# Data Structures & Algorithms - Problem Sheet 3

Dom Hutchinson

October 26, 2018

**Question 1**
How many simple directed (unweighted) graphs on the set of vertices $\{v_1, v_2, \dots, v_n\}$ are there that have at most one edge between any pair of vertices? (That is, for two vertices, $a$ &$b$, only at most one of the edges $(a, b)$ and $(b, a)$ is in the graph.) For this question vertices are distinct and isomorphic graphs are not the same (Graphs that have the same number of vertices and identical connections are said to be isomorphic). Justify your answer.

**My Solution 1**
There are $\frac{1}{2}n(n-1)$ unique pairs.
Each pair $(a, b)$ can be in one of three states: no edges; edge $a \to b$; or, edge $b \to a$.
Since each pair is independent this gives $\frac{3}{2}n(n-1)$ possible combinations.

**Question 2**
Given the visited node order for each of DFS and BFS starting with $s$, given the following adjancecy list

$s \to a, c, d.$

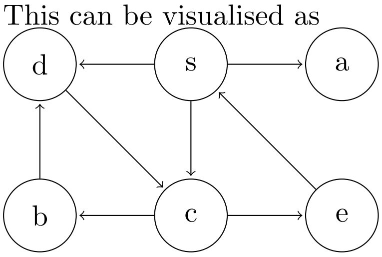
$c \to e, b.$

$b \to d.$

$d \to c.$

$e \to s.$

This can be visualised as




**My Solution 2**
**DFS**
Let the leftmost element represent the top of the stack.

| # | Visit | Stack |
|---|---|---|
| i) | s | (acd) |
| ii) | a | (cd) |
| iii) | c | (ebd) |
| iv) | e | (sbd) |
| | s | (bd) |
| v) | b | (dd) |
| vi) | d | (d) |
| | d | () |

In a depth first search the nodes are visited $s \rightarrow a \rightarrow c \rightarrow e \rightarrow b \rightarrow d$.

**BFS**

Consider elements to be push into the left of the queue, and popped off the right.

| # | Visit | Stack |
|---|---|---|
| i) | s | (dca) |
| ii) | a | (dc) |
| iii) | c | (bed) |
| iv) | d | (cbe) |
| v) | e | (scb) |
| vi) | b | (dsc) |
| | c | (ds) |
| | s | (d) |
| | d | () |

In a breadth first search the nodes are visited $s \rightarrow a \rightarrow c \rightarrow b \rightarrow e \rightarrow d$.

**Question 3.1**

Is it true that both DFS and BFS require $\omega(|V|)$ storage for their operation?

**My Solution 3.1**

Yes, since you always have to store where you have been, which is $|V|$ nodes and in the best case you only ever have one node in the stack/queue.

**Question 3.2**

How much storage do DFS and BFS require to visit all nodes of a tree starting at the root?

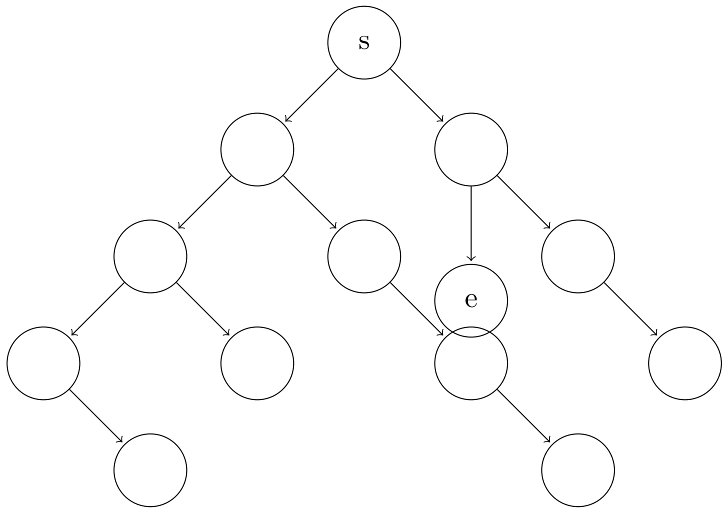
**My Solution 3.2**

**DFS**$=max(height)$.
**BFS**$=max(width)$.

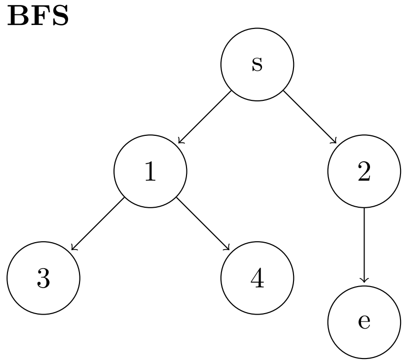
**Question 4**

In the Following graphs, assume that if there is ever a choice amongst multiple nodes, both the BFS & DFS algorithms will choose the left-most node first. Starting from $S$ at the top, which algorithm will visit the least number of nodes before visiting $E$?

**Question 4.1**

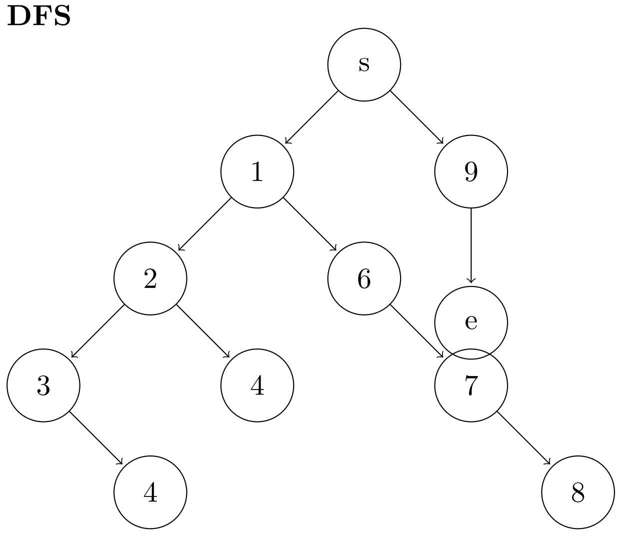



**My Solution 4.1**
**BFS**



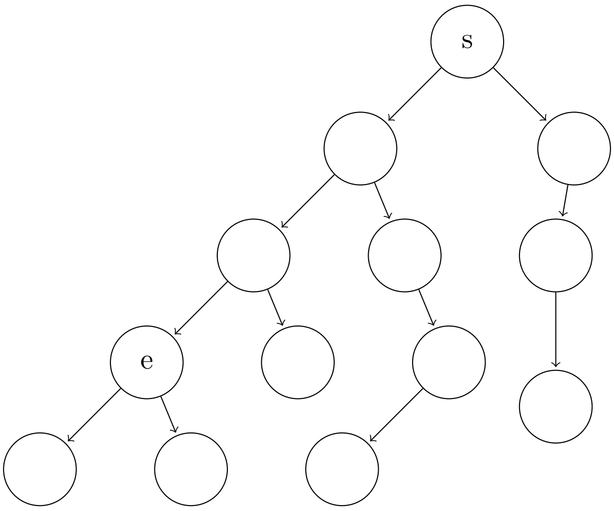

In a breadth first search the nodes are met in the order shown above. So $e$ is met after 5 nodes have been checked.

**DFS**

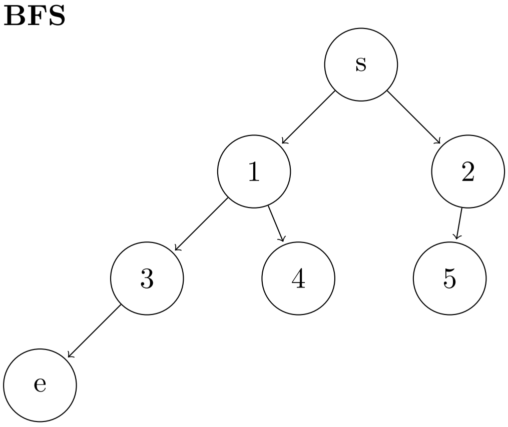



In a depth first search the nodes are met in the order shown above. So $e$ is met after 10 nodes have been checked.
Thus <u>BFS</u> meets $e$ quicker.

**Question 4.2**

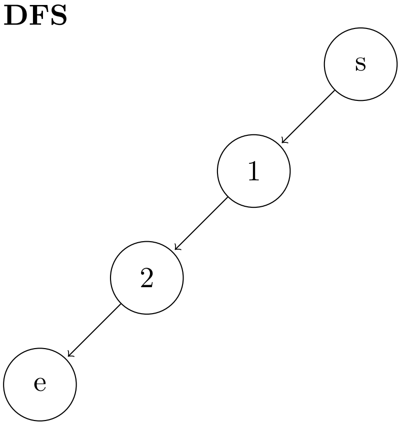


**My Solution 4.2**
**BFS**



In a breadth first search the nodes are met in the order shown above. So $e$ is met after 5 nodes have been checked.

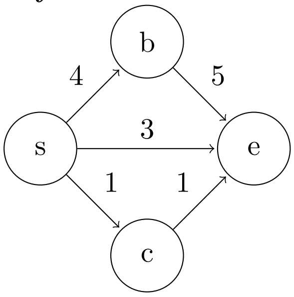
**DFS**



In a depth first search the nodes are met in the order shown above. So $e$ is met after 3 nodes have been checked.
Thus DFS meets $e$ quicker.

**Question 5**
Given an example of a weighted graph $G$ for which BFS gives the incorrect shortest path between two nodes.

**My Solution 5**




In this example a breadth first search would say the cheapest path from $s \to e$ costs 3, when in fact going $s \to c \to e$ only costs 2.

**Question 6**
What is the Discrete Fourier Transform of $1 + 2x + 3x^2 + 4x^3$?

i) $5/2, \frac{-1+i}{2}, -1/2, \frac{-1-i}{2}$

ii) $10, -2-2i, -2, -2+2i$

iii) $10, \frac{3-\sqrt{2}}{2} + \frac{2+\sqrt{2}}{2}i, 3 + \frac{\sqrt{2}}{2} + \frac{-2+\sqrt{2}}{2}i, -2, \frac{3+\sqrt{2}}{2} - \frac{-2+\sqrt{2}}{2}i, -2+2i, \frac{3+\sqrt{2}}{2} - \frac{2+\sqrt{2}}{2}i$

iv) $1, 2, 3, 4$

v) Something else

**My Solution 6**

$$\begin{aligned}
A(x) &= 1 + 2x + 3x^2 + 4x^3 \\
4^{th} \text{ Roots of Unity} &= 1, i, -1, -i \\
\\
A(1) &= 1 + 2 + 3 + 4 \\
&= 10 \\
A(i) &= 1 + 2i - 3 - 4i \\
&= -2 - 2i \\
A(-1) &= 1 - 2 + 3 - 4 \\
&= 2 \\
A(-i) &= 1 - 2i - 3 + 4i \\
&= 2i - 2 \\
\\
y &= (10, -2-2i, -2, 2i-2) \\
&\equiv \textbf{ii)}
\end{aligned}$$

**Question 7**
Describe the generalisation of the FFT procedure to the case in which $n$ is a power of 3. Give the recurrence for the running time and solve the recurrence.

**My Solution 7.1**
Define three polynomials

$$\begin{aligned}
A^{[0]} &= a_0 + a_3x + \cdots + a_{N-3}^{\frac{N}{3}-1} \\
A^{[1]} &= a_1 + a_4x + \cdots + a_{N-2}^{\frac{N}{3}-1} \\
A^{[2]} &= a_2 + a_5x + \cdots + a_{N-1}^{\frac{N}{3}-1}
\end{aligned}$$

Then
$$A(x) = A^{[0]}(x^3) + xA^{[1]}(x^3) + x^2A^{[2]}(x^3)$$
Evaluate $A(x)$ at $(\omega_N^0)^3, (\omega_N^1)^3, \dots, (\omega_N^{N-1})^3$.
Since $N$ is a power of 3 we can use the cancellation lemma to show

$$(\omega_N^m)^3 = \omega_N^3 m = \omega_{N/3}^m \quad \forall\ m \in \mathbb{N}_0$$

So there are a third as many complex roots of unity.

**My Solution 7.2**
This process has a recurrence equation

$$\begin{array}{rrcl}
 & T(1) & = & 1 \\
 & T(n) & = & 3T(\frac{n}{3}) + \alpha n \quad \alpha \in \mathbb{R} \\
 & & & \\
 & T(n) & = & 9T(\frac{n}{9}) + 2\alpha n \\
 & & \vdots & \\
 & & = & 3^m T(\frac{n}{3^m}) + \alpha mn \\
\text{Until} & \frac{n}{3^m} & = & 1 \\
\implies & m & = & \log_3 n \\
\implies & T(n) & = & 3^{\log_3 n} T(1) + \alpha n \log_3 n \\
 & & = & n.1 + \alpha n \log_3 n \\
 & & = & \underline{n(1 + \alpha \log_3 n)} \in O(n \log_3 n)
\end{array}$$