las legales. Para descubrir cuál es la falsa, se dispone de una balanza con dos platillos para comparar el peso de dos conjuntos de monedas. En cada pesada lo único que se puede observar es si la balanza queda equilibrada, si pesan más los objetos de la derecha o si pesan más los objetos de la izquierda. Indicad y demostrad informalmente cuantas pesadas hacen falta como mínimo, siendo $n \geq 3$, para determinar cuál es la moneda falsa y si pesa más o menos que las auténticas. No se pide el algoritmo.

Solución:

a) Si se utiliza la recurrencia un máximo de *r* veces antes de aplicar el subalgoritmo básico, el algoritmo resultante no mejorará de forma asintótica sobre el subalgoritmo básico desde el punto de vista de la notación Θ. Podrá ser un algoritmo más rápido que si no hubiera división, pero no más que un factor constante.
b) Se puede dividir el conjunto inicial de monedas en 3 grupos iguales, dejando aparte 1 o las 2 que puedan sobrar. Se pesan y comparan los 3 grupos (2 pesadas). Si la moneda falsa está en uno de los 3 grupos, su peso no coincidirá con el de los otros 2, por lo que descartamos los 2 grupos que pesan igual quedándonos con el que pesa distinto y repitiendo el procedimiento. Si los 3 grupos pesan igual, la moneda está entre la o las que se apartaron y con una pesada de una de ellas con una auténtica se determina cuál es la falsa y si pesa más o menos. Por lo que harán falta $O(2 \log_3(n))$ pesadas.

Problema (4 puntos).

Se quiere realizar en un soporte secuencial (cinta de dos caras) una recopilación de canciones preferidas. Se dispone de una lista de *n* canciones favoritas, junto con la duración individual de cada una. Lamentablemente, la cinta de *T* minutos no tiene capacidad para contener todas las canciones, por lo que se les ha asignado una puntuación (cuanto más favorita es, mayor es la puntuación). Se pretende obtener la mejor cinta posible según la puntuación, teniendo en cuenta que las canciones deben caber enteras y no es admisible que una canción se corte al final de una de las caras.

La resolución de este problema debe incluir, por este orden:

1. Elección del esquema más apropiado, el esquema general y explicación de su aplicación al problema (0,5 puntos).
2. Descripción de las estructuras de datos necesarias (0.5 puntos).
3. Algoritmo completo a partir del refinamiento del esquema general (2,5 puntos).
4. Estudio del coste del algoritmo desarrollado (0.5 puntos).

Solución:

1.- Ramificación y poda. Se trata de una optimización y no existe una función de selección de soluciones parciales que nos permita garantizar que dichas soluciones parciales construidas lleven a la óptima final, por lo que no se trata de un algoritmo voraz. Tampoco existe una forma de dividir el problema en subproblemas que se puedan resolver independientemente, por lo que tampoco es posible un esquema divide y vencerás.

```
funcion ramificacion_poda (ensayo) dev ensayo {ensayo es un nodo}
        m ← monticulo_vacio();
        cota_superior ← cota_superior_inicial;
        solucion ← primera_solucion;
        añadir_nodo(m, ensayo);
        mientras ¬ vacio(m) hacer
                nodo ← extraer_raiz(m);
                si valido(nodo) entonces {solución completa}
                        si coste_asig(nodo) < cota_superior entonces
                                solucion ← nodo;
                                cota_superior ← coste_asig(nodo)
                fsi
                si no
                        si cota_inferior(nodo) ≥ cota_superior entonces dev solucion
                        si no
                                para cada hijo en compleciones(nodo) hacer
                                        si cota_inferior(hijo) < cota_superior
                                                añadir_nodo(m, hijo)
                                        fsi
                                fpara
                        fsi
                fsi
        fmientras
ffuncion
```

Se considera en cada etapa una canción y se considera grabarla en cada una de las dos caras, si hay espacio suficiente, así como la posibilidad de no grabarla en la cinta. La posibilidad de grabarla en la segunda cara sólo se plantea cuando la ocupación de las dos caras es distinta.

Una cota superior podría ser la puntuación estimada considerando como espacio libre total la suma del espacio libre en cada cara y grabar el máximo posible de canciones, aunque alguna quede cortada.

2.- Se utiliza un montículo en el que cada nodo será necesario almacenar:

- Solución parcial
- Etapa
- Puntuación-estimada
- Puntuación acumulada
- Ocupación de cada cara

Además en un array tendremos las canciones con su puntuación y su duración.

Por lo que la estructura de cada nodo sería un registro con los siguientes campos:

Sol[1..n] de 0..2
K: 0..n
Puntuación_acumulada: real
Puntuación-estimada: real {prioridad}
Ocupada[1..2] de real

3.- Una descripción detallada de la solución puede encontrarse en el texto de

Estructuras de datos y métodos algorítmicos.
N. Martí Oliet, Y. Ortega Mallén, J.A. Verdejo López.
Prentice Hall, 2003.
Sección 15.3, pag. 513

4.- Una estimación del coste es el tamaño del árbol de búsqueda, que en el peor caso crece como O(n!), ya que cada nodo del nivel k puede expandirse con las n − k canciones que quedan por considerar.