ESTRUCTURA DE DATOS 2
Código ST0247

Laboratorio Nro. 3
Vuelta atrás (Backtracking)

Juan Sebastián Sanín V.
Universidad Eafit
Medellín, Colombia
jssaninv@eafit.edu.co

Juan Pablo Peña F.
Universidad Eafit
Medellín, Colombia
jppenaf@eafit.edu.co

3) Simulacro de preguntas de sustentación de Proyectos

3.1 Existe Dijkstra, Floyd y Prim. Todos son directamente algoritmos para encontrar el camino más corto en grafos.
3.2 Habría N número de nodos y N+1 numero de aristas.
3.3

Valor de N	Tiempo de ejecución
4	0
5	0
6	0
...	...
32	25min 24 seg
N	O(N^n)

3.4 Para implementar de una forma correcta los recorridos por DFS y BFS podemos detectar cual es mejor para dicho ejercicio, identificando como es el árbol y en qué lugar del grafo está ubicado el resultado que queremos obtener. Tambien dependiendo de la forma del árbol, si es más ancho que largo deberíamos utilizar BFS de lo contrario DFS.
3.5 La estructura que se utiliza es Backtraking, el algoritmo inicia comparando el peso entre los sucesores del primer nodo, primero elige el de menor peso y aplica el mismo proceso para el nodo que acabó de tomar, a medida que se va haciendo este proceso el algoritmo va sumando el peso entre los nodos. Cuando llega al nodo final se devuelve y hace la comparación con otro sucesor del nodo anterior, lo mismo para el anterior a este hasta que finalmente encuentra el camino de menor costo.
3.6 O(max(n, m, ai, bi, wi)) + O(v) + O(u) = O(bi+v+u)
3.7 n es el número de vértices, m es el número de arcos, ai es el número del primer nodo, bi es el número del segundo nodo, wi es el peso entre los dos nodos, v es el número de vertices que el usuario eligió, u es el tamaño del grafo.
3.8 Implementamos un algortimo de backtraking, el cual está constituido por un manejo de recursión y de recorrido de grafos, obteniendo su peso y haciendo constantes comparaciones de sus distancias en cada diferente camino que se toma. También se usó el recorrido por profundidad (DFS) ya que queremos encontrar el camino más corto a un nodo el cual está muy alejado del nodo inicial.

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

4) Simulacro de Parcial

4.1 *Adicionar al if (n<a &&n<b&&n<c)*

a. int res = solucionar *(n-a, a, b, c)*
b. res = Math.max *(res,solucion(n-b,a,b,c))*
c. res = Math.max *(res,solucion(n-c,a,b,c))*

4.2

a. pos == path.length
b. sePuede(v, path, graph, pos)
c. cicloHamilAux (graph, path, pos+1)

1.5

a. 1+…
b. return Math.max(ni , nj);

1.7

a. N==r
b. A[r]=i
c. sol (a, r+1)

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473