Tecnológico de Monterrey

E1. Actividad integradora 2

Análisis y diseño de algoritmos avanzados

Victor Manuel de la Cueva

Campus Santa Fe

30 de noviembre de 2023

Fernanda Cantú Ortega - A01782232

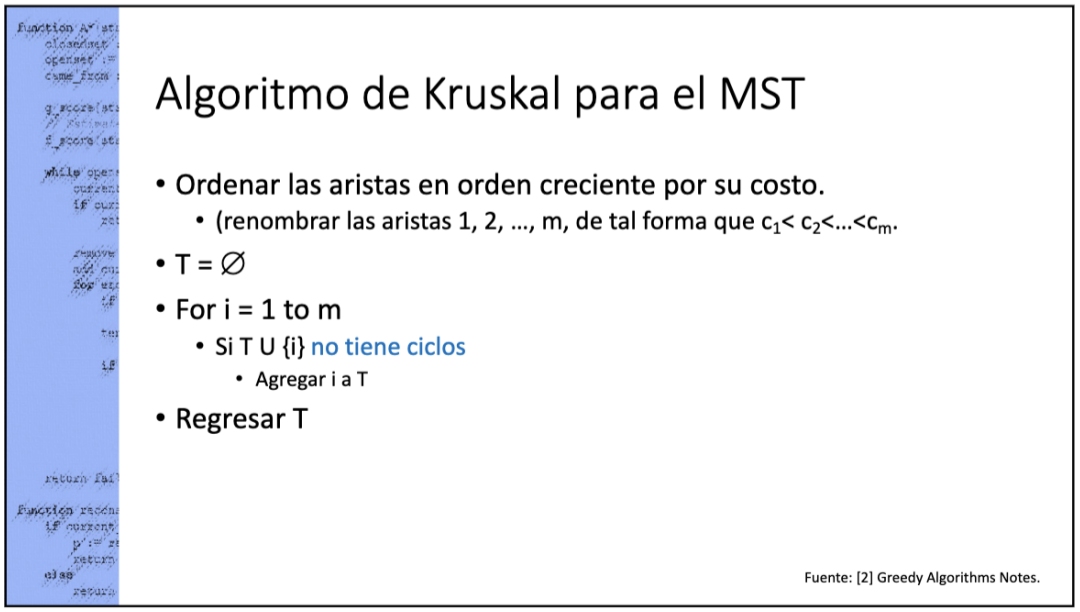
Alina Rosas Macedo - A01252720

Para esta situación problema, se nos pidió realizar un programa que solucione varios problemas con los que se ha estado enfrentando una compañía proveedora de servicios de internet. La problemática consiste en 3 partes que se mencionan a continuación.

Parte 1 (forma de cableado óptimo)

complejidad: O(n log m)

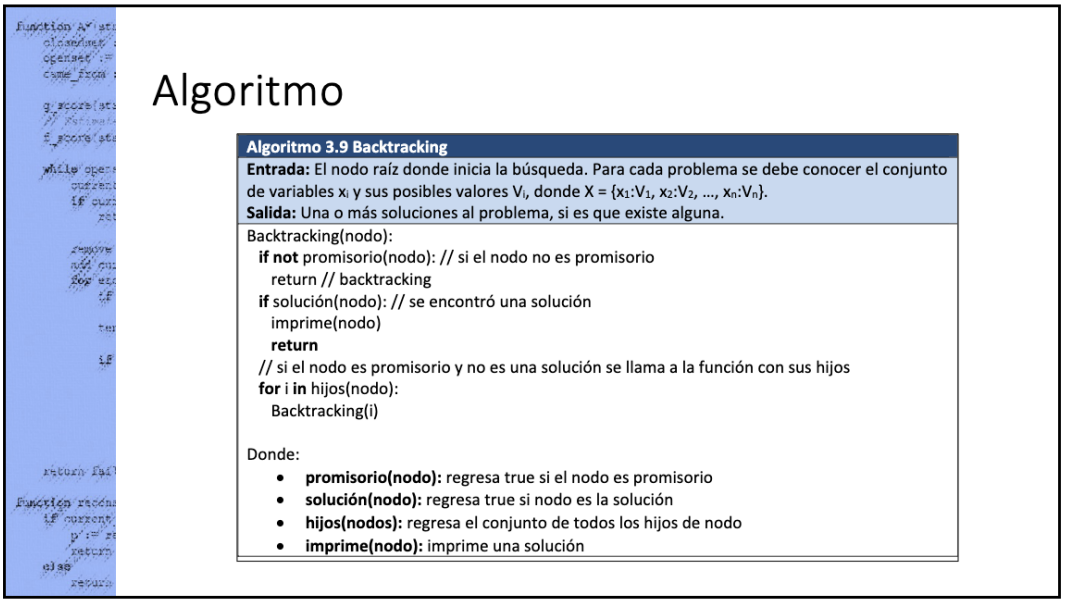
Para esta parte del código, quisimos retarnos a nosotras mismas e implementar el algoritmo de Kruskal. Nos basamos un poco en la teoría de las presentaciones pero en realidad tuvimos que ir más allá y hacer investigación por nuestra cuenta para lograr este algoritmo. El algoritmo de Kruskal funciona seleccionando las aristas en orden ascendente de peso y agregándolas al MST (Minimum Spanning Tree) si no forman un ciclo con las aristas ya seleccionadas. Para realizar esta verificación, se utiliza una estructura de conjuntos disjuntos (Union-Find) que mantiene información sobre la conexión entre los vértices.



Parte 2 (ruta para conectar colonias)

complejidad: O(n!)

En la parte 2, tratamos de implementar diferentes algoritmos para encontrar la mejor ruta para conectar las colonias. Finalmente, decidimos utilizar backtracking. Primero, esta función recibe un grafo representado como una matriz de adyacencia para llevar un registro de los nodos visitados así como dos estructuras para almacenar la ruta actual y la ruta más corta. Explora todas las posibles rutas iniciando desde cada nodo del grafo y ya que visito todos los nodos, verifica si puede regresar al nodo inicial. En caso de ser posible, actualiza la ruta como la más corta encontrada hasta ese momento. Utilizamos el siguiente pseudocódigo, además de las habilidades puestas en práctica en la actividad 1.3.



Algoritmo

Algoritmo 3.9 Backtracking

Entrada: El nodo raíz donde inicia la búsqueda. Para cada problema se debe conocer el conjunto de variables x_i y sus posibles valores V_i, donde $X = \{x_1:V_1, x_2:V_2, \ldots, x_n:V_n\}$.
Salida: Una o más soluciones al problema, si es que existe alguna.

```
Backtracking(nodo):
  if not promisorio(nodo): // si el nodo no es promisorio
    return // backtracking
  if solución(nodo): // se encontró una solución
    imprime(nodo)
    return
  // si el nodo es promisorio y no es una solución se llama a la función con sus hijos
  for i in hijos(nodo):
    Backtracking(i)
```

Donde:

- **promisorio(nodo):** regresa true si el nodo es promisorio
- **solución(nodo):** regresa true si nodo es la solución
- **hijos(nodos):** regresa el conjunto de todos los hijos de nodo
- **imprime(nodo):** imprime una solución

Parte 3 (distancia al servidor más cercano)

complejidad: O(n)

Finalmente para poder encontrar correctamente la distancia al servidor más cercano, utilizamos fuerza bruta con distancia euclidiana. Esta función, itera sobre las coordenadas centrales y calcula la distancia entre la nueva contratación (la cual se le pide al usuario que ingrese). Posteriormente, encuentra la distancia mínima y en caso de encontrar alguna, regresa su identificador y coordenadas.