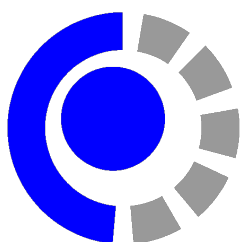


Diseño de algoritmos

Trabajo Práctico 1 - Estructuras de Datos Avanzadas y Análisis Amortizado

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

Segundo cuatrimestre 2022



Facultad de Informática
UNIVERSIDAD NACIONAL DEL COMAHUE



Índice

1. Punto 1	2
2. Punto 2	4

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 vértices. Si se trata de un grafo dirigido, la celda $M[i,j]$ indica si existe un arco entre el vértice 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.

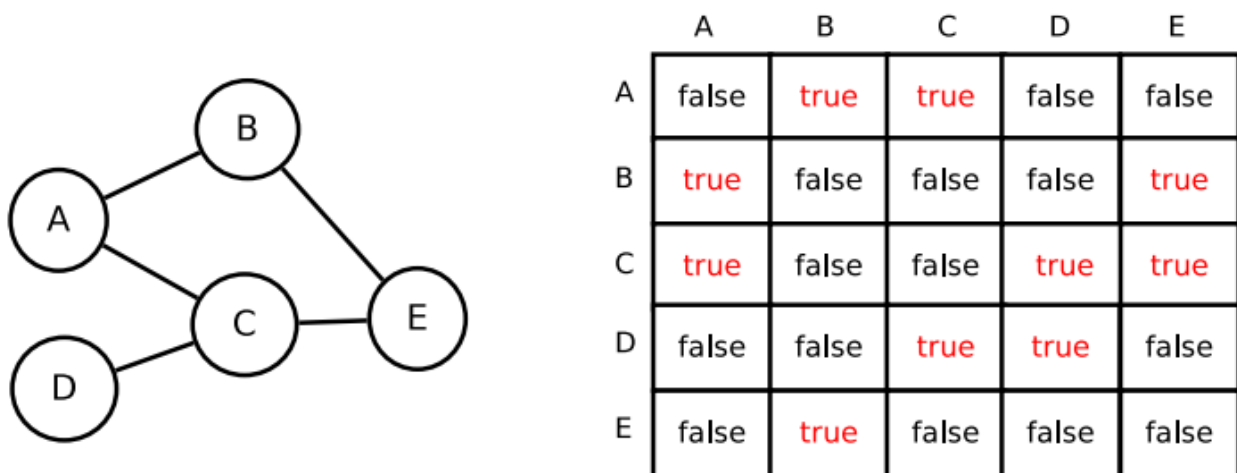


Figura 1: Representación de grafo no etiquetado con matriz de adyacencia

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 vértices son del tipo de los índices del arreglo, pero resulta poco practica cuando se necesita almacenar vértices de otro tipo. A medida que las celdas de la matriz tengan mas valores false (es decir que tiene pocos arcos entre los vértices), la matriz ocupara mas espacio del necesario. Las matrices que tienen 30 % o menos de sus celdas en valores distintos de vacío, 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 vértices, 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 ≥ 0)

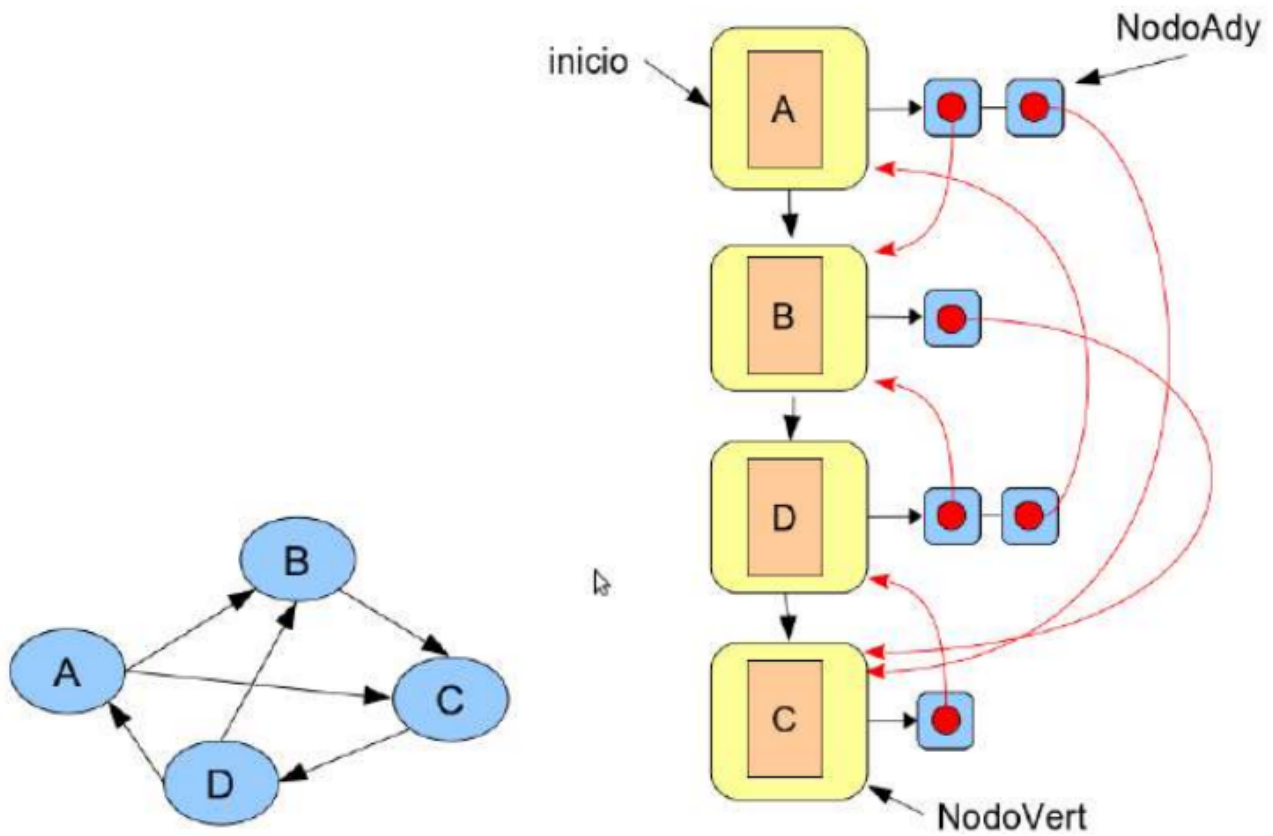


Figura 2: Representación de grafo no etiquetado con listas de adyacencia

- 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 4×4 .

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 vacío 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.

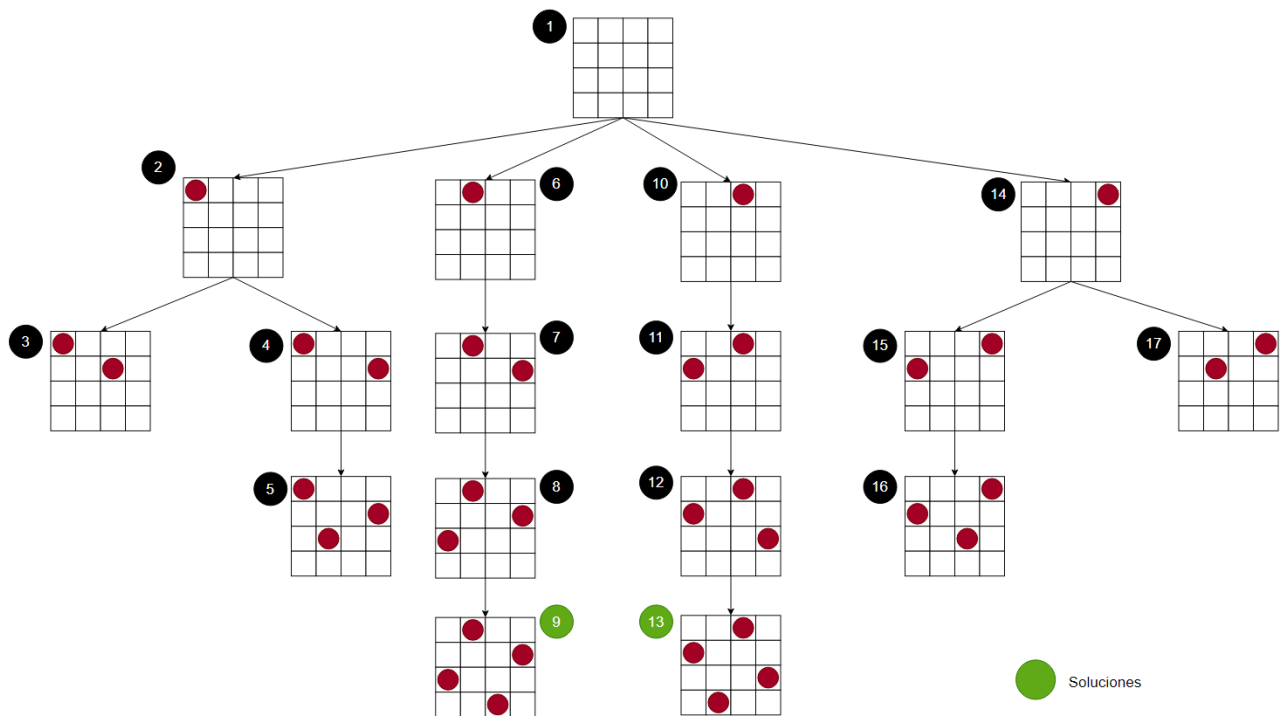


Figura 3: Solución al problema de las 4 reinas para un tablero de 4x4