Laboratorio Nro. 3: Vuelta atrás (Backtracking)

Camilo Ernesto Cruz Villegas
Universidad Eafit
Medellín, Colombia
ccruzvi@eafit.edu.co

Cristian Andrés Villamizar Ochoa
Universidad Eafit
Medellín, Colombia
Cvillam3@eafit.edu.co

1)

8. El algoritmo del numeral 1.6 recorre todos los nodos de un grafo de manera ordenada pero no uniforme, expande todos y cada uno de los nodos que va localizando, de forma recurrente, en un camino concreto.
En el algoritmo se tiene un arreglo en el que se irá marcando los nodos por los que ya pasé.
Y también se tiene un arrayList con los sucesores, otros ArrayList con la salida recursiva, y un ArrayList adicional el cual contendrá el recorrido resultante al aplicar DFS.

3) Preguntas para resolver en el informe PDF

1. Para resolver el problema del camino más corto en un grafo, aparte de fuerza bruta y backtracking, se pueden usar algoritmos voraces como Dijkstra; u otros algoritmos como A* search algorithm, Viterbi algorithm, Johnson-s algorithm, Bellman-Ford algorithm, Floyd-Warshall algorithm.
2.

Valor de N	Fuerza bruta	Vuelta atrás
4	0.001 segundo	0.001 segundo
8	0.529 segundo	0.023 segundo
16	Se demora más de 5 minutos.	3 minutos
32	Se demora más de 5 minutos.	Se demora más de 5 minutos.
N	O(n!)	O(n^n)

3. Es más conveniente usar DFS cuando el grafo tiene una profundidad corta, pero una gran amplitud.

4. **Algoritmo Dijkstra:** Encuentra el camino más corto entre dos nodos. Una variante más común corrige un único nodo como nodo "fuente" y encuentra caminos más cortos desde la fuete a todos los demás nodos en el grafo, la producción de un árbol de ruta más corta.

   Algoritmo Bellman-Ford: Es un algoritmo que calcula los caminos más cortos de una sola fuente de vértice de un dígrafo ponderado. Es más lento que el algoritmo de Dijkstra, pero más versátil, ya que es capaz de manejar grafos en los que algunos de los pesos de las aristas son números negativos.

   Algoritmo A* Search: Es un algoritmo que se utiliza ampliamente en la búsqueda de caminos y recorrido de grafos, el proceso de trazar un camino dirigido de manera eficiente entre varios puntos, llamados nodos. Goza de uso generalizado debido a su rendimiento y precisión.

   Algoritmo Floyd-Warshall: Es un algoritmo para encontrar el camino más corto en un grafo ponderado con pesos de las aristas positivas o negativas (pero sin ciclos negativos).

   Algoritmo Johnson: Es una manera de encontrar los caminos más cortos entre todos los pares de vértices en un grafo dirigido. Permite que algunos de los pesos de las aristas sean números negativos, pero sin ciclos negativos.

   Algoritmo Viterbi: Resuelve el problema del camino más corto estocástico con un peso probabilístico adicional en cada nodo.
5. En el algoritmo del numeral 2.1 se utiliza DFS.
   El algoritmo busca la ruta más corta entre el vértice 1 y el vértice n.

4) Simulacro de Parcial

1. *n-a, a, b, c*
2. *res, solucionar(n-b, a, b, c) +1*
3. res, solucionar(n-c, a, b, c) +1
4. graph.length
5. v, graph, path, pos
6. graph, path, v
7.
   1. 0 -> 3 -> 7 -> 4 -> 2 -> 1 -> 5 -> 6
   2. 1 -> 0 -> 3 -> 7 -> 4 -> 2 -> 6 -> 5
   3. 2 -> 1 -> 0 -> 3 -> 7 -> 5 -> 4 -> 6

4. 3 -> 7
5. 4 -> 2 -> 1 -> 0 -> 3 -> 7 -> 5 -> 6
6. 5 ->
7. 6 -> 2 -> 1 -> 0 -> 3 -> 7 -> 5 -> 4
8. 7 ->

8.

1. 0 -> 3 -> 4 -> 2 -> 7 -> 1 -> 6 -> 5
2. 1 -> 0 -> 5 -> 2 -> 3 -> 4 -> 6 -> 7
3. 2 -> 1 -> 4 -> 6 -> 0 -> 5 -> 3 -> 7
4. 3 -> 7
5. 4 -> 2 -> 1 -> 6 -> 0 -> 5 -> 3 -> 7
6. 5 ->
7. 6 -> 2 -> 1 -> 4 -> 0 -> 5 -> 3 -> 7
8. 7 ->

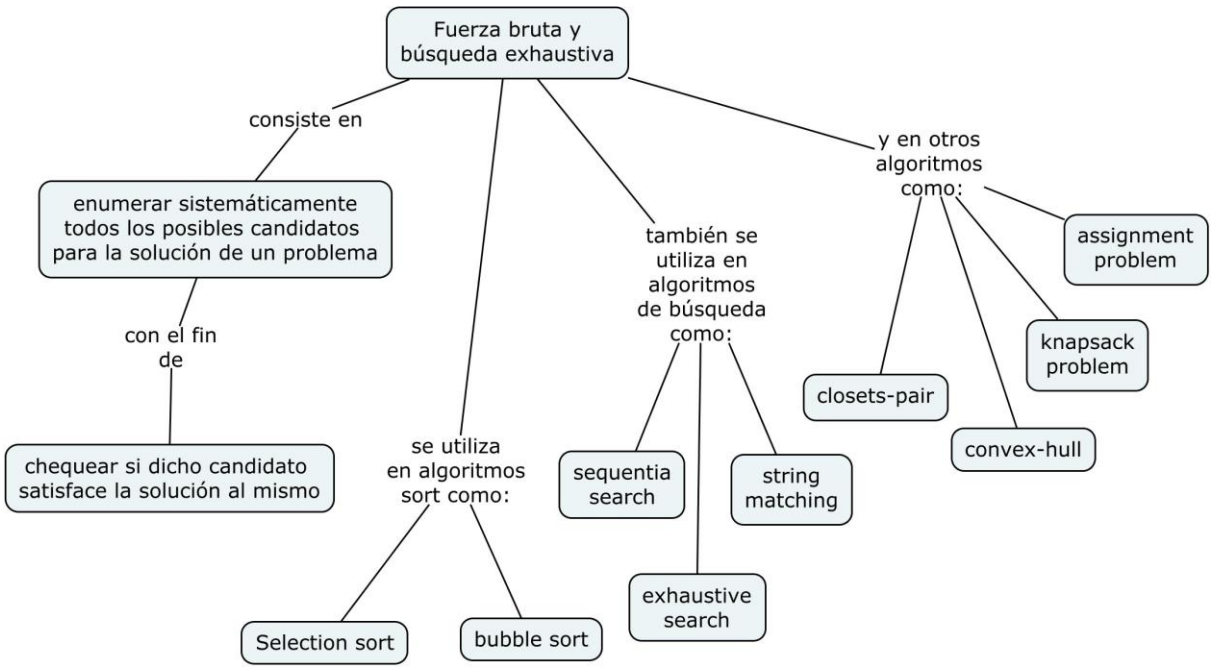
9.
```
public static LinkedList<Integer> unCamino(Graph g, int p, int q) {
      LinkedList<Integer> path = new LinkedList<>();
      If(unCaminoAux(g, p, q, list)) {
         return path;
      } else {
          return new LinkedList<>();
      }
}

public static LinkedList<Integer> unCaminoAux(Graph g, int p, int q,
LinkedList<Integer> list) {
      ArrayList<Integer> s = g.getSuccessors(p);
      If(p == q) {
         return true;
      } else {
          for(Integer i : s) {
              if(unCaminoAux(g, I, q, list) {
                 list.add(i);
                 return true;
              }
          }
      return false;
      }
}
```

5. Lectura recomendada (opcional)

a) Fuerza bruta y búsqueda exhaustiva.
b) Fuerza bruta es una técnica que consiste en enumerar sistemáticamente todos los posibles candidatos para la solución de un problema, con el fin de chequear si dicho candidato satisface la solución al mismo.
"¡Solo hazlo!" sería otra manera de describir la prescripción del enfoque de la fuerza bruta. Y a menudo, la estrategia de la fuerza bruta es de hecho la que es más fácil de aplicar.
“Selection sort” y “bubble sort” son algunos de los algoritmos donde se usa fuerza bruta.
Fuerza bruta también se puede aplicar en algoritmos de búsqueda como “Sequential search”, “Exhaustive search” y “String matching”.
También se puede utilizar para resolver problemas como “Closest-Pair”, “Convex-Hull”, “Knapsack problem” y “Assignment problems”.
En todas sus aplicaciones, los algoritmos de fuerza bruta son un buen comienzo, esto es porque la mayoría del tiempo es la solución más fácil, se pueden mejorar bastante fáciles y, a veces, los algoritmos más eficientes no valen la pena.

c) Mapa de conceptos



Bibliografía.

- Wikipedia. 2015. Problema del camino más corto. Tomado de: https://es.wikipedia.org/wiki/Problema_del_camino_m%C3%A1s_corto

DOCENTE MAURICIO TORO BERMÚDEZ
Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627
Correo: mtorobe@eafit.edu.co