Cerrar

Información

Un turista desea visitar todas las capitales europeas en sus próximas vacaciones. Para ello piensa emplear vuelos directos en avión, sin escalas, como único medio de transporte. Suponga que el turista ha desarrollado en su análisis un grafo G= <V, A>, dirigido y ponderado, donde V representa las ciudades y A los vuelos. El grafo G estará representado como una matriz de adyacencia de dimensión n, siendo n el número de ciudades a visitar. La celda en la posición i, j representará el tiempo del vuelo que conecta las ciudades i y j. Si no existe conexión entre las ciudades i y j, el valor de la celda será -INF. Teniendo en cuenta únicamente ese tiempo, el turista desea conocer si existe o no un recorrido tal que le permita visitar todas las ciudades exáctamente una única vez empleando un tiempo en vuelo inferior a un valor t, que es pasado por argumento al algoritmo que resuelve el problema. El turista parte siempre desde una ciudad c en su viaje, que es pasada también por argumento al algoritmo que resuelve el problema.

Información

Emplee para resolver el siguiente ejercicio una sintaxis que sea lo más parecida al lenguaje LEA (el lenguaje empleado en las diapositivas y textos del curso). Utilice letras en minúscula para los nombres de funciones y variables de más de un carácter de longitud. Emplee letras en mayúscula para los nombres de un solo carácter. Preste atención especial al sangrado de las instrucciones para representar los bloques de instrucciones. Emplee uno o varios caracteres de espacio en blanco para realizar el sangrado. No emplee el tabulador. Puede emplear la siguiente notación, en caso de ser necesaria, para representar algunos de los símbolos especiales del lenguaje:

- Asignación: <-
- Infinito: INF
- Conjunto vacío: VACIO
- Unión de conjuntos: U
- Operadores lógicos: Y (conjunción), O (disyunción), ! (negación)
- Disyunción lógica: O
- Operadores relacionales: <= (menor o igual), >= (mayor o igual), != (distinto)
- Producto cartesiano (separación de argumentos): x
- Recorrer los elementos de un conjunto (bucle "foreach"): para todo E en C
- Potencia: ^

Funciones especiales que no hace falta implementar:

- inserta(E, L): inserta el elemento E en el conjunto o al final de la lista L.
- elimina(E, L): elimina el elemento E del conjunto o la lista L.
- obtiene(L): obtiene el primer elemento de la lista L.
- extrae(L): obtiene el primer elemento de la lista L y lo elimina de ésta.
- ordena(L): ordena los elementos de la lista L en orden creciente.
- ordena-decreciente(L): ordena los elementos de la lista L en orden decreciente.
- min(A, B), max(A, B): devuelen el menor y mayor de los valores A y B.

Algoritmo de ejemplo:

```
nombre-de-funcion: arg1 x arg2 -> ret1
  i <- arg1 + arg2
  mientras i <= 10
    i <- i + 1
  ret1 <- i

algoritmo: A x B -> C
  inserta(A, B)
  C <- 0
  paratodo E en B
    C <- C + nombre-de-funcion(E, 2)
  M <- FALSO
  si C^2 = 10 Y !M
    C <- INF
```

/

Sin responder aún

Puntúa como 1,50

guía la plantilla que se proporciona a continuación, **a la que podrá añadir y quitar todas las instrucciones y/o variables que considere oportunas**. Revise que la salida del algoritmo se corresponda con lo solicitado en el enunciado como respuesta.

```
valor: S x A x n -> r
...

completable: S x A x k -> r
...

turista: A x n x k x S x M -> S x M
  si ...
    si valor(..., A, ...) > valor(..., A, ...)
      ...
  si-no
    desde i <- 0 hasta ...
      S[k] <- ...
      si ...(..., A, ...)
        S, M <- turista(A, n, ..., S, M)
```

```
valor: S x A x n -> r
 r <- 0
 desde i <- 1 hasta n-1
    r += G[S[i]][S[i+1]]

completable: S x G x k -> r
 r <- true
 desde i <- 1 hasta k
    r = (G[S[i]][S[i+1]] != -INF)

turista: G x n x k x S x M x c x t -> S x M
si k = 1
 S[k] <- c
      si completable(S, A, k)
        S, M <- turista(A, n, k + 1, S, M, c)
si no
  si k = n
    si valor(S, A, n) > valor(M, A, n)
      M <- S
  si-no
    desde i <- 0 hasta n
      S[k] <- i
      si completable(S, A, k)
        S, M <- turista(A, n, k + 1, S, M, c)
```

Sin responder aún

Puntúa como 0,50

Párrafo

En el peor de los casos tendríamos un orden temporal de O(n^2), ya que

Ruta: p

Pregunta **3**

Sin responder aún

Puntúa como 0,50

Realice un análisis de la complejidad espacial del algoritmo en el peor de los casos.

Párrafo

En el análisis temporal tendríamos en el peor caso un Orden de O(n^2) ya que este almacena una matriz de vuelos almacenar la solución disponemos de dos vectores de n ciudades, ambos con O(n).

Ruta: p

◄ Plantilla en blanco traza A star

Ir a...

¿NECESITAS AYUDA?

CAU Campus Virtual

campus.virtual@uca.es

Horario de 9:00 a 14:00 L-V
956 01 67 55

Canal YouTube