

SolucionesFinal2023.pdf



msm_EPS



Algoritmia y Estructuras de Datos Avanzadas



2º Grado en Ingeniería Informática



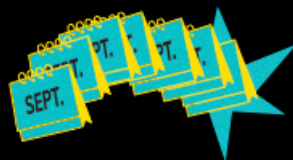
Escuela Politécnica Superior
Universidad Autónoma de Madrid



Quando creías
que lo habías
visto ~~todo~~

en internet_ 🔍


LLEGA BIZUM Y TE PASA UNOS
APUNTES PARA QUE APRENDAS
CÓMO COMPRAR ONLINE

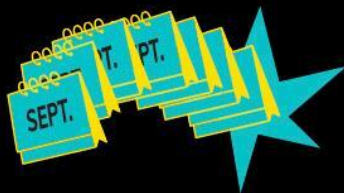


TEMA 1

- Elige "Pagar con Bizum"
- Introduce tu número
- Valida tu compra



 bizum



TEMA 1

- Elige "Pagar con Bizum"
- Introduce tu número
- Valida tu compra



Soluciones Final 2023

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 $[23, 2, 22, 22, 23, 22, 2]$

y, por tanto, sus probabilidades $1/25 \cdot [23, 2, 22, 22, 23, 22, 2]$

. Aplicando la fórmula anterior, el valor de la entropía será:

$$H = -1/25 \cdot (23 \cdot \log_2(23/25) + 2 \cdot \log_2(2/25) + 22 \cdot \log_2(22/25) + 22 \cdot \log_2(22/25) + 23 \cdot \log_2(23/25) + 2 \cdot \log_2(2/25) + 2 \cdot \log_2(2/25)) = -(1/2) \cdot \log_2(1/22) - 3 \cdot (1/23) \cdot \log_2(1/23) - (1/23) \cdot \log_2(1/24) = 1 + 98 + 48 = 218 = 2.625 \text{ bits}$$

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 lista de aristas. Cada vez que se procesa una arista (u,v)

se invoca a la función $\text{union}(u, v)$ del conjunto disjunto donde los parámetros de la función son los nodos u, v que forman la arista. (Nota: asumimos que $\text{union}(u, v)$ sobre dos elementos cualesquiera u e v

- invoca a $\text{find}(u)$ y a $\text{find}(v)$, sino fuese así en el algoritmo había que invocar a la función con los parámetros $\text{union}(\text{find}(u), \text{find}(v))$
- Al finalizar del algoritmo habrá tantas componentes conexas como subconjuntos en el Conjunto Disjunto.

- ```
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 disjoint subsets **Sx**

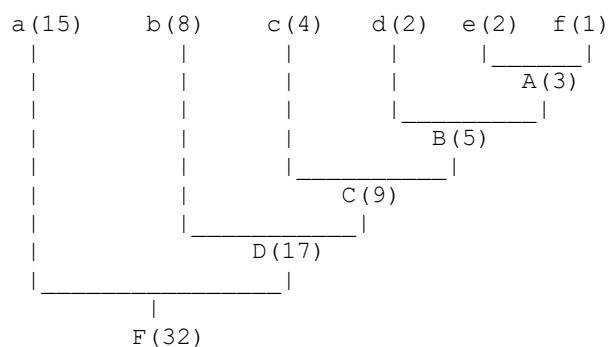
S = init (G[V]) ...

	{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}				

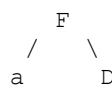
- Tal y como se observa en la tabla anterior al finalizar el algoritmo tenemos un único subconjunto y, por tanto, una única componente conexas.

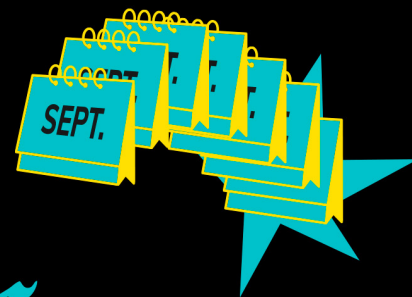
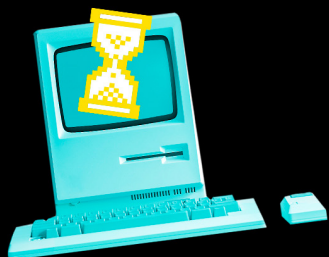
Problema 2(c)

Codificación de Huffman



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



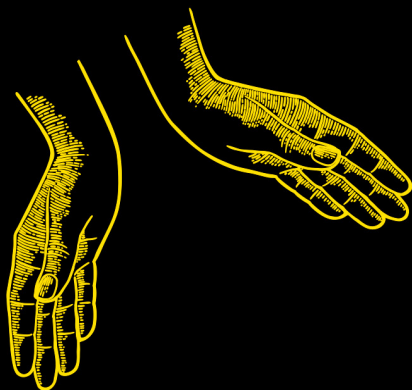


Cuando creías que lo habías visto ~~todo~~

en internet_



LLEGA BIZUM Y TE PASA UNOS
APUNTES PARA QUE APRENDAS
CÓMO COMPRAR ONLINE



TEMA 1

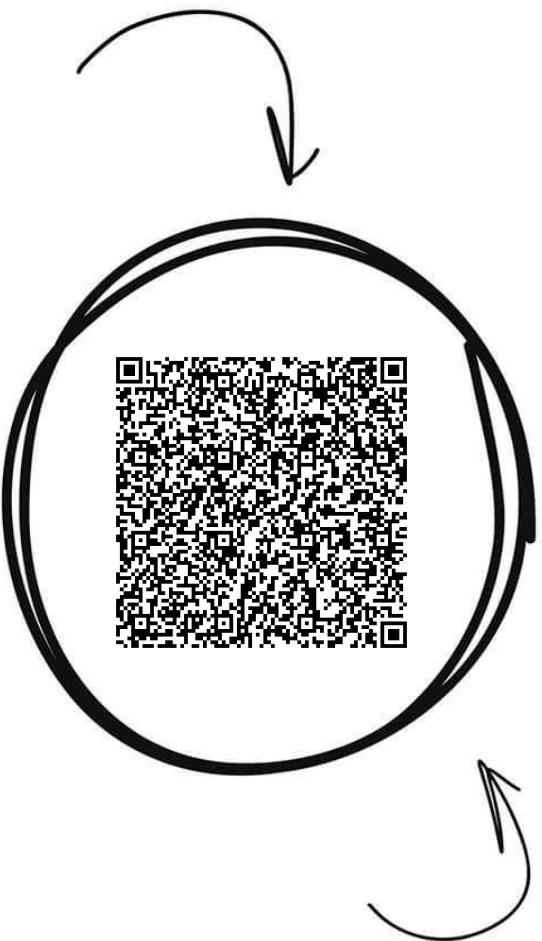
- Elige “Pagar con Bizum”
- Introduce tu número
- Valida tu compra



Algoritmia y Estructuras de...



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



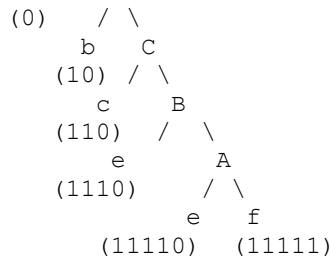
Banco de apuntes de la

WUOLAH

- 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





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).

*abcdef*15842211523272931328121416174689245230111110111
1011101101

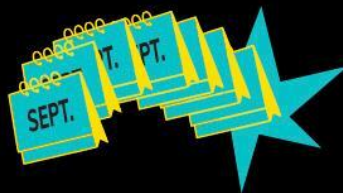
Breve explicación

En la primera iteración la *parte de arriba* estará formada por un único carácter $\{a\}$

con frecuencia acumulada 15 y la parte de abajo por los caracteres $\{b,c,d,e,f\}$ con frecuencia acumulada $32 - 15 = 17$. Al carácter 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 carácter $\{b\}$ con frecuencia acumulada 8 y la parte de abajo por los caracteres $\{c,d,e,f\}$ con frecuencia acumulada $17 - 8 = 9$. Al carácter b se le asigna el bit 1 y al resto el bit 0

Quando creías
que lo habías
visto ~~todo~~

en internet_



LLEGA BIZUM Y TE PASA UNOS
APUNTES PARA QUE APRENDAS
CÓMO COMPRAR ONLINE



TEMA 1

- Elige "Pagar con Bizum"
- Introduce tu número
- Valida tu compra



. Como la longitud de la *parte inferior* es superior a 1, continua la iteración sobre 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

Parcial 2

Problema (1a), 2 pts

Solución:

Dada una tabla con los tiempos de descubrimiento y finalización de los nodos del grafo, el **teorema del paréntesis** nos permite hallar de que tipo es cada una de arista del grafo con complejidad $O(1)$

:

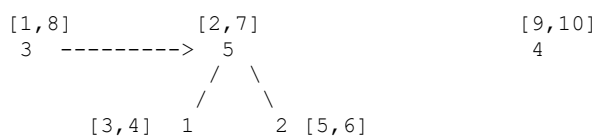
- Arista (4,5)

: Los intervalos de tiempo descubrimiento/finalización de los nodos 4 y 5, $I_4 = [9, 10]$ y $I_5 = [2, 7]$ respectivamente, son disjuntos. Es decir, $I_4 \cap I_5 = \emptyset$

• . Por tanto, los nodos 5 y 4 pertenecen a árboles (o sub-árboles) dfs diferentes. Al conectar la arista (4,5) nodos de diferentes árboles dfs, es una arista **de cruce**.

• Arista (5,3): El intervalo de tiempo del nodo 5, $I_5 = [2, 7]$, está incluido en el intervalo de tiempo del nodo 3, $I_3 = [1, 8]$, es decir, $I_5 \subset I_3$

- . Por tanto, el nodo 5 es descendiente del nodo 3 en un árbol dfs siendo la arista (5,3) **hacia atrás (ascendente o backward)**



Problema 1(b), 4 pts.

Solución

- Al ser G

un grafo **no dirigido** hay que aplicar el **algoritmo DFS**. Tras su ejecución, **cada uno de los árboles del bosque dfs representará una componente conexa**. En este caso

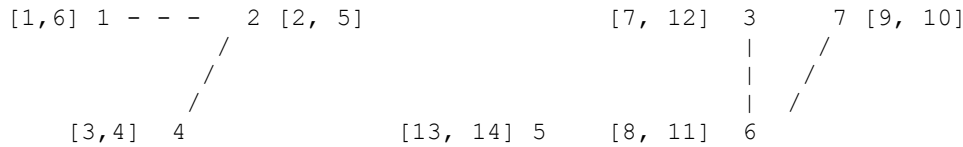


 bizum

WUOLAH

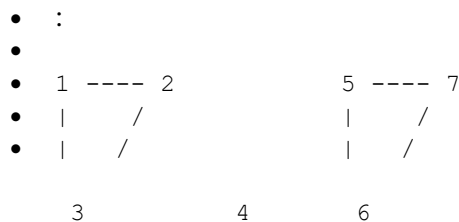
tendremos 3 árboles dfs y, por tanto, tres componentes conexas $\{1,2,4\}$, $\{5\}$ y $\{3,6,7\}$. Entre paréntesis se muestran los tiempos de descubrimiento y finalización de los nodos de G

- Las aristas se han procesado en orden lexicográfico.



- Para que la matriz de adyacencia sea diagonal se deberá crear un grafo G'

isomorfo a G tal que los nodos que pertenezcan a una misma componente conexa tengan índices consecutivos. Por ejemplo, el grafo no dirigido $G'=\{V,E\}$ tal que $V=\{1,2,3,4,5,6\}$ y $E=\{(1,2),(1,3),(2,3),(5,6),(5,7),(6,7)\}$ es isomorfo a G



El grafo G'

tiene como matriz de adyacencia

$$A_{(*)} = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

o si se asume que los nodos están conecados consigo mismo (tal y como se hace habitualmente):

$$A = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Problema 1(c)

Aplicar los algoritmos QuickSelect y QuickSelect5 a la tabla inferior para buscar su undécimo elemento

Solución:

Sea la tabla $S=[6,4,14,8,13,2,10,7,11,1,9,15,12,5,3]$

y k el undécimo elemento ($k=11$) en la ordenación de la tabla. En la solución que presentamos se ha seguido el criterio de Python donde los índices de las tablas $0,...,(n-1)$ siendo n

es el tamaño de la tabla.

QuickSelect:

```
[4, 2, 1, 5, 3] , 6, [14, 8, 13, 10, 7, 11, 9, 15, 12]
pivot = 6
index_pivot = 5
Como k > (index_pivot + 1), (11 > 6), se actualiza el valor de k = 11
- (5+1) = 5, y se ejecuta la recursión sobre la segunda sub-tabla

[8, 13, 10, 7, 11, 9, 12] , 14, [15]
pivot = 14
index_pivot = 7
k < (index_pivot + 1), (5 < 8)

[7] , 8, [13, 10, 11, 9, 12]
pivot = 8
index_pivot = 1
k > (index_pivot + 1), (5 < 2), se actualiza el valor de k = 3

Como el tamaño de la sub-tabla [13, 10, 11, 9, 12] es <= 5 (cutoff),
se sale de la recursión, la tabla se ordena [9, 10, 11, 12, 13] y se
devuelve el valor correspondiente a la posición indicada por k.
En este caso la tercera posición (k=3), cuyo valor es 11
```

QuickSelect5:

```
k = 11

Cálculo del pivote: [6, 4, 14, 8, 13, 2, 10, 7, 11, 1, 9, 15, 12,
5, 3]
medianas= 8, 7, 9
Mediana_de_medianas = 8

[6, 4, 2, 7, 1, 5, 3] , 8, [14, 13, 10, 11, 9, 15, 12]
pivote = 8
index_pivot = 7
k > (index_pivot + 1), (11 > 8). Se actualiza el valor de k = 3

Calculo del pivote: [14, 13, 10, 11, 9, 15, 12]
medianas = 11, 12(*)
Mediana_de_medianas = 11

[10, 9] , 11, [14, 13, 15, 12]
pivote = 11
index_pivot = 2
Como k == (index_pivot + 1), (3 == 3), se devuelve el pivote
```