Algoritmos y Estructuras de Datos (VJ1215) - Universitat Jaume I

Examen final - 2024/2025 - Primera parte
Recuperación del tercer examen parcial

5 de junio de 2025

Nombre:

El examen final consta de dos partes, cada una con una duración de dos horas. La primera parte está destinada a aquellos estudiantes que han solicitado intentar mejorar la calificación obtenida en cualquiera de los exámenes parciales, renunciando a la misma. Este tercer enunciado de la primera parte está dirigido a quienes buscan mejorar la nota del tercer examen parcial.

La prueba es individual. No puedes consultar libros, apuntes ni dispositivos electrónicos. Escribe tu solución empleando el lenguaje C++, y no olvides los costes que se piden. Al finalizar entrega tu solución junto con el enunciado (no es necesario que entregues todas las hojas). Pon tu nombre en cada hoja que entregues.

En cada uno de los siguientes dos ejercicios, debes decidir qué algoritmo utilizar e implementarlo en C++, añadiendo el método que se pide a la siguiente clase utilizada para representar grafos dirigidos en los ejercicios de la asignatura. No puedes añadir ahí otros atributos.

```
class GrafoDirigido {
   struct Arco {
      int vecino;
      float peso;
      Arco * siguiente;
      Arco(int, float, Arco *);
   };
   struct Vertice {
      Arco * primerArcoDeEntrada;
      Arco * primerArcoDeSalida;
      int gradoDeEntrada;
      int gradoDeSalida;
      Vertice();
   };
   vector<Vertice> vertices;
 public:
   ...
};
```

Ejercicio 3 — 1,25 puntos

Consideremos un grafo dirigido $G = (V, E)$, que puede contener ciclos. Implementa el siguiente método:

```
bool estaEnCiclo(int v) const
```

Este método debe devolver *true* si el vértice `v` forma parte de algún ciclo en el grafo. En caso contrario, debe devolver *false*.

En esta implementación, no puedes utilizar la clase `queue`, ni implementarla, ni recurrir a vectores u otras estructuras de datos para suplir la funcionalidad de una cola.

Para que tu solución sea válida, debe ser eficiente. Además, se valorará que el algoritmo detenga el recorrido del grafo tan pronto como sea posible determinar el resultado.

Indica cuál es el siguiente coste de tu solución en función de $|V|$ (cantidad de vértices) y $|E|$ (cantidad de arcos). No es necesario justificarlo.

Coste temporal en el peor caso	$O($ $)$

Consideremos un grafo dirigido ponderado $G = (V, E)$, que modela el escenario de un juego con $n \geq 2$ jugadores. Cada arco tiene un peso no negativo y el grafo puede contener ciclos.

Inicialmente, cada jugador j (para j desde 0 hasta $n-1$) comienza en el vértice *posiciones*$[j]$. Es posible que varios jugadores compartan la misma posición.

El objetivo de cada jugador es alcanzar el vértice t. El peso de un camino se define como la suma de los pesos de sus arcos. La distancia del jugador al objetivo es el peso del camino de mínimo peso desde su posición inicial hasta t. Si un jugador ya se encuentra en t, su distancia es 0.

Para ajustar la dificultad del juego, es necesario determinar si la diferencia entre la distancia del jugador más cercano al objetivo y la del más alejado supera un umbral preestablecido.

Implementa el siguiente método:

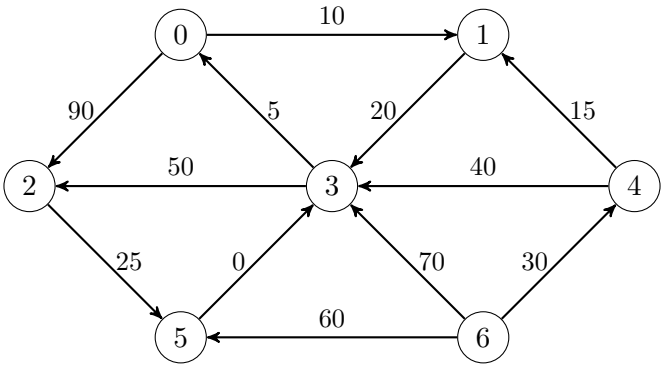
```
bool esBrechaDeDistanciaAceptable(int t, const vector<int> & posiciones, float umbral) const
```

Este método debe:

- Devolver *true* si la diferencia entre la distancia del jugador más cercano y la del más lejano no supera el umbral.
- Devolver *false* si la diferencia es mayor que el umbral.
- Lanzar una excepción si algún jugador no puede alcanzar el objetivo.

Puedes utilizar cualquier estructura de datos que necesites, sin tener que implementarla.

Por ejemplo, dados $t = 2$, *posiciones* $= [4, 6, 0, 6]$, y el siguiente grafo:



- El jugador en 4 está a distancia 85 del objetivo, por el camino $4 \to 1 \to 3 \to 2$.
- Los jugadores en 6 están a distancia 110, por el camino $6 \to 5 \to 3 \to 2$.
- El jugador en 0 está a distancia 80, por el camino $0 \to 1 \to 3 \to 2$.

La diferencia entre la distancia del jugador más cercano y la del más lejano es $110 - 80 = 30$. Si el umbral fuese 30 o 40, el método devolvería *true*. Si el umbral fuese 20, el método devolvería *false*.

Para que tu solución sea válida, debe ser eficiente. Además, se valorará que el algoritmo detenga el recorrido del grafo tan pronto como sea posible determinar el resultado.

Indica cuál es el siguiente coste de tu solución en función de n, $|V|$ y $|E|$. No es necesario justificarlo.

Coste espacial en el peor caso sin contar el grafo	$O($ $)$