Proyecto 2

Análisis de Algoritmos

Primer Semestre 2021, Prof. Cecilia Hernández

Fecha Inicio: Miércoles 9 de Junio 2021.

Fecha Entrega: Miércoles 30 de Junio 2021 (23:59 hrs).

1. Descripción
   Este proyecto consiste en analizar y aplicar algoritmos aleatorizados, amortizados y diseño greedy. Use C/C++ para implementar sus algoritmos.

2. A continuación se detallan los requerimientos del proyecto.

   a) Implementar un generador de grafos de grilla de $n \times m$, donde n y m son parámetros de entrada. Este grafo se define como $G(V, E)$, donde V es el conjunto de vértices con ids en el rango $[0..n \times m]$, y existe una arista (u, v), entre los vértices vecinos tal como se muestra en la Figura 1. Puede desplegar su grafo de grilla usando texto o bien usar python con paquete networkx. (1 punto)

   b) Diseñe e implemente una función *Random(a,b)* que retorna un valor aleatorio en el rango de valores enteros positivos $[a, b]$. Su algoritmo debe considerar que para lograr aleatoriedad debe usar la función *Random(0,1)* la cual retorna 0 o 1. Determine además cual es el tiempo esperado de este algoritmo en función de a y b. Pista: Considere una representación binaria para los valores a y b. (1 punto)

   c) Agregar pesos a las aristas a su generador de grafo de grilla, de manera que el valor aleatorio sea según una función aleatoria definida en *Random(a,b)*, donde $[a, b]$ es un rango, a y b son ingresados como parámetros. (0.5 puntos)

   d) Implemente la estructura *Union-Find* y sus operaciones *MakeSet(x)*, *Find(x)* y *Union(x,y)* usando las heurísticas de uso de ranks y compresión de ruta, y úsela para implementar el algoritmo greedy de kruskal para encontrar el árbol de cobertura mínima del grafo de grilla.(1.5 puntos)

   e) Utilice como base el grafo de grillas sin pesos y la estructura *Union-Find* para construir un laberinto. Un ejemplo para un laberinto de 6×6 aparece en la Figura 2, donde el origen del laberinto

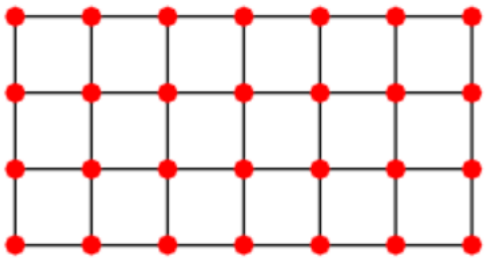

Figura 1: Ejemplo de grafo de grilla.

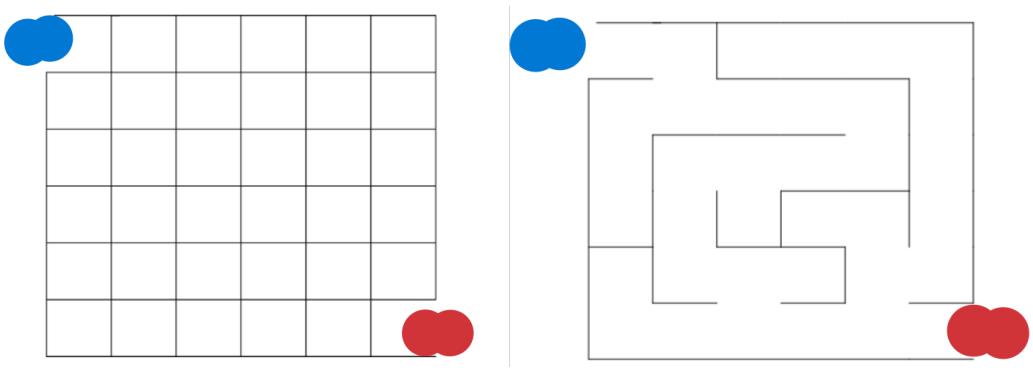

Figura 2: Ejemplo de laberinto. A la izquierda grilla inicial, a la derecha un buen laberinto.

está en la celda superior izquierda (ficha azul) y el fin está en la celda inferior derecha (ficha roja). La construcción del laberinto debe considerar que las paredes externas deben conservarse, con la excepción de una pared en el origen y una en el destino. El laberinto no puede contener un ciclo, es decir, no se deben formar islas en el interior del laberinto. Además el laberinto tiene que permitir que cada celda del laberinto esté alcanzable por cualquier otra celda. Se pide diseñar e implementar la construcción de un laberinto usando *Union-Find*. Puede visualizar el laberinto construido usando despliegue en texto, definiendo caracteres bien definidos para representar una pared y para representar una celda.(2.0 puntos)