Cerrar

Información

Un alto ejecutivo desea visitar todas las sedes de una empresa. El ejecutivo desea obtener un trayecto que le ahorre el máximo dinero posible en los desplazamientos. Para ello ha calculado el coste que le supondría el viaje entre cada dos sedes de la empresa. Esta información la anota en un grafo G=<V, A>, que es cíclico, ponderado y completo, donde V representa las sedes y A los trayectos entre las sedes. El grafo G estará representado como una matriz de costes de dimensión n, siendo n el número de sedes a visitar. La celda en la posición i, j representará el coste que le supone al ejecutivo viajar desde la sede i a la j. El coste del camino inverso, esto es, de la sede j a la sede i puede ser diferente. El ejecutivo decide visitar las sedes en función del coste que le supone viajar a la siguiente, priorizando por tanto las que les suponen un trayecto de menor coste desde la que se encuentra visitando en ese momento. Es decir, que la segunda sede en visitar será la que le suponga un menor coste en el trayecto desde la primera, la tercera será la que le suponga un menor coste desde la segunda, y así sucesivamente. Suponga que el ejecutivo desea visitar las sedes exactamente una única vez y que siempre parte de una de las sedes de la empresa en su recorrido, que es dada siempre como parámetro del problema.

Pregunta **1**

Sin responder aún

Puntúa como 0,50

Según el esquema general de los algoritmos voraces,

Seleccione una:

- La función objetivo consiste en maximizar el número de sedes de la empresa que el ejecutivo puede visitar y el objetivo es maximizar.
- La función de factibilidad descartará sedes de la empresa que se encuentren en el conjunto de candidatos seleccionados.
- Ninguna de las otras respuestas es correcta.
- La función de selección incluirá entre sus instrucciones una llamada a la función de factibilidad para descartar la selección de sedes de la empresa previamente visitadas.

Quitar mi selección

Pregunta **2**

Sin responder aún

Puntúa como 0,50

¿Es la estrategia devoradora propuesta en el enunciado óptima? Demuéstrelo. En caso de no serlo puede demostrarlo empleando un contraejemplo.

Párrafo

No, la estrategia no es óptima ya que al elegir la sede más cercana del momento, no tenemos en cuenta los costes de via futuro. La estrategia óptima sería aquella que minimice los costes de pasar por todas las sedes con el menor coste total, p puede darse el caso de que el coste de pasar por todas las sedes no sea el menor.

Pongamos un ejemplo:

Sede 1 a 2: 5

Sede 2 a 3: 7

Sede 1 a 3: 8

Sede 3 a 2: 2

Si partimos de la 1ra sede, la estrategia devoradora escogería de 1 a 2, y de 2 a 3, que tendría un coste total de 13

En cambio la mejor estrategia sería partir de la sede 1 a la 3, y de la 3 a la 2, que tendría un coste total de 10

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 1,00

desarrollado debe ser lo más eficiente posible. Una solución de orden de complejidad temporal superior al requerido se considerará incorrecta.

```
//G: Grafo representado por matriz de costes
//Primera_Sede: Primera sede en la que parte el alto ejecutivo
//Sedes: Conjunto/Lista de sedes que debe de recorrer el alto ejecutivo

Alto_Ejecutivo: G x Primera_Sede x Sedes -> S
S <- VACIO
C <- Sedes
P <- Primera_Sede
C <- C - P            //Eliminamos la sede de la que parte del conjunto de candidatos
mientras C != 0
     M <- selecciona_sede(C, P, G)
     C <- C - M
     S <- S U M
     P <- M

//Selecciona la siguiente sede con menor coste de la que se encuentra
Selecciona_Sede: C x Sede_Anterior x G -> K
aux <- INF
paratodo i en C
   si(G[Sede_Anterior][i] < aux)
       aux <- G[Sede_Anterior][i]
       K <- i
```

Pregunta **4**

Sin responder aún

Puntúa como 0,50

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

Párrafo

Sea n = |C| es decir, al conjunto de sedes

- El bucle principal se realizará n -1 veces en el peor de los casos.

- La función de selección realizará en el peor de los casos n - 1 comparaciones.

Por lo tanto tendría un coste de O(n^2) en el peor caso.

Ruta: p

de problemas (grupo B1)

¿NECESITAS AYUDA?

CAU Campus Virtual

campus.virtual@uca.es

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

Canal YouTube

/