

RepartidorDeBebidas.pdf



Anónimo



Estructuras de Datos no Lineales



2º Grado en Ingeniería Informática



Escuela Superior de Ingeniería
Universidad de Cádiz

A promotional banner for the MSI Claw Aim. The background is a vibrant, abstract digital cityscape with blue and purple hues. On the left, the text 'CLAW AIM' is prominently displayed in a bold, white, sans-serif font. Below it, a tagline reads 'Para jugársela tanto como tú estudiando el día de antes.' in a smaller, white font. In the center, a black MSI Claw Aim handheld device is shown at an angle, displaying a colorful dragon-like graphic on its screen. To the right of the device, there is a purple button with the text 'COMPRAR AHORA' and a QR code below it. The MSI logo is visible in the top right corner of the banner.

CLAW AIM

Para jugársela tanto como tú estudiando el día de antes.

msi

COMPRAR AHORA



Si estás más cansado
que un perezoso
con anemia, aquí
tienes tu push.

DIA

REPARTIDOR DE BEBIDAS (2022)

// Un repartidor de una empresa de distribución de bebidas
// tiene que visitar a todos sus clientes cada día.
// Pero, al comentar su jornada de trabajo, no conoce que cantidad de bebidas
// tiene que servir cada cliente, por lo que no puede planificar una ruta
// óptima para visitarlos a todos. Por tanto, nuestro repartidor decide llevar
// a cabo la siguiente estrategia:

// - El camión parte del almacén con la máxima carga permitida rumbo
// a su cliente más próximo
// - El repartidor descarga las cajas de bebidas que le pide el cliente.
// si no tiene suficientes cajas en el camión, le entrega todas las que tiene.
// Este cliente terminará de ser servido en algún otro momento a lo largo del día
// cuando la estrategia de reparto vuelva a llevar al repartidor hasta él
// - Después de servir a un cliente:
// - Si quedan bebidas en el camión, el repartidor consulta su sistema de navegación
basado en el GPS
// para conocer la ruta que le lleva hasta su cliente más próximo pendiente de ser
servido
// - Si no quedan bebidas en el camión, vuelve al almacén por el camino más corto y
otra vez carga
// el camión completamente
// - Después de cargar el camión, el repartidor consulta su sistema de navegación y se va
por el camino más corto
// a visitar al cliente pendiente de ser servido más próximo

// Implementa un subprograma que calcule y devuelva, la distancia total recorrida en un
día por nuestro repartidor
// a partir de lo siguiente :
// - Grafo representado mediante la matriz de costes con las distancias de los caminos
directos entre
// los clientes y entre ellos y la central
// - Capacidad máxima del camión (cantidad de cajas de bebidas)
// - Asumiremos que existe una función `int Pedido()` que devuelve el número de cajas
que quedan por servir
// al cliente en el que se encuentra el repartidor

// NOTA: Es absolutamente necesario definir todos los tipos de datos implicados en la
resolución del problema ,
// así como los prototipos de las operaciones utilizadas de los TADS conocidos y también
los prototipos de los
// algoritmos de grafo utilizados de los estudiados en las asignaturas

Ponte un caféeeé
con aroma del DIA

¡Compra ya!



*Dia recomienda
el consumo
responsable de
bebidas con
caféina

WUOLAH

```
float calcularDistanciaTotal(const GrafoP<tCoste>& distanciasDirectas, const int
capacidadMaximaCamion) {
```

```
    matriz <vertices> P;
    matriz <tCoste> m = Floyd(distanciasDirectas, P);
    int cajasDisponiblesCamion = capacidadMaximaCamion;
    int numTotalParadas = distanciasDirectas.numVert();
    vertice paradaActual = 0, destino; // asumimos que 0 es el almacen
    tCoste valorMinimo = INF; //coste a la ciudad mas cercana
    vector<bool> ciudadesAbastecidas(distanciasDirectas.numVert(), false);
    int i;
    float distanciaTotalRecorrida = 0 ;

    while (cajasDisponiblesCamion > 0) {
        i = 1;
        //while (paradaActual != 0 && i != numTotalParadas ){
        while (i != numTotalParadas) {
            if (m[paradaActual][i] < valorMinimo) {
                if (paradaActual != i && !ciudadesAbastecidas[i]) {
                    valorMinimo = m[paradaActual][i];
                    destino = i;
                    i++;
                }

                if (destino.Pedido() <= cajasDisponiblesCamion) {
                    cajasDisponiblesCamion = cajasDisponiblesCamion - destino.Pedido();
                    ciudadesAbastecidas[destino] = true;
                } else {
                    destino.Pedido() = destino.Pedido() - cajasDisponiblesCamion;
                    cajasDisponiblesCamion = 0;
                }
                distanciaTotalRecorrida += m[paradaActual][destino];

                if (cajasDisponiblesCamion == 0) {
                    distanciaTotalRecorrida += m[paradaActual][0];
                    paradaActual = 0;
                    cajasDisponiblesCamion = capacidadMaximaCamion;
                } else paradaActual = i;
            }
        }
    }
}
```

```
    return distanciaTotalRecorrida;  
}
```