# Tópicos Avanzados de Optimización Combinatoria y Teoría de Grafos

*Docentes:* Graciela Nasini, Daniel Severin, Pablo Torres

## Práctica nº 2

1. Pruebe que la cantidad de ciclos hamiltonianos entre $n$ puntos fijos, para $n \geq 3$, es $\frac{(n-1)!}{2}$.

2. Determine el número de operaciones elementales que realiza el *Algoritmo del Vecino más Cercano* para el *Problema del Viajante de Comercio Euclidiano* [LCO, pág. 1] sobre una instancia de $n$ puntos en el plano.

   Considere que los $n$ puntos del plano se ingresan en dos vectores $x$ e $y$, conteniendo las coordenadas de los puntos y que consideramos las siguientes operaciones como elementales:

   - Evaluar una expresión aritmética o booleana.
   - Inicializar/actualizar una variable.
   - Leer/modificar una componente de un vector.

3. Suponga que tiene la posibilidad de utilizar 24 horas de una computadora que realiza una operación elemental en un nanosegundo ($10^{-9}$ segundos). Determine qué tan grande puede ser la instancia del *Problema del Viajante de Comercio Euclidiano* si pretende:

   *a*) resolver el problema por optimalidad chequeando todos los posibles circuitos hamiltonianos (suponemos que el calcular el costo de cada ciclo hamiltoniano es una operación elemental).
   *Sugerencia:* Utilice *Scilab* para el cálculo de factoriales (ver tareas de laboratorio al final de la práctica).

   *b*) obtener una solución factible, utilizando el *Algoritmo del Vecino más Cercano*.

   Responda las mismas preguntas de los apartados (a) y (b) para los casos en que la computadora fuera 10 y 100 veces más rápida.

4. Pruebe que la solución del *Problema del Viajante de Comercio en grafos* que genera el *Algoritmo del Vecino más Cercano* puede alejarse de la óptima tanto como se quiera.
   *Sugerencia:* Pruebe que, para todo $M \geq 8$, existe un grafo de 4 vértices y costos adecuados tales que la solución óptima del problema tiene costo 7 mientras que la solución por el *Algoritmo del Vecino más Cercano* tiene costo $M$.

5. Pruebe que la solución obtenida por el algoritmo *goloso* para el *Problema de Matching Euclídeo (de Peso Mínimo)* [LCO, pág. 4] puede acercarse tanto como se quiera al triple de la solución óptima.

6. Una forma habitual de representar un grafo $G = (V, E)$ con $V = \{1, \ldots, n\}$ en una computadora es mediante los conjuntos $N(1), \ldots, N(n)$ con las vecindades de cada vértice. Determine el orden de complejidad de las siguientes operaciones en términos del tamaño de la entrada:

   *a*) obtener una arista de $G$, o contestar que $G$ no tiene aristas,

   *b*) determinar el vértice de mayor grado de $G$,

   *c*) dado un conjunto $Q \subset V$, verificar si $Q$ es una clique o no,

   *d*) dado un conjunto $B \subset V$ y un entero $k$, reconocer si $B$ es un $k$-empaquetamiento de $G$ o no,

   *e*) dado un conjunto $M \subset V \times V$, reconocer si $M$ es un matching de $G$ o no.

   *Observación:* Recuerde que primero debe contar la cantidad de operaciones elementales asumiendo el peor caso posible [LOC pág. 6]. Considere como operaciones elementales a las nombradas en el Ejercicio 2 más las siguientes:

   - Crear un conjunto vacío.
   - Obtener un elemento de un conjunto, o contestar que es vacío.
   - Agregar/quitar/chequear pertenencia de un elemento de un conjunto.
   - Obtener el cardinal de un conjunto.

7. Las instancias de los problemas de *Matching Perfecto de Peso Mínimo* y *Matching de Peso Máximo* se definen por un grafo $G = (V, E)$ y costos $c \in \mathbb{R}^E$. En el primero, el objetivo es hallar (si existe) un matching perfecto $M \subset E$ de $G$ tal que $\sum_{e \in M} c_e$ es mínimo. En el segundo, el objetivo es hallar un matching $M \subset E$ (no necesariamente perfecto) tal que $\sum_{e \in M} c_e$ es máximo.

   Pruebe que el Problema de Matching Perfecto de Peso Mínimo *se reduce* al Problema de Matching de Peso Máximo. ¿Cuántas operaciones elementales requiere esa *reducción*?

   *Observación:* Decimos que un problema $\Pi_1$ se reduce a un problema $\Pi_2$ cuando, dada cualquier instancia de $\Pi_1$, es posible construir una instancia de $\Pi_2$ tal que, a partir de la solución encontrada para $\Pi_2$, podemos reconstruir la solución de $\Pi_1$.

8. Una instancia del *Problema del Camino más Corto* consiste en un digrafo $D = (V, A)$, costos $c \in \mathbb{R}^A$ y un vértice inicial $a \in V$ y el objetivo es hallar, para cada $v \in V$, el costo mínimo entre los costos de todos los $av$-caminos dirigidos [LMDC, pág. 657; LCO, pág. 20]. Considere las siguientes dos variaciones de este problema y pruebe que ambas pueden reducirse al problema original.

   a) *Entrada*: Un grafo $G = (V, E)$, un vértice $a \in V$ y un vector de costos $c \in \mathbb{R}^E$.
   *Objetivo*: Hallar, para cada $v \in V$, el costo mínimo entre los costos de todos los $av$-caminos.

   b) *Entrada*: Un grafo $G = (V, E)$, un vértice $a \in V$ y un vector de costos $c \in \mathbb{R}^V$.
   *Objetivo*: Hallar, para cada $v \in V$, el costo mínimo entre los costos de todos los $av$-caminos, donde el costo de un camino es la suma de los costos de cada uno de sus vértices.

9. Sean $D = (V, A)$, $c \in \mathbb{R}^A$ y $a \in V$ una instancia del Problema del Camino más Corto tal que para todo $v \in V$ existe al menos un $av$-camino dirigido en $D$. Pruebe que para todo $v \in V$ existe un $av$-camino dirigido de costo mínimo en $D$ si y sólo si $D$ no posee ningún ciclo dirigido $C \subset A$ de costo negativo (es decir, $\sum_{e \in C} c_e < 0$).

10. *a*) Aplique el Algoritmo de Dijkstra sobre la siguiente instancia: $D = (V, A)$ con $V = \{1, 2, 3, 4\}$ y $A = \{(1, 2), (1, 3), (2, 4), (3, 2), (3, 4)\}$, $a = 1$, $c = (60, 40, 20, 10, 90)$.
*Observación:* Recordamos el *Algoritmo de Dijkstra* [LMDC, pág. 660; LCO, pág. 32].

**ALGORITMO DE DIJKSTRA**
**Entrada**: Un digrafo $D = (V, A)$, un vértice $a \in V$ y un vector de costos $c \in \mathbb{R}^A_+$.
**Salida**: Etiquetas de la forma $(L(v), ant(v))$ para todo $v \in V$ tales que $L(v)$ es el costo del $av$-camino dirigido $P$ de costo mínimo y $ant(v)$ es el vértice anterior a $v$ en $P$.

- **Inicialización**: Etiquetamos el vértice $a$ con $(0, -)$ y el resto de los vértices de $D$ con $(+\infty, -)$. También hacemos $V_1 \leftarrow \{a\}$ y $V_2 \leftarrow \varnothing$.
- **Iteración**: Mientras $V_1 \neq \varnothing$ hacer:

*Paso 1.* Tomamos el vértice $v \in V_1$ tal que $L(v)$ es mínimo.

*Paso 2.* Para cada vecino no visitado $w \in N^+(v) \backslash V_2$, si el costo de alcanzar a $w$ pasando por $v$ es menor al costo actual, es decir $L(v) + c_{vw} < L(w)$, entonces actualizamos la etiqueta de $w$ con $(L(v) + c_{vw}, v)$ y agregamos $w$ a $V_1$.

*Paso 3.* Eliminamos $v$ de $V_1$ y lo agregamos en $V_2$.

*b*) Verifique sus cálculos utilizando la rutina `dijkstra` (ver tareas de laboratorio).

*c*) Pruebe que el Algoritmo de Dijkstra no resuelve el Problema del Camino más corto cuando admitimos costos negativos.
*Sugerencia:* Construya una instancia pequeña donde el vector $L$ obtenido al finalizar el algoritmo no contenga los costos de los caminos mínimos.

11. Sean $D = (V, A)$, $c \in \mathbb{R}^A$ y $a \in V$ una instancia del Problema del Camino más Corto. Decimos que $L \in \mathbb{R}^V$ es un *potencial factible* si $L(v) + c_{vw} \geq L(w)$ para todo $vw \in A$ [LCO, pág. 20].

*a*) Pruebe que para todo $ab$-camino dirigido $P$, $L(b)$ es cota inferior del costo de $P$.

*b*) Pruebe que si $L$ es un potencial factible y $P$ un $ab$-camino dirigido tal que $L(b)$ coincide con el costo de $P$, entonces $P$ es un $ab$-camino dirigido de costo mínimo.

12. Pruebe que el *Algoritmo de Ford-Bellman* [LCO, pág. 29] es correcto para resolver el Problema del Camino más Corto (recuerde que para probar la correctitud de un algoritmo se debe probar que, bajo la hipótesis de que la ENTRADA es satisfecha, el algoritmo termina y satisface la SALIDA).

**ALGORITMO DE FORD-BELLMAN**
**Entrada**: Un digrafo $D = (V, A)$, un vértice $a \in V$ y un vector de costos $c \in \mathbb{R}^A$.
**Salida**: Si $i = |V|$ entonces existe un ciclo dirigido con costo negativo. Caso contrario, $L(v)$ contiene los tamaños de los caminos más cortos desde $a$ hasta $v$, para todo $v \in V$.

- **Inicialización**: Etiquetamos el vértice $a$ con $(0, -)$ y el resto de los vértices de $D$ con $(+\infty, -)$. También hacemos $i \leftarrow 0$ y hubo_actualización $\leftarrow Verdadero$.
- **Iteración**: Mientras $i < |V|$ y hubo_actualización, hacer:

*Paso 1.* Hacemos $i \leftarrow i + 1$ y hubo_actualización $\leftarrow Falso$.

*Paso 2.* Para cada arco $vw \in A$, si $L(v) + c_{vw} < L(w)$ entonces actualizamos la etiqueta de $w$ con $(L(v) + c_{vw}, v)$ y hacemos hubo_actualización $\leftarrow Verdadero$.

13. Aplique el Algoritmo de Ford-Bellman para resolver las instancias de los Ejercicios 10.a y 10.c. Compruebe sus iteraciones con `ford_bellman` (ver tareas de laboratorio).

14. Determine el orden de complejidad computacional de los algoritmos de Dijkstra y Ford-Bellman, en términos del tamaño de la instancia.

    *Sugerencias:*

    *a*) Determine la cantidad de operaciones elementales evaluadas en cada Paso de los algoritmos, en función del tamaño de los conjuntos involucrados en ese Paso.

    *b*) Analice, en términos de la cantidad de vértices y aristas, cómo varía el tamaño de los conjuntos que intervienen en cada paso ($V_1$, $V_2$, $N^+$) en el *peor* caso posible (el que genera mayor cantidad de operaciones elementales).

**Tareas de Laboratorio:**

Para la instalación y uso del Scilab, comience realizando los siguientes pasos:

- Descargue el archivo denominado `Instalador-scilab-5.4.1.exe` de la pág. de la Cátedra y ejecútelo. Luego de realizada la instalación, puede borrar este archivo.
- Cree en su máquina un directorio de trabajo y descargue en éste los archivos `dijkstra.sci` y `ford_bellman.sci` de la pág. de la Cátedra.
- Cada vez que ejecute *Scilab*, antes de comenzar a trabajar, elija el en el *Navegador de Archivos* el directorio de trabajo creado en el ítem anterior.

Para trabajar con los ejercicios de la práctica:

- Factoriales (Ej. 3.a): Por ejemplo, para calcular 20! ejecute `factorial(20)`. La respuesta `2.433D+18` significa $2{,}433 \times 10^{18}$.
- Algoritmos de Dijkstra y Ford-Bellman (Ej. 10 y 13): Para verificar la aplicación de los algoritmos, utilice las rutinas `dijkstra` y `ford_bellman`. Por ejemplo, los siguientes comandos resuelven la instancia del Ej. 10.a con Dijkstra:

```
exec('dijkstra.sci',-1)
D(1,:) = [1 1 2 3 3];
D(2,:) = [2 3 4 2 4];
costos = [60 40 20 10 90];
a = 1;
[L, ant] = dijkstra(D, costos, a)
```

  El uso de la rutina de Ford-Bellman es idéntico. Ambas rutinas toman como entrada el digrafo, el cual está representado como una matríz de 2 filas por $m$ columnas en donde cada columna $(u, v)^T$ define un arco que va desde el vértice $u$ a $v$, un vector de costos en donde la componente $i$-ésima se correponde con el arco $i$-ésimo del digrafo, y un vértice inicial. Ambas rutinas retornan los vectores $L$ y $ant$, en donde la componente de $i$-ésima de cada uno de ellos se corresponde con el vértice $i$.

Bibligrafía:

[LMDC] R. Grimaldi. *Matemática Discreta y Combinatoria*. 3ra. edición. Pearson.

[LCO] W. Cook, W. Cunningham, W. Pulleyblank A. Schrijver. *Combinatorial Optimization*. Wiley-Interscience.