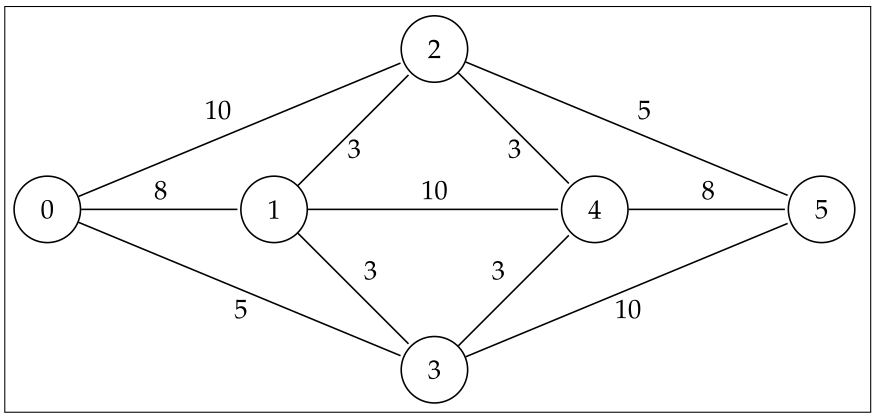
Este es el examen final del curso *Programación Imperativa Modular (PIMO)*, 2014-1. El examen tiene 10 preguntas y otorga un total de 70 puntos. El examen es *individual* y no es permitido el uso de libros, apuntes o equipos electrónicos.

Nombre y código: ____________________

Pregunta	1	2	3	4	5	6	7	8	9	10	**Total**
Puntos	10	10	10	10	10	10	10	10	10	10	100
Puntaje											

1. Considere el siguiente grafo $G = (V, E, w)$ con función de peso $w : E \to \mathbb{N}$:



   (a) (1 punto) ¿Es G un grafo dirigido?
   (b) (1 punto) ¿Es G acícliclo?
   (c) (4 puntos) Dibuje una matriz de adyacencia para (V, E, w).
   (d) (4 puntos) Dibuje una lista de adyacencia para (V, E, w).

2. (10 puntos) Encuentre un árbol mínimo de cubrimiento para el grafo G (del numeral 1). Nombre el algoritmo usado y exhiba una simulación con la cual validar su respuesta.

3. (10 puntos) Determine la distancia mínima entre cualquier par de vértices de G (del numeral 1). Nombre el algoritmo usado y exhiba una simulación con la cual validar su respuesta.

4. (10 puntos) En cada una de las situaciones descritas a continuación, modifique el grafo G (del numeral 1) para que cumpla con las condiciones dadas:
   (a) Cambie el signo de la función de peso para al menos dos de los arcos en E de tal manera que el grafo resultante contenga un ciclo de costo 0.
   (b) Cambie el signo de la función de peso para exactamente un arco en E de tal manera que el grafo resultante contenga al menos dos ciclos de costo negativo.

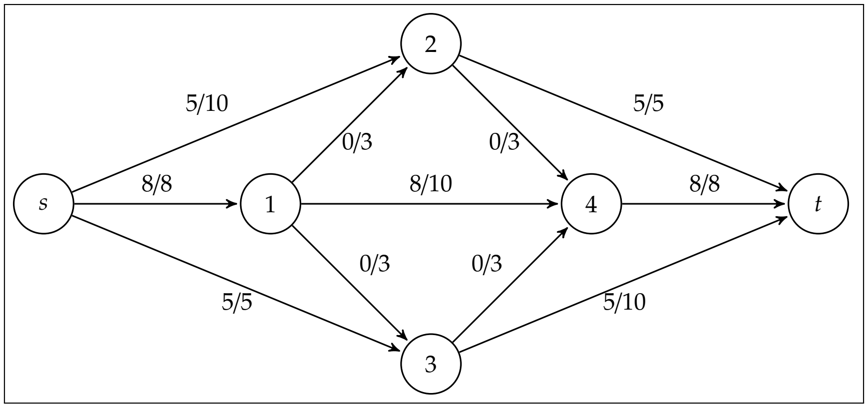
   Dibuje el grafo resultante en en cada uno de los casos y justifique sus respuestas.

5. Sea $G' = (V, E', w')$ el grafo dirigido que resulta de G (del numeral 1) de la siguiente manera:

$$E' = \{(u, v) \in E \mid u < v\} \qquad \text{y} \qquad w'(e) = w(e) \quad \text{para } e \in E'.$$

   (a) (3 puntos) Dibuje G'.
   (b) (1 punto) ¿Es G' acíclico?
   (c) (6 puntos) De ser posible, encuentre un orden topológico para G'. De lo contrario, explique por qué dicho orden no existe. En cualquier caso justifice claramente su respuesta.

6. Considere el siguiente grafo $G = (V, E, s, t, w)$, que representa una red de flujo con fuente s, destino t y capacidad w, y el siguiente flujo $f : E \to \mathbb{N}$ para G:



Por ejemplo, del vértice 1 al vértice 4 la capacidad es 10 y el flujo es 8.

(a) (7 puntos) Dibuje el grafo residual G_f.

(b) (3 puntos) Calcule el flujo de s a t con respecto a f.

7. Considere el grafo G del numeral 6.

(a) (5 puntos) Calcule un flujo máximo de s a t. Nombre el algoritmo usado y exhiba una simulación con la cual validar su respuesta.

(b) (5 puntos) Dibuje el grafo residual de G correspondiente al flujo máximo calculado en la parte (a).

8. (10 puntos) Sea $A[0..N)$ un arreglo de números enteros, con $N \geq 0$. Diseñe un algoritmo de complejidad temporal $O(N)$ que calcule la máxima suma entre todos los subarreglo de A. Justifique su respuesta.

9. (10 puntos) Diseñe una estructura de datos `dforest` para representar conjuntos disyuntos con *path compression* y *ranking*. En particular, exhiba pseudo-código para las funciones `find(x)` que calcula representante de x y `union(x,y)` que realiza la unión de los conjuntos a los cuales x e y pertenecen.

10. (10 puntos) El *arbitramiento* es el uso de discrepancias en tazas de cambio para transformar una unidad de una moneda en más de una unidad de la misma moneda. Por ejemplo, suponga que 1 dólar estadounidense compra 49 rupias indias, 1 rupia india compra 2 yenes japoneses y 1 dolar japonés compra 0.0107 dolares estadounidenses. Entonces, al hacer cambios de divisa, alguien puede iniciar con 1 dólar estadounidense y comprar $49 \cdot 2 \cdot 0.0107 = 1.0486$ dólares estadounidenses, obteniendo una ganancia de 4.86%.

Suponga que cuenta con n divisas $c_0, c_2, \ldots, c_{n-1}$ y una tabla $R[0..n)[0..n)$ definiendo las tazas de cambio, de tal forma que una unidad de la divisa c_i compra $R[i][j]$ unidades de la divisa c_j.

Diseñe un algoritmo eficiente que determine si existe o no una secuencia de divisas $\left\langle c_{i_1}, c_{i_2}, \ldots, c_{i_k} \right\rangle$ tal que

$$R[i_1][i_2] \cdot R[i_2, i_3] \cdots R[i_{k-1}][i_k] \cdot R[i_k][i_1] > 1.$$

Analice el tiempo de ejecución de su algoritmo.