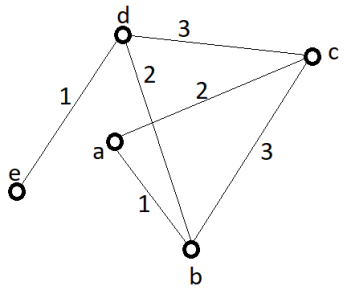
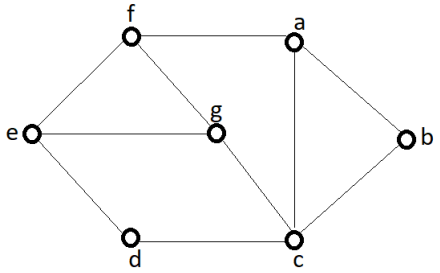
Algorithms - CS5800 - Summer 2, 2017 - Assignment 3

1. Apply Prim's algorithm on the following graph starting at the vertex a. Explain each step of the algorithm.



2. Apply the BFS algorithm on the following graph starting at the vertex a. Explain each step of the algorithm.



3. For a graph $G = (V, E)$ and $v_1, v_2 \in V$, we define $d_3(v_1, v_2)$ in the following way:

   $d_3(v_1, v_2) = \min_p\{|p| \ : \ p \text{ is a path connecting } v_1 \text{ and } v_2 \text{ with } 3k \text{ edges } (k \text{ is some integer})\}$

   ($|p|$ is the number of edges in a path p.)
   Set $v_0 \in V$.
   Describe an algorithm that computes $d_3(v_0, v)$ for any $v \in V$.

4. Let $G = (V, E)$ be a directed graph. If we apply the DFS algorithm on the graph G then we can divide the edges of G into four types in terms of the DFS forest G_π.

   - Tree edges are edges in the DFS forest G_π.
   - Back edges are edges (u, v) connecting a vertex u to an ancestor v in the DFS tree. (We consider self-loops, which may occur in directed graphs, to be back edges).
   - Forward edges are non-tree edges (u, v) connecting a vertex u to a descendant v in the DFS tree.

- Cross edges are all other edges. They can go between vertices in the same DFS tree, as long as one vertex is not an ancestor of the other, or they can go between vertices in different DFS trees (note that a DFS forest may have several DFS trees).

(a) Make sure you understand why the following inequalities hold (you do not need to prove them, but it is recommended that you understand why they are true):
If (u, v) is a tree edge then $d(u) < d(v) < f(v) < f(u)$.
If (u, v) is a back edge then $d(v) < d(u) < f(u) < f(v)$.
If (u, v) is a forward edge then $d(u) < d(v) < f(v) < f(u)$.
If (u, v) is a cross edge then $d(v) < f(v) < d(u) < f(u)$.

Deduce that only for a back edge we have $f(u) < f(v)$.

(b) Prove:
The graph G has a cycle in it, if and only if the DFS forest of the graph G_π has a back edge.

(c) Suggest an algorithm that determines whether a given directed graph is acyclic or not.

5. We saw the Heapify(A, n) algorithm in class, and we saw that its running time is $O(\log n)$ where n is the length of A.
We also saw how to use it to turn an array into a heap:
MakeHeap(A, n)

- For $i = n$ down to 1
- Heapify(T_i)

(a) The running time of the algorithm MakeHeap(A, n) is $O(n)$ (It appears in the slides and as an optional exercise below).
Use the algorithms MakeHeap(A, n) and Heapify(A, n) to construct a (comparison) sorting algorithm with complexity $O(n \log n)$.

(b) Use the fact that the complexity of MakeHeap(A, n) is $O(n)$ to deduce the following fact:
There exists a constant $c > 0$ such that an array A with n distinct numbers can be ordered as a heap in at least $\frac{n!}{2^{cn}}$ different ways. (Note that $\frac{n!}{2^{cn}}$ is very large. You can lookup Stirling's approximation of $n!$ to see this.)
(The point of this question is: if it is enough to do a "small" number of comparisons to order A as a heap then there must be many different heaps.)

(c) This part is optional.
In the complexity analysis of MakeHeap in the slides we use the fact that

$\sum_{k=1}^{\infty} k2^{-k} = 2$. We will now see how to prove it.
The formula for the sum of a geometric sequence is:

$$\sum_{k=0}^{n} x^k = \frac{1 - x^{n+1}}{1 - x}$$

If we differentiate both sides of the formula we get the formula:

$$\sum_{k=0}^{n} kx^{k-1} = \left(\frac{1 - x^{n+1}}{1 - x}\right)' = ...$$

and then multiplying both sides by x would give us:

$$\sum_{k=0}^{n} kx^{k} = x \cdot \left(\frac{1 - x^{n+1}}{1 - x}\right)' = ...$$

Use an explicit expression of the right side of the above formula to show that $\sum_{k=0}^{n} \left(k \left(\frac{1}{2}\right)^k\right) < 2$.
Explain the proof in the slides that shows that the complexity of the Make-Heap algorithm is $O(n)$.