algoritmia SOLUCIONES PROFESORES...

gemma_o2

Algoritmia y Estructuras de Datos Avanzadas

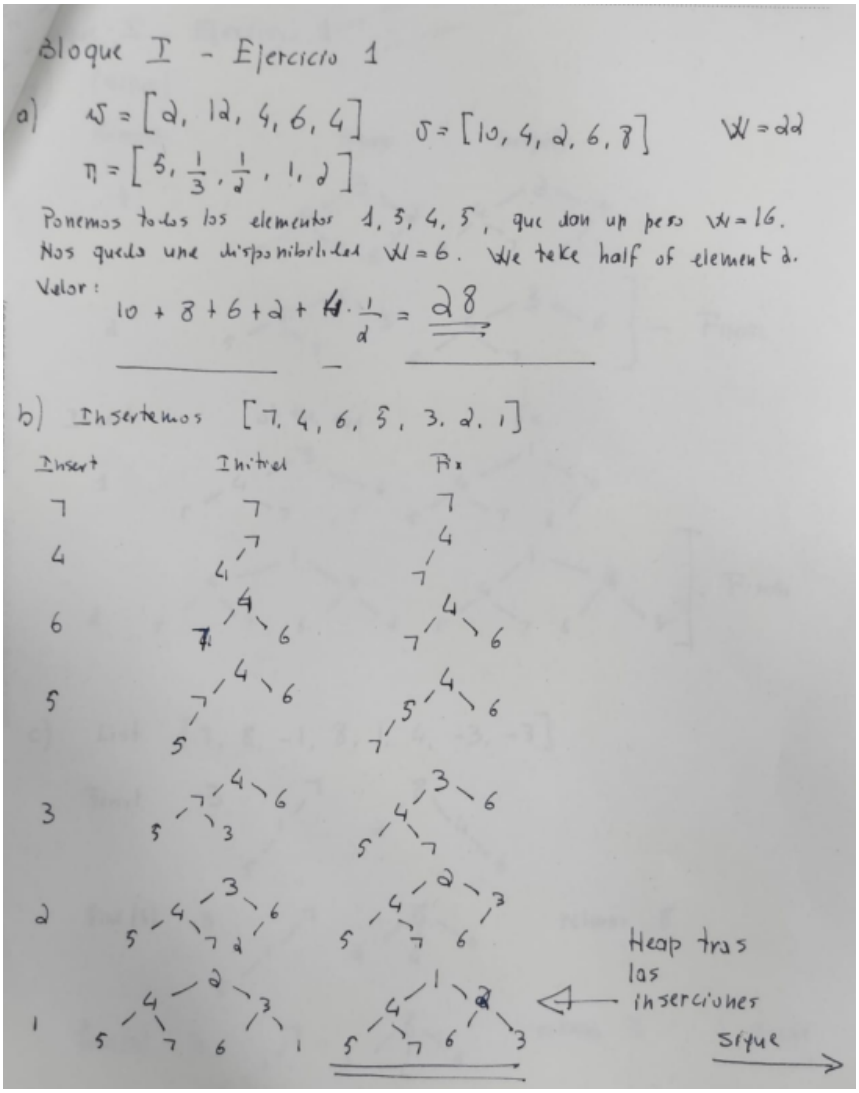
2º Grado en Ingeniería Informática

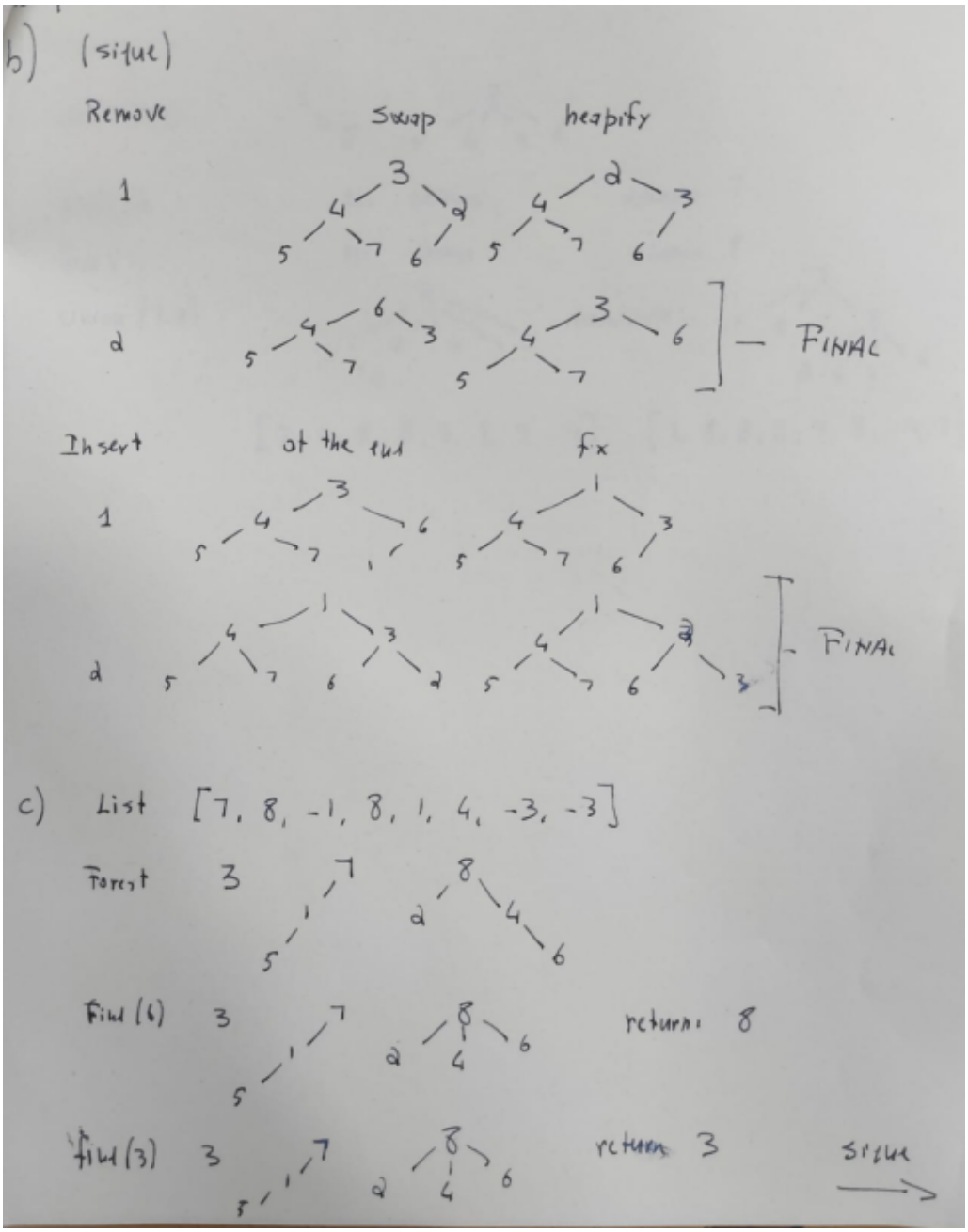
Escuela Politécnica Superior
Universidad Autónoma de Madrid

ALGORITMIA SOLUCIONES ENERO 2023

(las soluciones están ordenadas según los ejercicios que ha corregido cada profesor.)

SOLUCIONES





si lees esto me debes un besito

ING BANK NV se encuentra adherido al Sistema de Garantía de Depósitos Holandés con una garantía de hasta 100.000 euros por depositante. Consulta más información en ing.es

1/6
Este número es indicativo del riesgo del producto, siendo 1/6 indicativo de menor riesgo y 6/6 de mayor riesgo.

Si ya tuviste sufi con tanto estudio...
Te dejamos este espacio
para desahogarte.

Pinta, arranca,
llora... tú decides ;)

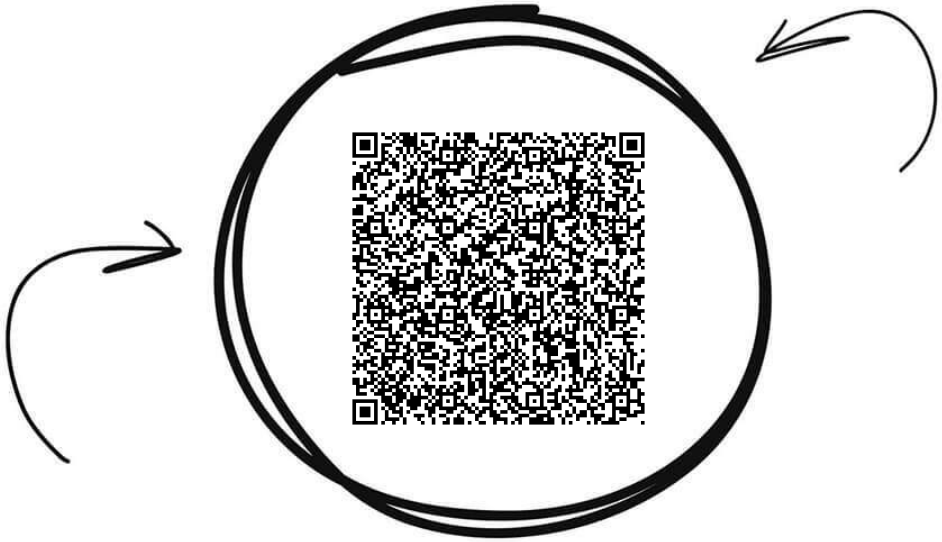
¿Te sientes más liberado?
Sigue siéndolo con la **Cuenta NoCuenta:**
libre de comisiones*, y de lloraditas.

¡Quiero una de esas!

*TIN 0 % y TAE 0 %.

do your thing

Algoritmia y Estructuras de...



Banco de apuntes de la

WUOLAH

Comparte estos flyers en tu clase y consigue más dinero y recompensas

1. Imprime esta hoja
2. Recorta por la mitad
3. Coloca en un lugar visible para que tus compis puedan escanar y acceder a apuntes
4. Llévate dinero por cada descarga de los documentos descargados a través de tu QR

Bloque II - Ejercicio 2

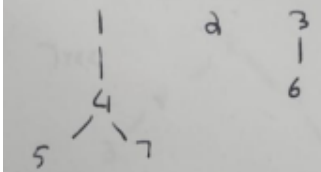
a) i) Función $d: V \times V \to \mathbb{R}$ tal que, para todos nodos u, v, w es $d(u,v) \leq d(u,w) + d(w,v)$

ii) El método T con el minimal spanning tree da un coste como mucho el doble del óptimo, por tanto, en este caso, el máximo es $2 \cdot |a| = 2 \cdot 2 = 4$

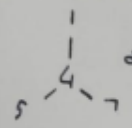
b) i) Lista de adyacencia: (con d, f del DFS)

d	f	nodo	ady
1	8	1	4 → 7
9	10	2	1 → 6
11	14	3	2 → 6
2	7	4	5 → 7
3	4	5	1
12	13	6	NIL
5	6	7	5

DFS Trees



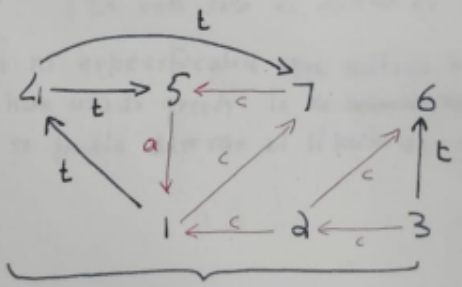
Nota: hay quien ha empezado el segundo árbol con el nodo 3, consiguiendo



— Lo he considerado correcto

Arcos:

t: tree
d: descendent
a: ancestor
c: cross

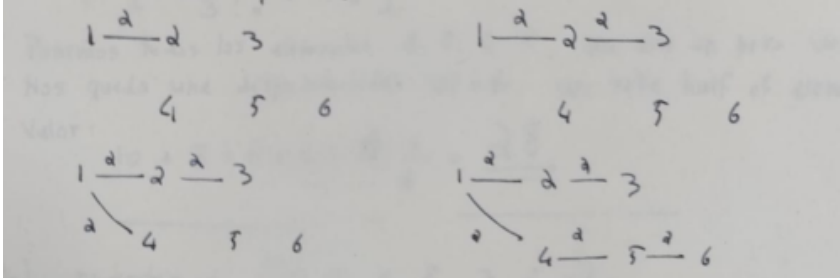


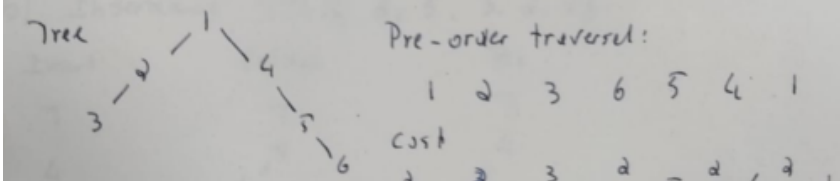
Con el otro árbol la solución salía diferente pero también era correcta.

Bloque II - Ejercicio 2

La suma de dos arcos es siempre ≥ 4, y ningún arco (dibujado o no) tiene coste mayor de 4, por tanto la desigualdad se cumple.

TSP: Minimal Spanning Tree



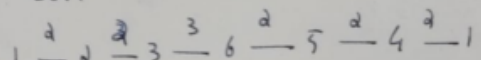


Tree

Pre-order traversal:

1 2 3 6 5 4 1

Cost

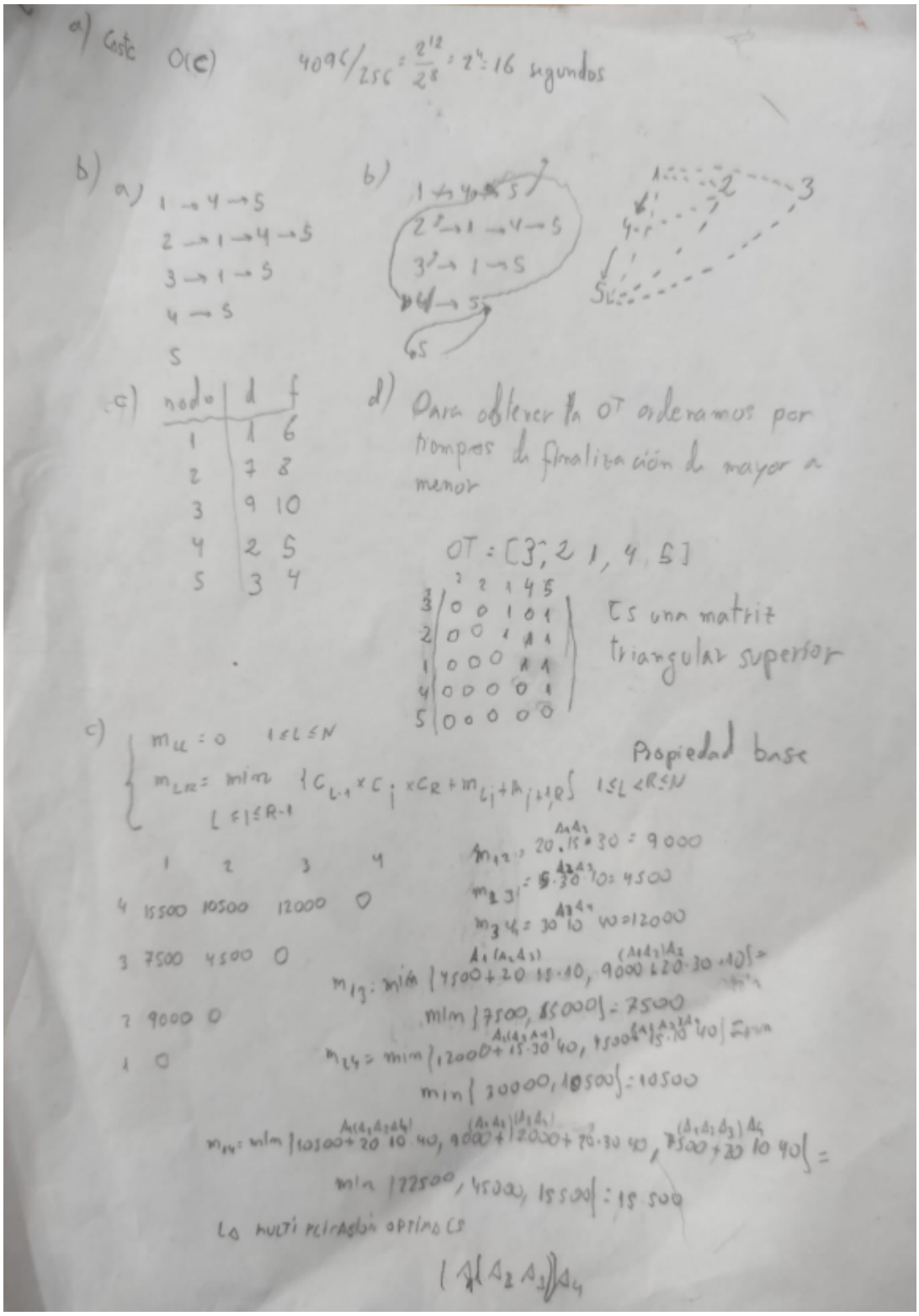


Cost: 13

Optimal: $\geq \frac{13}{2} = 6.5$

(En este caso el óptimo es 13)

Nota: El enunciado no especificaba que método había que usar. Hay unos pocos que han usado greedy: lo he ~~considerado~~ dado por bueno - pero así no se podía derivar el límite de 6.5.



Problema (2b)

Sea el grafo $G = (V, E)$ donde V es un conjunto de nodos y E el conjunto de aristas $(u, v)\ \forall u, v \in V$, el siguiente algoritmo utiliza el TaD Conjunto Disjunto para hallar las componentes conexas de un grafo no dirigido:

- Inicializar el conjunto disjunto. Se crearán tantos subconjuntos como nodos tenga el grafo.
- Iterar sobre la lsta de aristas. Cada vez que se procesa una arista (u, v) se invoca a la función `union(u, v)` del conjunto disjunto donde los parámetros de la función son los nodos u, v que forman la arista. (*Nota:* asumimos que `union(u,v)` sobre dos elementos cualesquiera u e v invoca a `find(u)` y a `find(v)`, sino fuese así en el algoritmo había que invocar a la función con los parámetros `union(find(u),find(v))`)
- Al finalizar del algoritmo habrá tantas componentes conexas como subconjuntos en el Conjunto Disjunto.

```python
def connected_components (G: Grafo) -> DS:
    ''' Devuelve un Subconjunto Disjunto con las componentes conexas del grafo G'''

    # Generar un subconjunto por cada nodo del grafo
    ds = DS_init (V)
    # Procesar todas las aristas del grafo
    for u, v in E:
        DS_union (u, v, ds)  # DS_union (find(u), find(v))
    return ds
```

- Evolución del algoritmo anterior *paso a paso*. En negrita se indica el representante de cada subconjunto disjunto.

Primitive DS	**edge processed**	**disjoin subsets S_x**					
S = init (G[V])	···	{**1**}	{**2**}	{**3**}	{**4**}	{**5**}	{**6**}
union (1, 4, S)	(1,4)	{**1**, 4}	{**2**}	{**3**}	{**5**}	{**6**}	
union (1, 5, S)	(1,5)	{**1**, 4, 5}	{**2**}	{**3**}	{**6**}		
union (2, 3, S)	(2,3)	{**1**, 4, 5}	{**2**, 3}	{**6**}			
union (2, 6, S)	(2,6)	{**1**, 4, 5}	{**2**, 3, 6}				
union (5, 6, S)	(5,6)	{**1**, 4, 5, 2, 3, 6}					

- Tal y como se observa en la tabla anterior al finalizar el algoritmo tenemos un único subconjunto y, por tanto, una única componente conexa.

Parcial 1

Problema (2a)

El menor número posible de bits promedio por caracter está dado por la entropía

$$H = -\sum \rho_i \cdot \log_2 \rho_i$$

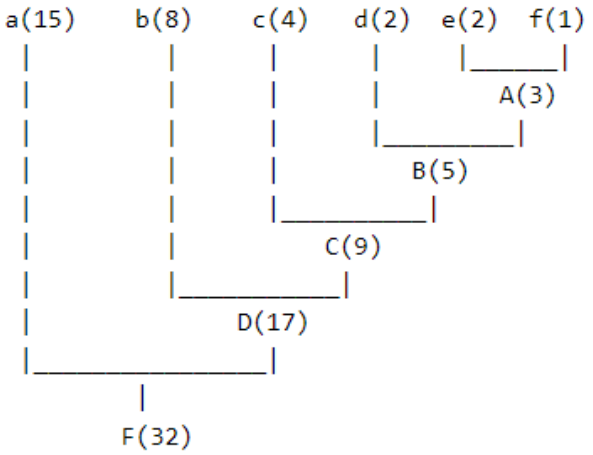
siendo ρ_i la probabilidad del i-ésimo caracter.

En el problema, las frecuencias de los caracteres valen $[2^3, 2, 2^2, 2^2, 2^3, 2^2, 2]$ y, por tanto, sus probabilidades $1/2^5 \cdot [2^3, 2, 2^2, 2^2, 2^3, 2^2, 2]$. Aplicando la fórmula anterior, el valor de la entropía será:

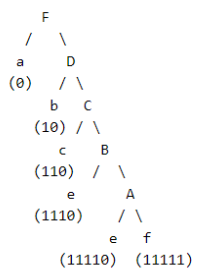
$$\begin{aligned} H &= -1/2^5 \cdot (2^3 \cdot \log_2(2^3/2^5) + 2 \cdot \log_2(2/2^5) + 2^2 \cdot \log_2(2^2/2^5) + 2^2 \cdot \log_2(2^2/2^5) + 2^3 \cdot \log_2(2^3/2^5) + 2^2 \cdot \log_2(2^2/2^5) + 2 \cdot \log_2(2/2^5)) \\ &= -(1/2) \cdot \log_2(1/2^2) - 3 \cdot (1/2^3) \cdot \log_2(1/2^3) - (1/2^3) \cdot \log_2(1/2^4) \\ &= 1 + \frac{9}{8} + \frac{4}{8} = \frac{21}{8} = 2.625 \text{ bits} \end{aligned}$$

Problema 2(c)

Codificación de Huffman



El árbol de Huffman anterior podemos representarlo, según el mismo convenio que el empleado en las transparencias, como:



La profundidad de cada una de las code-words será su tamaño en bits. Por tanto el tamaño del archivo F codificado con el código anterior será $\tau(F)_h = 15 \cdot 1 + 8 \cdot 2 + 4 \cdot 3 + 2 \cdot 4 + 2 \cdot 5 + 1 \cdot 5 = 66$ bits

Codificación de Shanon:

El siguiente algoritmo genera un código de Shanon:

- Crear una lista con los símbolos del alfabeto ordenados por frecuencias decrecientes
- Dividir iterativamente la lista con los caracteres ordenados en parte *superior e inferior* de tal manera que la suma de las probabilidades de los caracteres en ambas partes sea aproximadamente la misma.
  - Asignar un bit 0 a los caracteres de la parte superior y un bit 1 a los de la inferior.
- Iterar el procedimiento sobre las partes superior e inferior mientras estas tengan un tamaño superior a uno.

Evolución paso a paso del algoritmo (se ha seguido la misma *notación* que los ejemplos de las transparencias).

a	15	15					0				
b	8	23	8				1	0			
c	4	27	12	4			1	1	0		
d	2	29	14	6	2		1	1	1	0	
e	2	31	16	8	4	2	1	1	1	1	0
f	1	32	17	9	5	3	1	1	1	1	1

Breve explicación

En la primera iteración la *parte de arriba* estará formada por un único caracter $\{a\}$ con frecuancia acumulada 15 y la parte de abajo por los caractres $\{b, c, d, e, f\}$ con frecuencia acumulada $32 - 15 = 17$. Al caracter a se le asigna el bit 1 y al resto el bit 0. El proceso se repite sobre las partes cuya tamaño sea superior a uno, en este caso, por tanto, **solo** la *parte inferior*. En la segunda iteración del algoritmo, la *parte de arriba* estará formada por un único

Breve explicación

En la primera iteración la *parte de arriba* estará formada por un único caracter $\{a\}$ con frecuancia acumulada 15 y la parte de abajo por los caractres $\{b, c, d, e, f\}$ con frecuencia acumulada $32 - 15 = 17$. Al caracter a se le asigna el bit 1 y al resto el bit 0. El proceso se repite sobre las partes cuya tamaño sea superior a uno, en este caso, por tanto, **solo** la *parte inferior*. En la segunda iteración del algoritmo, la *parte de arriba* estará formada por un único caracter $\{b\}$ con frecuancia acumulada 8 y la parte de abajo por los caractres $\{c, d, e, f\}$ con frecuencia acumulada $17 - 8 = 9$. Al caracter b se le asigna el bit 1 y al resto el bit 0. Como la longitud de la *parte inferior* es superior a 1, continua la iteración sobe esta parte....

Tamaño del archivo codificado con el código anterior $\tau(F)_s = 15 \cdot 1 + 8 \cdot 2 + 4 \cdot 3 + 2 \cdot 4 + 2 \cdot 5 + 1 \cdot 5 = 66$ bits