Evidencia 2

TC2038: Análisis y diseño de algoritmos avanzados
Grupo 601

Alejandro Arouesty
Tec de Monterrey,
Campus Santa Fe
a01782691@tec.mx

Andrés Tarazona
Tec de Monterrey,
Campus Santa Fe
a01023332@tec.mx

Joaquín Badillo
Tec de Monterrey,
Campus Santa Fe
a01026364@tec.mx

Bajo la instrucción de
Víctor de la Cueva

30 de Noviembre de 2023

1 Reflexión

1.1 Árbol de Expansión Mínima (MST)

```cpp
void mst(const utils::AdjMatrix& adj_matrix) {
  MinHeap pq;
  std::unordered_set<int> permanent;
  std::vector<Edge> mst;

  pq.push({
    .id = 0,
    .tag = 0,
    .weight = 0
  });

  while (!pq.empty() && mst.size() < adj_matrix.size() - 1) {
    Edge current = pq.top();
    pq.pop();

    if (permanent.find(current.id) != permanent.end())
      continue;

    if (current.id != current.tag)
      mst.push_back(current);
      permanent.insert(current.id);

    for (int i = 0; i < adj_matrix.size(); i++) {
      if (i == current.id || permanent.find(i) != permanent.end())
        continue;

      pq.push({
        .id = i,
        .tag = current.id,
        .weight = adj_matrix[current.id][i]
      });
    }
  }

  for (auto edge : mst)
    std::cout << "("  << edge.tag << ", "  << edge.id  << ")\t";

  std::cout << std::endl;
}
```

Implementación 1: Algoritmo de Prim

El algoritmo de Prim, el cual fue implementado como se muestra en la Implementación 1 nos permite encontrar el árbol de expansión mínima de un grafo no dirigido y ponderado. Es decir, el conjunto de aristas que conectan todos los vértices del grafo con el menor costo posible. Este algoritmo es muy similar al algoritmo de Dijkstra, con una variación en la función de costos, mientras que Dijkstra acumula el costo (puesto que busca un camino), el algoritmo de Prim solo utiliza el costo de la arista actual.

El algoritmo de Prim es un algoritmo voraz, es decir, en cada iteración toma la mejor decisión local y si los costos son positivos, el algoritmo siempre encontrará el árbol de expansión mínima.

Como se estaba buscando una forma de conectar múltiples centrales, de tal manera que existiera un camino entre cualesquiera dos centrales y minimizar el uso de fibra óptica, el algoritmo de Prim resulta ser eficiente y correcto, dado que las distancias son estrictamente positivas.

1.1.1 Complejidad Computacional

La complejidad computacional del algoritmo de Prim es de $\mathcal{O}(E \log V)$, donde E es el número de aristas y V es el número de vértices. Esto se debe a que el algoritmo utiliza un min-heap para obtener la arista de menor costo en cada iteración.

Como la inserción de un elemento a un min-heap de tamaño N tiene una complejidad en timepo $\mathcal{O}(\log N)$, al igual que para extraer el mínimo. En el peor escenario se puede considerar un min-heap que contiene todas las aristas del grafo, que como sugiere la matriz de adyacencia son $\mathcal{O}(V^2)$ aristas. Por lo tanto, la complejidad computacional del algoritmo de Prim es

$$\mathcal{O}(E \log V^2) = \mathcal{O}(2E \log V) = \mathcal{O}(E \log V) \quad \blacksquare$$

.