Algoritmos y Estructuras de Datos II - 1° cuatrimestre 2019

Práctico 3 - Parte 4: DFS y BFS.

1. En cualquier árbol finitario:
   (a) caracterizar en cuáles casos una recorrida pre-orden asocia a v un número menor o igual a w
   (b) caracterizar en cuáles casos una recorrida pos-orden asocia a v un número mayor o igual a w
   (c) utilizar estás ideas para dar un algoritmo que, dados dos vértices v y w en un árbol determina si v es o no ancestro de w
   (d) analizar en qué casos dicho algoritmo podría resultar conveniente

2. Considerar el grafo $G = (V, A)$ con vértices $V = \{1, 2, 3, 4, 5, 6, 7, 8\}$ y aristas $A = \{(1, 2), (1, 3), (1, 5), (3, 4), (5, 3), (5, 4), (6, 1), (7, 6), (7, 8), (6, 7)\}$. Describir como se efectúan las llamadas recursivas al procedimiento *dfs* para el grafo G, y dar el orden en que se visitan sus nodos. Mostrar cómo va cambiando la pila de la versión iterativa.

3. Mostrar cómo evoluciona la cola y el orden en que se visitan los nodos al aplicar el procedimiento *bfs* al grafo del ejercicio 2.

4. Repetir los dos ejercicios anteriores para el grafo no dirigido que se obtiene reemplazando (i, j) por $\{i, j\}$ en el grafo G del ejercicio 2.

ejercicios adicionales

5. Para el problema de la moneda hemos visto la siguiente definición que utiliza backtracking:

$$cambio(i, j) = \begin{cases} 0 & j = 0 \\ 1 + \min\{cambio(i', j - d_{i'}) \mid i' \in \{1, 2, \ldots, i\} \wedge d_{i'} \leq j\} & j > 0 \end{cases}$$

   Reescribir este algoritmo utilizando DFS iterativo. Para ello, considere el árbol cuyos vértices son ternas (i, j, r), la raíz es $(n, k, 0)$ y hay una arista de (i, j, r) a $(i', j'.r')$ si y sólo si se cumple la siguiente condición:

   - $1 \leq i' \leq i \wedge d_{i'} \leq j \wedge j' = j - d_i \wedge r' = r + 1$

   (a) Escribir primero la función children, que aplicado a un vértice devuelve el conjunto o lista de sus hijos
   (b) Caracterizar las ternas (i, j, r) que corresponden a hojas del árbol.
   (c) Escribir la función visitar, que debe recoger el tercer elemento de cada hoja.

6. Reescribir el algoritmo anterior utilizando BFS. ¿Qué optimización puede hacer?

7. A los fines de este ejercicio, un grafo viene dado por su conjunto de vértices V y la función **neighbours** (que aplicada a cada vértice v devuelve la lista o conjunto de los vértices w tales que hay una arista de v a w). Escribir un algoritmo que reciba un grafo cualquiera y devuelva una tabla con las marcas correspondientes a la visita en DFS pero en pos-orden. Es importante tener en cuenta los ciclos. Una traducción ingenua del algoritmo en pre-orden del teórico fallaría en presencia de ciclos.