Laboratorio Nro. 3: Backtracking

Rafael Villegas
Universidad Eafit
Medellín, Colombia
rvillegasm@eafit.edu.co

Felipe Cortes
Universidad Eafit
Medellín, Colombia
felipecortes@eafit.edu.co

1) Explicación ejercicio 1.6

Este algortimo intenta encontrar el camino mas corto usando DFS, pero nunca es seguro que este camino sea en efecto el mas corto. Consiste en evaluar el vertice vecino mas a la derecha del actual, y cuando encuentra el vertice objetivo empieza a meter los vertives del camino a una pila. Luego, cada elemento de la pila se inserta en un arreglo, que será el camino a retornar.

3) Simulacro de preguntas de sustentación de Proyectos

1. Ademas de las tecnicas de fuerza bruta y backtracking, existen los algoritmos voraces como dijkstra y A*, los cuales tienen propiedades diferentes a los dos mencionados y ventajas las cuales destacan su estructura.
2. **Tiempos nReinas:**

N	Tiempo (ms)
4	63
5	1
6	2
7	2
8	3
9	3
10	3
11	4
12	3
13	3
14	6
15	4
16	6

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

17	4
18	23
19	3
20	63
21	6
22	587
23	13
24	158
25	28
26	176
27	191
28	1343
29	715
30	27955
31	6768
32	47770

3. Un buen ejemplo es cuando los aristas tienen peso, es decir el arbol necesariamente contiene una ruta con un peso menor que otra, en este caso es importante hacer DFS ya que hay que llegar al final y apartir de este ignorar los otros casos que se pasarian del peso ya buscado o que no llegan al vertice buscado, mientras que los arboles BFS son buenos para buscar el camino mas corto entre dos vertices (numero de vertices) ya que es importatne saber el nivel al cual se encuentran de diferencia los vertices, por lo que recorrerlos por niveles (anchura) lo hace mas efectivo y facil.
4. Existe el algoritmo de dijkstra y A*, los cuales son algoritmos famosos en la busqueda dentro de los grafos.
5. El programa encuentra un camino desde el vertice inicial hasta el final usando dfs, ya que no encontramos una forma efectiva de hallar el camino de menor costo usando backtracking, sin ningun algoritmo voraz. Al encontrar el camino, retorna una pareja que consiste en los vertices que reorrió y el peso total del camino.
6. Su complejidad es de O(n*n!).
7. n es el numero de vertices que tiene el grafo.

4) Simulacro de Parcial

1. *n-a, a, b,c*
   res, solucionar(n, b, c, a)
   res, solucionar(n, c, a, b)
2. *graph.lenght*
   v, graph, path, pos

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

graph, path, pos+1

3. *a)*

0 → 0, 3, 7, 4, 2, 1, 5, 6
1 → 1, 0, 3, 7, 4, 2, 5
2 → 2, 1, 0, 3, 7, 4, 5, 6
3 → 3, 7
4 → 4, 2, 1, 0, 3, 7, 5, 6
5 → 5
6 → 6, 2, 1, 0, 3, 7, 4, 5
7 → 7
b)
0 → 0, 3, 4, 7, 2, 1, 6, 5
1 → 1, 0, 2, 5, 3, 4, 6, 7
2 → 2, 1, 4, 6, 0, 5, 3, 7
3 → 3, 7
4 → 4, 2, 1, 6 , 0, 5 , 3, 7
5 → 5
6 → 6, 2, 1, 4, 0, 5 , 3 , 7
7 → 7

4. *R/*

```
public static ArrayList<Integer> hayCaminoDFS(Graph g, int a, int b) {
   boolean visitados[] = new boolean[g.size()];
   ArrayList<Integer> camino = new ArrayList<Integer>();
   Stack<Integer> caminoAux = new Stack<Integer>();
   hayCaminoDFSAux(g, a, b, visitados, caminoAux);
   camino.add(a);
   while(!caminoAux.isEmpty()) {
     camino.add(caminoAux.pop());
   }
   return camino;
}

private static boolean hayCaminoDFSAux(Graph g, int v, int w, boolean[] visitados, Stack<Integer>
caminoAux) {
      visitados[v] = true;
      ArrayList<Integer> sucesores = g.getSuccessors(v);
      for (int i = 0; i < sucesores.size(); i++) {
            if (!visitados[sucesores.get(i)] && (sucesores.get(i) == w || hayCaminoDFSAux(g, sucesores.get(i), w,
                        visitados, caminoAux))) {
                  caminoAux.push(sucesores.get(i));
                   return true;
            }
      }
      return false;
}
```

5. 1
   ni, nj

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

$T(n) = 2*T(n-1)$

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