

## Pregunta 1

¿Cuál es la complejidad del algoritmo de Edmonds-Karp? Probarlo. (NOTA: EN LA PRUEBA SE DEFINEN UNAS DISTANCIAS, Y SE PRUEBA QUE ESAS DISTANCIAS NO DISMINUYEN EN PASOS SUCEсивOS DE EK. UD. PUEDE USAR ESTO SIN NECESIDAD DE PROBARLO)

### Solución

Completar prueba

## Pregunta 2

Probar que si, dados vértices  $x, z$  y flujo  $f$ , definimos la distancia relativa a  $f$  como la longitud del menor  $f$ -camino aumentante entre  $x$  y  $z$  (si existe), o infinito si no existe, o 0 si  $x = z$ , denotándola como  $d_f(x, z)$ , y definimos  $d_k(x) = d_{f_k}(s, x)$ , donde  $f_k$  es el  $k$ -ésimo flujo en una corrida de Edmonds-Karp, entonces  $d_k(x) \leq d_{k+1}(x)$ .

### Solución

Para llegar a una contradicción, supongamos que existe un vértice  $z$  tal que  $d_k(z) > d_{k+1}(z)$ , y sea

$$A = \{y : d_{k+1}(y) < d_k(y)\}$$

un conjunto **no vacío**.

Y sea  $y \in A$  tal que:

$$d_{k+1}(y) = \min d_{k+1}(x) : x \in A$$

es decir que  $y$  es el vértice más cercano a  $s$  en el flujo  $f_{k+1}$ , entre todos los vértices de  $A$ .

Notar que:

- Como  $d_{k+1}(y)$  es el mínimo sobre los elementos de  $A$ :

$$d_{k+1}(x) < d_{k+1}(y) \Rightarrow x \notin A \Rightarrow d_k(x) \leq d_{k+1}(x) \quad (1)$$

- $y \neq s$  ya que  $d_k(s) = d_{k+1}(s) = 0 \Rightarrow s \notin A$ .
- $d_{k+1}(y) < \infty$  ya que  $y \in A \Rightarrow d_{k+1}(y) < d_k(y) \leq \infty$ . Por lo tanto *existe un camino aumentante (en  $f_{k+1}$ ) entre  $s$  e  $y$  y en dicho camino hay al menos un vértice además de  $y$ .*

Tomando el camino de menor longitud y  $x$  vértice inmediatamente anterior a  $y$  en ese camino, tenemos que:

$$d_{k+1}(y) = d_{k+1}(x) + 1 \quad (2)$$

Por lo tanto,  $d_{k+1}(x) < d_{k+1}(y)$ , y vale la ecuación 1.

$$d_k(x) \leq d_{k+1}(x)$$

y así:

$$d_k(y) > d_{k+1}(y) = d_{k+1}(x) + 1 \geq d_k(x) + 1 \quad (3)$$

Ahora bien, si  $\overrightarrow{xy}$  estuviese disponible en el paso  $k$ , tendríamos que  $d_k(y) \leq d_k(x) + 1$ , ya que, si se puede llegar a  $x$  a  $y$  en un solo paso, entonces el camino mas corto a  $y$  no puede ser mas largo que tomar el camino mas corto a  $x$  y luego dar el paso hacia  $y$  y por lo tanto  $d_k(y) \leq d_k(x) + 1$ . Pero esto contradice la ecuación 3. Entonces, **el lado  $\overrightarrow{xy}$  no puede haber estado disponible en el paso  $k$ .**

Pero aún así,  $\overrightarrow{xy}$  forma parte de un camino de  $s$  a  $y$  en el paso  $k + 1$ , pero si no está disponible en el paso  $k$ , pero si en el paso  $k + 1$  es debido a que *se usó como backward en el paso  $k$ .*

1. Luego del paso  $k - 1$ , quedó  $f_{k-1} = c(\overrightarrow{xy})$ .
2. En el paso  $k$ , se usó como backward, por lo tanto se disminuye el flujo en  $f_k$  en  $\overrightarrow{xy}$ , es decir,  $f_k(\overrightarrow{xy}) < c(\overrightarrow{xy})$ .
3. En el paso  $k + 1$ , se usó como forward.

Por lo tanto usamos  $\overleftarrow{xy}$  en el paso  $k \Rightarrow$  tenemos un  $f_k$ -camino aumentante de la forma  $s, \dots, \overleftarrow{xy}$ , entonces, en el paso  $k$ . Y además  $d_k(x) = d_k(y) + 1$  (**ya que estamos usando EK**).

$$\Rightarrow d_k(y) > d_k(x) + 1 = d_k(y) + 1 + 1 = d_k(y) + 2 \Rightarrow 0 > 2$$

Y acá llegamos al **absurdo**, que vino de suponer que el conjunto  $A$  no era vacío. Por lo tanto,  $A$  es vacío y por lo tanto  $d_k(x) \leq d_{k+1}(x)$  para todo  $x$ .

### Pregunta 3

Probar que si, dados vértices  $x, z$  y flujo  $f$ , definimos la distancia relativa a  $f$  como la longitud del menor  $f$ -camino aumentante entre  $x$  y  $z$  (si existe), o infinito si no existe, o 0 si  $x = z$ , denotándola como  $d_f(x, z)$ , y definimos  $b_k(x) = d_{f_k}(x, t)$ , donde  $f_k$  es el  $k$ -ésimo flujo en una corrida de Edmonds-Karp, entonces  $b_k(x) \leq b_{k+1}(x)$ . (*Este teorema solo se tomará a partir de diciembre 2025*).

#### Solución

Completar prueba

### Pregunta 4

¿Cuál es la complejidad del algoritmo de Dinic? Probarla en ambas versiones: Dinic original y Dinic-Even. (*No hace falta probar que la distancia en redes auxiliares sucesivos aumenta*).

#### Solución

Completar prueba

### Pregunta 5

¿Cuál es la complejidad del algoritmo de Wave? Probarla. (*No hace falta probar que la distancia en redes auxiliares sucesivos aumenta*).

#### Solución

Completar prueba

### Pregunta 6

Probar que la distancia en redes auxiliares sucesivos aumenta. (*Este teorema solo se tomará a partir de diciembre 2025*).

#### Solución

Completar prueba

### Pregunta 7

Probar que si  $f$  es un flujo maximal, entonces existe un corte  $S$  tal que  $v(f) = \text{cap}(S)$ . (*Puede usar sin necesidad de probarlo que si  $f$  es flujo y  $S$  es corte, entonces  $v(f) = f(S, \bar{S}) - f(\bar{S}, S)$* ).

## Solución

Definamos

$$S = \{s\} \cup \{x \in V : \text{exista un } f\text{-camino aumentante desde } s \text{ a } x\}$$

- Como  $f$  es maximal entonces no existen  $f$ -caminos aumentantes desde  $s$  a  $t$  pues si existiese un tal  $f$ -camino aumentante, podríamos mandar un  $\varepsilon > 0$  a través de él, obteniendo un flujo  $f^*$  tal que  $v(f^*) = v(f) + \varepsilon > v(f)$  lo cual contradice que  $f$  sea maximal. Por lo tanto, como no existen  $f$ -caminos aumentantes desde  $s$  a  $t$  entonces  $t \notin S$ .
- Como  $s \in S$  y  $t \notin S$ , entonces  $S$  es un corte.

Observemos que en el resto de la prueba no usaremos que  $f$  es maximal, solo que es flujo y que  $S$  es corte. Es decir, la prueba valdría para cualquier  $f$  tal que el  $S$  definido sea un corte. Esto será importante luego.

Como  $f$  es flujo y  $S$  es corte, entonces  $v(f) = f(S, \bar{S}) - f(\bar{S}, S)$ . Calculemos  $f(S, \bar{S})$  y  $f(\bar{S}, S)$ .

1. Cálculo de  $f(S, \bar{S})$ :

$$f(S, \bar{S}) = \sum_{x,y} f(\vec{xy})[x \in S][y \notin S][xy \in E]$$

Consideremos un par  $x, y$  de los que aparecen en esa suma. Como  $x \in S$ , entonces existe un  $f$ -camino aumentante entre  $s$  y  $x$ , digamos  $s = x_0, x_1, \dots, x_r = x$ . Pero como  $y \notin S$ , entonces no existe ningún  $f$ -camino aumentante entre  $s$  e  $y$ . En particular, el camino

$$s = x_0, x_1, \dots, x_r = x, y$$

**no es un  $f$ -camino aumentante.** Pero  $\vec{xy} \in E$ , así que debería poder serlo.

¿Por qué  $s = x_0, x_1, \dots, x_r = x, y$  no es un  $f$ -camino aumentante a pesar de que  $s = x_0, x_1, \dots, x_r = x$  si lo es y  $\vec{xy}$  existe? La única razón por la cual no es un  $f$ -camino aumentante es porque no podemos usar el lado  $\vec{xy}$  por estar saturado, es decir:

$$f(\vec{xy}) = c(\vec{xy})$$

Esto es cierto para cualesquiera  $x, y$  que aparezcan en esa suma. Entonces:

$$\begin{aligned} f(S, \bar{S}) &= \sum_{x,y} f(\vec{xy})[x \in S][y \notin S][xy \in E] \\ &= \sum_{x,y} c(\vec{xy})[x \in S][y \notin S][xy \in E] \\ &= c(S, \bar{S}) = \text{cap}(S) \end{aligned}$$

2. Cálculo de  $f(\bar{S}, S)$ :

$$f(\bar{S}, S) = \sum_{x,y} f(\vec{xy})[x \notin S][y \in S][xy \in E]$$

Consideremos un par  $x, y$  de los que aparecen en esa suma. Como  $y \in S$ , entonces existe un  $f$ -camino aumentante entre  $s$  e  $y$ , digamos  $s = x_0, x_1, \dots, x_r = y$ . Pero como  $x \notin S$ , entonces no existe un  $f$ -camino aumentante entre  $s$  y  $x$ . En particular

$$s = x_0, x_1, \dots, x_r = y, x$$

NO ES un  $f$ -camino aumentante. Pero  $\vec{xy} \in E$ , así que PODRÍA serlo, usando  $y, x$  como lado backward.

¿Por qué  $s = x_0, x_1, \dots, x_r = y, x$  no es un  $f$ -camino aumentante a pesar de que  $s = x_0, x_1, \dots, x_r = y$  si lo es y  $\vec{xy}$  existe? La única razón es que no podemos usarlo como lado backward por estar vacío, es decir, que  $f(\vec{xy}) = 0$ .

Esto es cierto para cualesquiera  $x, y$  que aparezcan en esa suma. Entonces:

$$\begin{aligned} f(\bar{S}, S) &= \sum_{x,y} f(\overrightarrow{xy})[x \notin S][y \in S][xy \in E] \\ &= \sum_{x,y} 0[x \notin S][y \in S][xy \in E] \\ &= 0 \end{aligned}$$

Entonces probamos que para este  $S$ :

- $f(S, \bar{S}) = \text{cap}(S)$
- $f(\bar{S}, S) = 0$

Por lo tanto

$$\begin{aligned} v(f) &= f(S, \bar{S}) - f(\bar{S}, S) \\ &= \text{cap}(S) - 0 = \text{cap}(S) \end{aligned}$$

## Pregunta 8

Probar que si  $G$  es conexo y no regular, entonces  $\chi(G) \leq \Delta(G)$ .

### Solución

Sea  $G$  un grafo conexo con grado máximo  $\Delta(G)$ . Si  $G$  es un ciclo impar o un grafo completo, es conocido que  $\chi(G) = \Delta(G) + 1$ . Supongamos que  $G$  no es ninguno de estos casos y probemos que  $\chi(G) \leq \Delta(G)$ .

Primero vamos a plantear la estrategia de la demostración:

1. Elegimos un vértice  $x$  de grado mínimo  $\delta$ .
2. Luego, con ese vértice ejecutamos BFS, para generar un orden de los vértices.
3. Tomamos el orden inverso al obtenido y coloreamos usando Greedy.
4. Mostramos que en cada paso hay a lo sumo  $\Delta - 1$  colores ocupados en los vecinos anteriores.
5. Con  $\Delta$  colores disponibles, siempre hay un color libre.

Elegimos un vértice arbitrario  $x$  (tal que  $d(x) = \delta(x)$ ) y ejecutamos un *BFS* desde  $x$ . Esto nos da un árbol generador de  $G$  y un orden en el que se descubren los vértices. Denotemos este orden como:

$$y_1, y_2, \dots, y_n, \quad \text{donde } y_1 = x.$$

Invertimos este orden y llamamos a la secuencia resultante:

$$x_1, x_2, \dots, x_n, \quad \text{donde } x_n = x.$$

En este nuevo orden, salvo por  $x_n = x$ , cada vértice  $x_i$  tiene al menos un vecino posterior en la secuencia. Esto es porque en el BFS cada vértice (salvo la raíz) es descubierto por algún vecino anterior en el BFS, lo que se traduce en un vecino posterior en el orden invertido.

Utilizamos un coloreo greedy en el orden  $x_1, x_2, \dots, x_n$ :

- Asignamos el color 1 a  $x_1$ .

- Para cada  $x_i$  con  $i > 1$ , elegimos el menor color disponible que no haya sido asignado a un vecino anterior en el orden.

Observamos que, al momento de colorear  $x_i$ , la cantidad de colores que ya están usados en sus vecinos anteriores es a lo sumo  $\Delta(G) - 1$ . Esto se debe a que  $x_i$  siempre tiene al menos un vecino posterior en el orden, por lo que no todos sus  $\Delta(G)$  vecinos pueden haber sido coloreados antes que él.

Dado que hay  $\Delta(G)$  colores disponibles, siempre hay un color libre para cada vértice.

El coloreo greedy nunca usa más de  $\Delta(G)$  colores, por lo que  $\chi(G) \leq \Delta(G)$ , como queríamos demostrar.

## Pregunta 9

Probar que 2-COLOR es polinomial.

### Solución

Para probar esto, vamos a dar un algoritmo, que cumpla lo siguiente:

1. Resuelve el problema de 2-color.
2. Corre en tiempo polinomial.

La idea es dar primero el algoritmo, luego probar que es polinomial y por último en la **correctitud**, probar que el algoritmo da respuestas correctas, es decir, funciona.

Antes de comenzar con la prueba, vamos a dar una observación que nos será útil mas adelante:

- $\chi(G) \leq 2 \iff \chi(C) \leq 2 \quad \forall c.c \in C$  **del grafo  $G$ , es decir, si cada componente conexa del grafo tiene número cromático menor o igual a 2, entonces el grafo en sí también lo tiene.**

[−] Para resolver el problema de 2-color, vamos a usar el algoritmo de BFS, que nos permite colorear un grafo de manera eficiente. La idea es la siguiente:

- Tomo un vértice  $x$  cualquiera en un grafo conexo  $G$ .
- Construyo un árbol generador  $T$  de  $G$  usando BFS con raíz en  $x$ .
- Ejecutar BFS desde  $x$  y determinar el nivel de cada vértice  $z$ .
- Luego asigna los colores según el nivel de los vértices, si el nivel es impar, asigno color 1, si es par, asigno color 2.
- Por último verifico que no haya aristas que conecten vértices del mismo color, si las hay, entonces el grafo no es 2-coloreable.

[2.] Ahora tenemos que probar la complejidad, como ya sabemos el colorear es gratis, el peso de complejidad recae en el BFS y en el chequeo de las aristas, ambos corren en tiempo  $O(m)$  donde  $m$  es la cantidad de aristas, por lo tanto, el algoritmo corre en tiempo polinomial.

Ahora bien, si el algoritmo anterior devuelve **Sí**, es porque el coloreo es propio, y por lo tanto  $\chi(G) \leq 2$ . En caso contrario,  $\chi(G) \geq 3$ .

Para probar esto, vamos a ver que el grafo  $G$  **tiene un ciclo impar**. Para comenzar, voy a plantear la estrategia de la demostración:

[1] Por hipótesis, tenemos que el coloreo del *ciclo for* de nuestro algoritmo **no es propio**. Esto quiere decir que existe una arista  $zv \in E(G)$  tal que  $c(z) = c(v)$ , en este dato vamos a basar la demostración.

---

**Algorithm 1** Verificar si un grafo es bipartito (2-color)

---

```
1: Entrada: Grafo  $G = (V, E)$  conexo
2: Salida: "Sí" si  $G$  es bipartito, "No" en caso contrario
3: Seleccionar un vértice inicial  $x \in V(G)$ 
4: Construir un árbol generador  $T$  de  $G$  usando BFS con raíz en  $x$ 
5: Ejecutar BFS desde  $x$  y determinar el nivel de cada vértice  $z$ 
6: for cada vértice  $z \in V(G)$  do
7:   if nivel de  $T(z)$  es impar then
8:     Asignar  $c(z) \leftarrow 1$ 
9:   else
10:    Asignar  $c(z) \leftarrow 2$ 
11:   end if
12: end for
13: for cada arista  $(u, v) \in E(G)$  do
14:   if  $c(u) = c(v)$  then
15:     Retornar "No" {El grafo no es bipartito}
16:   end if
17: end for
18: Retornar "Sí" {El grafo es bipartito}
```

---

1. Considerar el spanning tree  $T$  de  $G$  generado a partir de un vértice  $x$ .
2. Analizar los caminos únicos de  $x$  a  $z$  y  $v$ .
3. Usar la arista  $zv$  para construir un ciclo impar.
4. Probar que el ciclo es impar.

Sea  $T$  el spanning tree con raíz  $x$ . Como  $T$  es un árbol, hay un único camino desde  $x$  a cualquier otro vértice.

- **Camino de  $x$  a  $z$  en  $T$ :**  $xz_1z_2 \dots z_j$  y el nivel de  $z$  es  $j$ .
- **Camino de  $x$  a  $v$ :**  $xv_1v_2 \dots v_i$  y el nivel de  $v$  es  $i$ .

Como el coloreo no es propio, entonces  $c(z) = c(v)$ , lo que implica que ambos son pares o ambos son impares, por lo tanto **la suma de los niveles es par** ( $i+j$  es par).

Dado que ambos caminos empiezan desde el mismo punto  $x$  y terminan en distintos puntos  $z$  y  $v$ , en algún punto se separan en  $T$ , sea  $w$  el último vértice común de ambos caminos, de forma tal que:

$$z_0 = v_0, z_1 = v_1, \dots, z_p = v_p = w$$

Teniendo en cuenta que  $xv$  es un lado de  $G$  y que  $z_k = z$  y  $v_j = v$ , entonces en  $G$  tenemos el ciclo:

$$C = w \underbrace{z_{p+1}z_{p+2} \dots z_{k-1}z}_{k-p \text{ vértices}} \underbrace{vv_{j-1} \dots v_{p+1}}_{j-p \text{ vértices}} w$$

en total  $C$  tiene  $1 + k - p + j - p$  vértices, es decir,  $k + j - 2p + 1$  vértices. Como  $i + j$  es par, entonces  $k + j - 2p + 1$  es impar, por lo tanto  $C$  es un ciclo impar.

## Pregunta 10

Enunciar y probar el Teorema de Hall.

## Solución

Completar prueba

### Pregunta 11

Enunciar y probar el teorema del matrimonio de Kónig.

#### Solución

Completar prueba

### Pregunta 12

Probar que si  $G$  es bipartito entonces  $\chi'(G) = \Delta(G)$ . (Este teorema solo se tomará a partir de diciembre 2025).

#### Solución

Completar prueba

### Pregunta 13

Demostrar las complejidades  $O(n^4)$  y  $O(n^3)$  del algoritmo Húngaro. (Solo a partir de diciembre 2025).

#### Solución

Completar prueba

### Pregunta 14

Enunciar el teorema de la cota de Hamming y probarlo.

#### Solución

Completar prueba

### Pregunta 15

Probar que si  $H$  es matriz de chequeo de  $C$ , entonces:

$$\delta(C) = \min\{j \mid \exists \text{ un conjunto de } j \text{ columnas LD de } H\}$$

(LD significa “linealmente dependiente”).

#### Solución

Completar prueba

### Pregunta 16

Sea  $C$  un código cíclico de dimensión  $k$  y longitud  $n$ , y sea  $g(x)$  su polinomio generador. Probar que:

- i)  $C$  está formado por los múltiplos de  $g(x)$  de grado menor que  $n$ .
- ii)  $C = \{v(x) \cdot g(x) : v(x) \text{ es un polinomio cualquiera}\}$
- iii)  $gr(g(x)) = n - k$
- iv)  $g(x)$  divide a  $1 + x^n$

### Solución

Completar prueba

### Pregunta 17

Probar que 3SAT es NP-completo.

### Solución

Completar prueba

### Pregunta 18

Probar que 3-COLOR es NP-completo.

### Solución

Completar prueba

### Pregunta 19

Probar que Matrimonio3D (matrimonio trisexual) es NP-completo.

### Solución

Completar prueba