# Diseño de algoritmos

## Trabajo Práctico 1 - Algoritmos sobre Grafos

Manuel Latorre FAI-1931
manuel.latorre@est.fi.uncoma.edu.ar

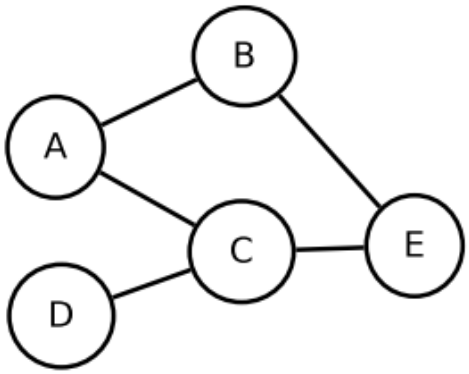
Segundo cuatrimestre 2022


# Índice

# 1. Punto 1

*Describir las ventajas y desventajas con respecto a las implementaciones estática y dinámica de un grafo, considerando las operaciones principales como inserción y eliminación de vértices y arcos, recorridos, etc.*

**Implementación estática:** Un ejemplo de representación estática sencilla para grafos es la matriz de adyacencia, esta consiste en una matriz M de tamaño $N \times N$ de tipo boolean en la cual se puede representar un grafo de N vertices. Si se trata de un grafo dirigido, la celda M[i,j] indica si existe un arco entre el vertice i y el vértice j. Si existe un arco (i,j), la celda esta en true y false en caso contrario. En caso de utilizarse para modelar un grafo no dirigido, para una arista (i,j) se inicializan a true ambas celdas M[i,j] y M[j,i]. por lo que la matriz sera simétrica. En la figura 1 se muestra un ejemplo de un grafo no etiquetado representado con una matriz de adyacencia.




| | A | B | C | D | E |
|---|---|---|---|---|---|
| A | false | true | true | false | false |
| B | true | false | false | false | true |
| C | true | false | false | true | true |
| D | false | false | true | true | false |
| E | false | true | false | false | false |

Figura 1: Representación de grafo no etiquetado con matriz de adyacencia

Este tipo de representación tiene la ventaja del acceso de $O(1)$ a cualquier arco del grafo. Por lo general, es la representación mas adecuada cuando los vertices son del tipo de los indices del arreglo, pero resulta poco practica cuando se necesita almacenar vertices de otro tipo. A medida que las celdas de la matriz tengan mas valores false (es decir que tiene pocos arcos entre los vertices), la matriz ocupara mas espacio del necesario. Las matrices que tienen 30 % o menos de sus celdas en valores distintos de vació, se llaman ralas y siempre se recomienda utilizar una implementación alternativa para estas. Como suele ocurrir con las implementaciones estáticas, tiene la desventaja de ocupar mas espacio de memoria del que generalmente se necesita.

**Implementación dinamica:** Un ejemplo son las listas de adyacencia donde se definen nodos vertices, uno por cada vértice del grafo, que se entrelazan entre ellos formando una lista. Ademas se definen los denominados nodos adyacentes que se enlazan a los nodos vertices conformando una lista que representa todos los vertices adyacentes de un determinado vértice, en la figura 2 se puede ver un ejemplo para un digrafo no etiquetado

En esta representación se pueden destacar las siguientes características:

- La implementación es similar para vertices de cualquier tipo (a diferencia de la implementación estática que, al menos en java, solo seria eficiente para vertices de tipo entero $\geq 0$)

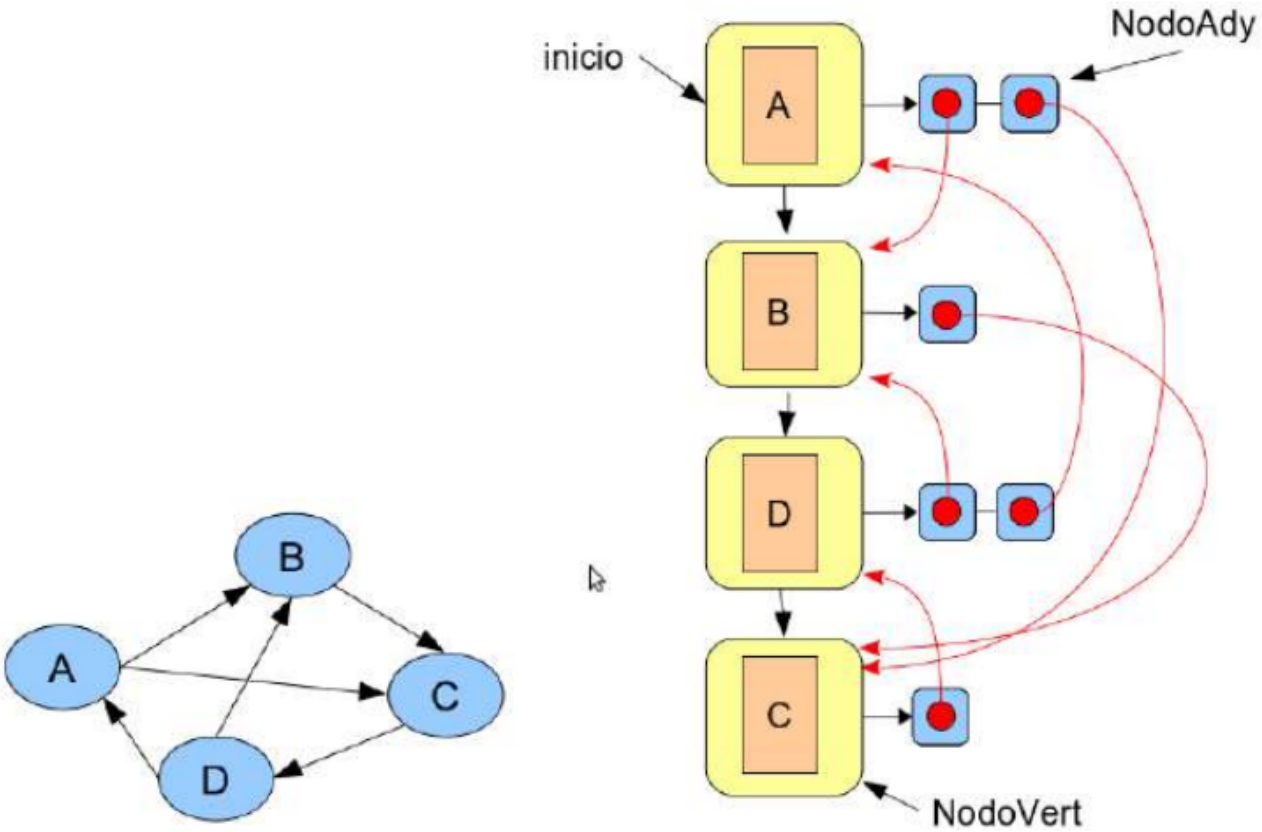



Figura 2: Representación de grafo no etiquetado con listas de adyacencia

- La cantidad de vertices puede crecer y decrecer durante la ejecución, simplemente agregando un nuevo nodo a la lista de vertices
- La lista de nodos adyacentes de cada vértice también puede crecer o decrecer como sea necesario, aunque en este caso a mayor cantidad de adyacentes menor eficiencia para recorrer el grafo y realizar las operaciones correspondientes

# 2. Punto 2

*Resolver gráficamente, utilizando backtracking, el juego de ubicar 4 reinas en un tablero de 4x4.*

Como se puede ver en el grafo (en este caso un árbol) de la figura 3 inicialmente se parte de que el vector se encuentra vació luego se sabe que cumplirá con las restricciones por lo que inserta la reina en la primer posición y crea un nuevo nodo (2) que verificara si se puede insertar una nueva reina, al encontrarse en el nivel 2 del árbol verificara si para la segunda fila del vector se puede insertar alguna reina, en este caso detecta que la columna 3 para la fila 2 cumple con la restricción por lo que inserta la posición de la reina y genera el nodo (3) el cual vuelve a realizar la verificación y se encuentra con que no hay una posición factible para insertar reinas por lo que el algoritmo vuelve al nodo padre de este (hace backtracking). Una vez en el nodo padre busca si es posible insertar una nueva reina en otra columna para la fila 2 y el algoritmo iterara hasta que se llegue a un vector solución o se verifiquen todas las posibles combinaciones.

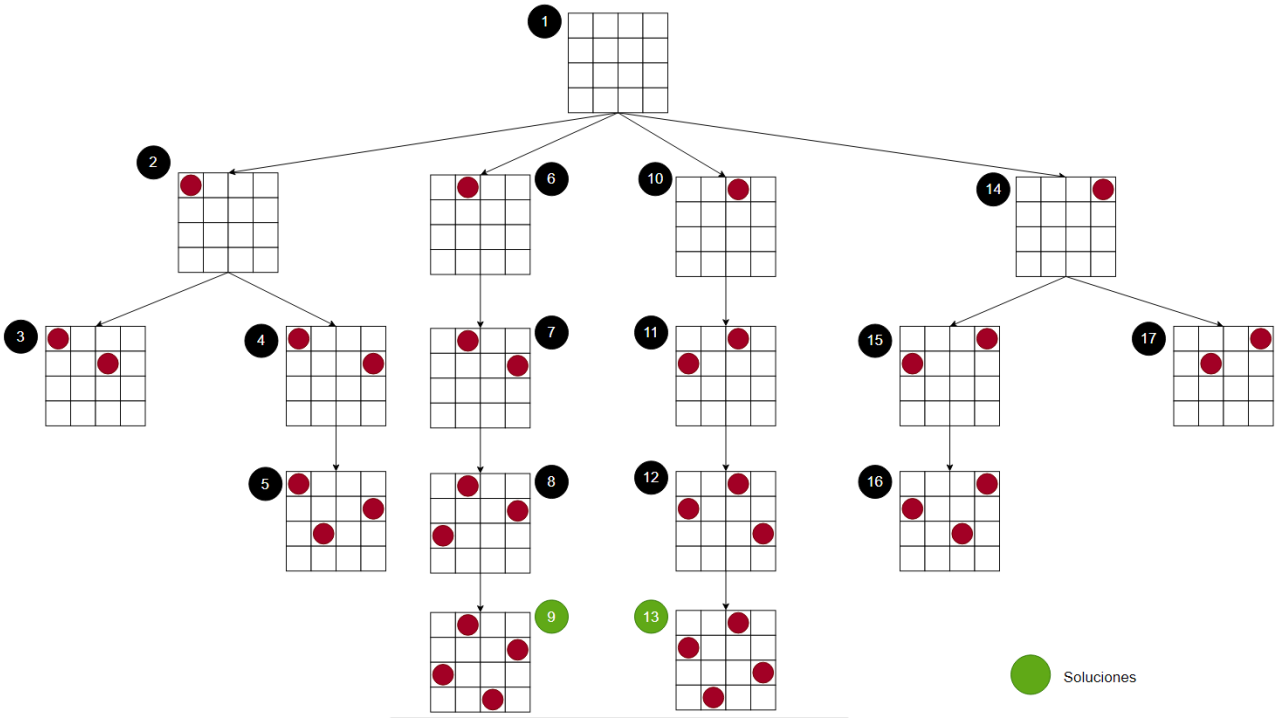



Figura 3: Solución al problema de las 4 reinas para un tablero de 4x4

# 3. Punto 3

*Aplicar la técnica de ramificación y poda al problema de la Asignación de Tareas, utilizando la siguiente tabla con los costos correspondientes:*

| | $t_1$ | $t_2$ | $t_3$ |
|---|---|---|---|
| **A** | 25 | 18 | 15 |
| **B** | 28 | 21 | 30 |
| **C** | 22 | 19 | 23 |

Figura 4: Tabla de costos

Inicialmente se pueden calcular las cotas inferiores y superiores para eliminar posibles casos definiendo un intervalo donde se espera este el resultado, entonces para calcular la cota inferior se tomaran los menores costos de cada fila $A \rightarrow T3 = 15$, $B \rightarrow T2 = 21$, $C \rightarrow T1 = 22$, obteniendo $LB = 15 + 21 + 22 = 58$. Luego para la cota superior se puede calcular la sumatoria de la diagonal principal y de la diagonal secundaria y tomar la que de menor resultado, en este caso $DiagonalPrincipal = 25 + 21 + 23 = 69$ y $DiagonalSecundaria = 15 + 21 + 22 = 58$ es decir que $UB = 58$, finalmente se tendrá que el menor resultado estará en el intervalo $[58, 58]$ por lo que se deduce que el menor resultado sera 58

Luego se construye un árbol donde a partir de una raíz vacía se van a expandir todas las posibles asignaciones de tareas para A y para cada uno de los nuevos nodos expandidos se calculara su cota inferior nuevamente, luego se expandirá el nodo con menor cota inferior hasta que no queden nodos que puedan expandirse ya que todos los restantes son mayores a la cota inferior del ultimo nodo expandido.

Para el desarrollo de la imagen 5 a la hora de calcular la cota inferior de cada nodo se obtuvieron los menores costos para cada tarea, es decir se tomo el menor valor de cada columna

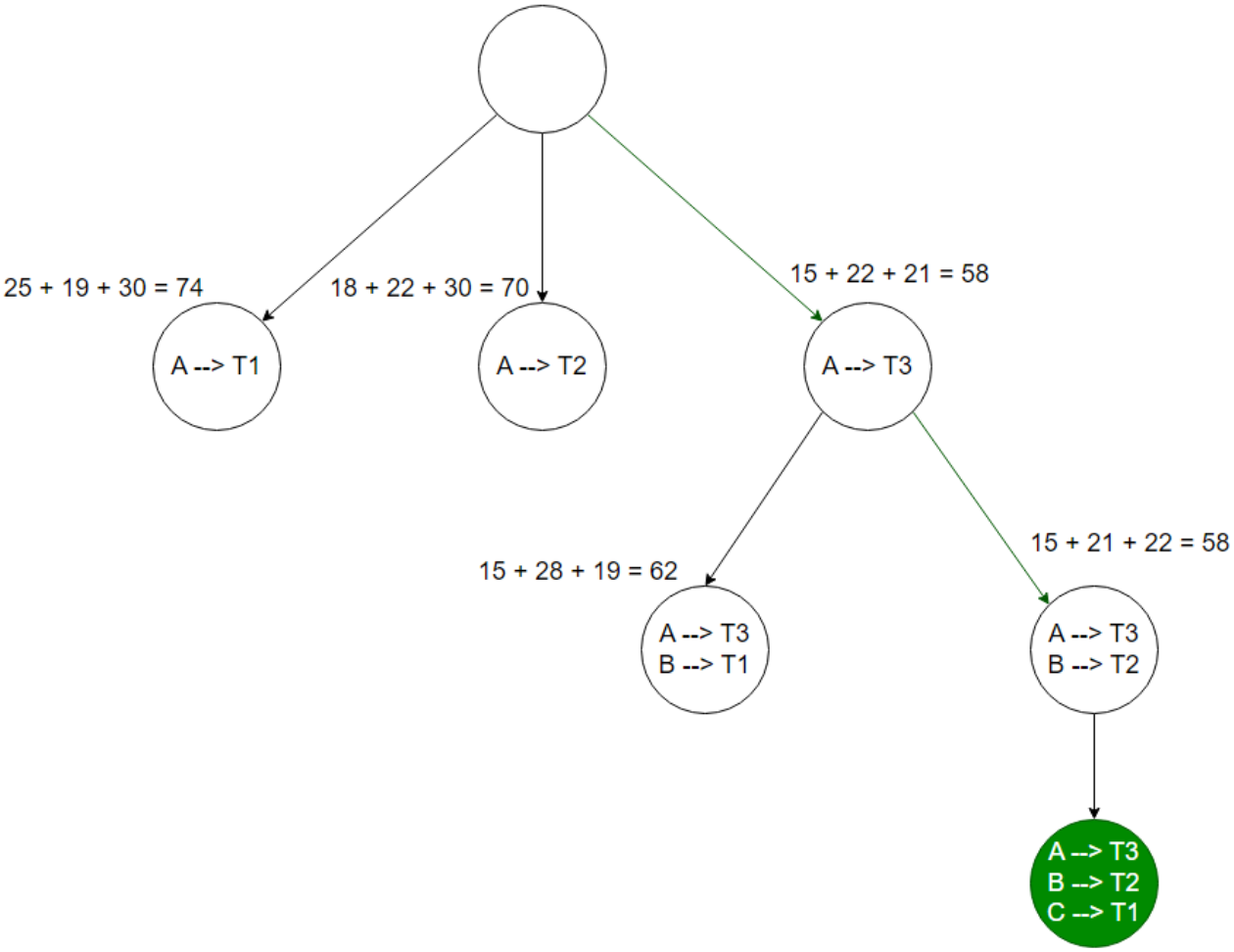



Figura 5: Resolución aplicando técnica de ramificación y poda

# 4. Punto 4

*Dar ejemplos de los siguientes grafos o explicar por qué tales ejemplos no pueden existir.*

1. *Grafo con un circuito hamiltoniano pero sin un circuito euleriano*
2. *Grafo con un circuito euleriano pero sin un circuito hamiltoniano*
3. *Grafo con un circuito hamiltoniano y un circuito euleriano*
4. *Grafo con un ciclo que incluye todos los vértices pero sin un circuito hamiltoniano ni un circuito euleriano*

- **Circuito euleriano:** Es un circuito simple que contiene a todas las aristas de un grafo G
- **Circuito hamiltoniano:** Es un circuito simple que contiene a todos los vértices de un grafo G

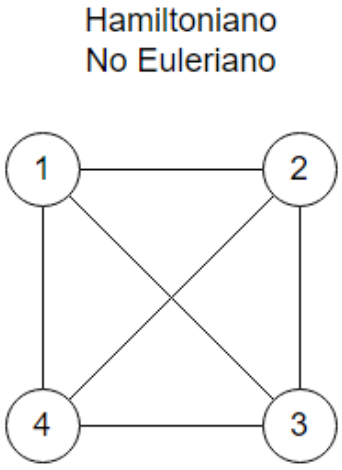



Figura 6: Circuito hamiltoneano no euleriano

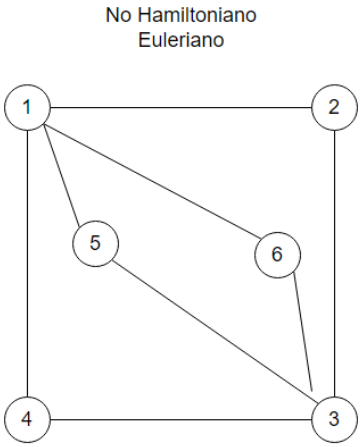



Figura 7: Circuito no hamiltoneano euleriano

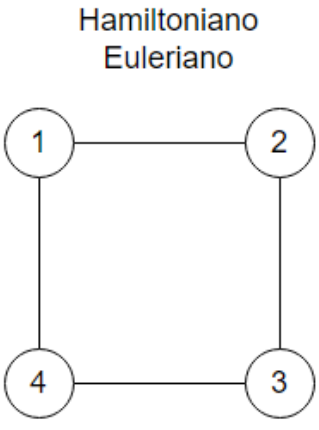



Figura 8: Circuito hamiltoneano euleriano

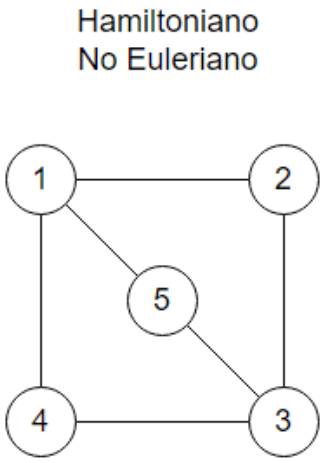



Figura 9: Circuito no hamiltoneano no euleriano