Algoritmos y Estructuras de Datos (VJ1215) - Universitat Jaume I
Evaluación continua - 2022/2023

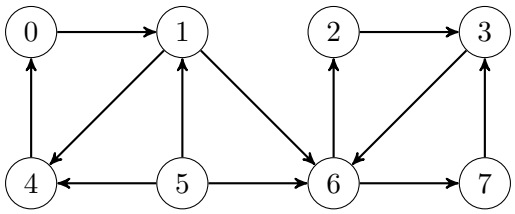
16 de diciembre de 2022

Nombre:

La duración de esta prueba es de 80 minutos. La prueba es individual. No puedes consultar libros, apuntes ni dispositivos electrónicos. Al finalizar entrega tu solución junto con el enunciado (no es necesario que entregues todas las hojas). Pon tu nombre en todo lo que entregues.

Ejercicio 1 — 2 puntos

Aplica a este grafo el algoritmo de Kosaraju-Sharir. Dibuja los bosques que representan la exploración del grafo y la pila que se obtiene como resultado de la primera etapa. ¿Qué problema resuelve el algoritmo?



Ejercicio 2 — 8 puntos

En los siguientes dos apartados, debes decidir qué algoritmo utilizar para resolver eficientemente el problema planteado, e implementarlo 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. Escribe tus soluciones empleando el lenguaje C++. Puedes hacer uso de los Tipos Abstractos de Datos *Pila*, *Cola* y *Cola de Prioridad*, con las operaciones que necesites, sin tener que implementarlas (puedes poner sus nombres en inglés o en castellano). No puedes hacer uso de otros métodos o funciones si no los implementas.

```
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:
   ...
};
```

En los dos apartados, diseña una solución que no siga recorriendo el grafo si, con lo visto hasta ese momento, ya se puede determinar el resultado.

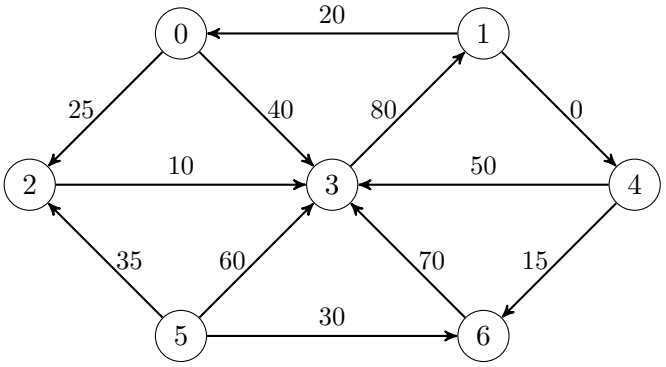
a) **[4 puntos]** Sea $G = (V, E)$ un grafo dirigido ponderado que representa posibles desplazamientos en un juego. En el juego participan n jugadores, que están situados en algunos de los vértices del grafo y deben alcanzar un objetivo situado en un vértice t. El grafo puede tener ciclos. El peso de cada arco $(u,\ v)$ es una cantidad no negativa que se corresponde con la energía que consume el jugador al moverse del vértice u al vértice v utilizando ese arco. Al desplazarse desde un vértice hasta otro siguiendo un camino en el grafo se consume una energía dada por la suma de los pesos de los arcos utilizados.

Implementa un método eficiente

```
int contarJugadoresCercanos(int t, float e, const vector<int> & jugadores) const
```

que recibe el vértice t, una cantidad de energía $e > 0$ y un vector de talla n que contiene los vértices en los que están situados los jugadores. El método debe devolver la cantidad de jugadores que pueden llegar a t consumiendo una energía total menor o igual que e. La energía que consumen para llegar a t los jugadores que ya están en t es 0. Puede haber varios jugadores en el mismo vértice.

Por ejemplo, con el siguiente grafo



con $t = 3$, $e = 45$ y $jugadores = [1, 5, 3, 4, 5, 0, 3, 4, 5]$ el resultado sería 6 porque cada uno de los 6 jugadores que están en los vértices 0, 3 y 5 puede llegar hasta el vértice 3 consumiendo una cantidad de energía menor o igual que 45, mientras que los jugadores que están en los vértices 1 y 4 necesitarían una energía superior a 45 para llegar al vértice 3.

b) **[4 puntos]** Si supieras que todos los arcos del grafo tienen el mismo peso, ¿qué algoritmo utilizarías para resolver el problema más eficientemente en ese caso? Impleméntalo.