Problem 1

Question: Run the $Dijkstra(G, s)$ algorithm on graph G in Figure 1 of Lecture Note 9, starting at v.

Answer:

Iteration	F	d values for $v \in F$
0	v	$d(v) = 0$
1	v, x	$d(v) = 0$, $d(x) = 2$
2	v, x, z	$d(v) = 0$, $d(x) = 2$, $d(z) = 3$
3	v, x, z, c	$d(v) = 0$, $d(x) = 2$, $d(z) = 3$, $d(c) = 6$
4	v, x, z, c, y	$d(v) = 0$, $d(x) = 2$, $d(z) = 3$, $d(c) = 6$, $d(y) = 6$
5	v, x, z, c, y, q	$d(v) = 0$, $d(x) = 2$, $d(z) = 3$, $d(c) = 6$, $d(y) = 6$, $d(q) = 7$
6	v, x, z, c, y, q, u	$d(v) = 0$, $d(x) = 2$, $d(z) = 3$, $d(c) = 6$, $d(y) = 6$, $d(q) = 7$, $d(u) = 12$
7	v, x, z, c, y, q, u, w	$d(v) = 0$, $d(x) = 2$, $d(z) = 3$, $d(c) = 6$, $d(y) = 6$, $d(q) = 7$, $d(u) = 12$, $d(w) = 13$

Iteration	F	d values for $v \notin F$
0	v	$d(x) = 2, d(c) = 6, d(z) = 3$
1	v, x	$d(c) = 6$, $d(z) = 3$
2	v, x, z	$d(c) = 6$, $d(y) = 6$
3	v, x, z, c	$d(y) = 6$, $d(u) = 12$
4	v, x, z, c, y	$d(u) = 12$, $d(q) = 7$
5	v, x, z, c, y, q	$d(u) = 12$
6	v, x, z, c, y, q, u	$d(w) = 13$
7	v, x, z, c, y, q, u, w	

Explanation for each Iteration

- At iteration 0: The source vertex v is added to F , the relaxation process provides the estimates for vertices x, c, z
- At iteration 1: Vertex x is added to F
- At iteration 2: Vertex z is added to F and Vertex y has its distance estimated from z.
- At iteration 3: Vertex c is added to F and Vertex u has its distance estimated from c.
- At iteration 4: Vertex y is added to F and Vertex q has its distance estimated from y.
- At iteration 5: Vertex q is added to F.
- At iteration 6: Vertex u is added to F and Vertex w has its distance estimated from u.
- At iteration 7: Vertex w is added to F and algorithm terminates as every other vertex is unreachable.

Problem 2

Question: Let T be a heap tree (representing an Array H) of height n

(a) What is the number of internal nodes of T with exactly one child?

Answer:

At most one node has exactly one child. Each parent can have at most, two child nodes. Consider a heap represented by an array A . When you insert a new node into the heap, the node is added to the first free array slot. This means that each if a parent only has one node, every parent before it, must have two children. So either number of internal nodes with exactly one child is either 0 or 1.

(b) Explain why every leaf of T is at distance n or $n-1$ from the root of T.

Answer:

Similarly to the answer above, when vertices are inserted, into a heap represented by an array A , they are inserted at the first free array slot. These new child nodes will not have any children, and therefore they are leafs. These leafs are at the furthest distance from the root, ie. the height of the heap, n. Similarly, the parents of these leaves can potentially have no children, so they are also leaves, but at a distance of $n-1$.

Problem 3

Question: Let G be a graph. A k-colouring of G is a colouring of vertices with at most k colours such that no two adjacent vertices have the same colour.

(a) Describe 1-colourable graphs.

Answer:

1-colourable graphs are graphs which have no edges, if the graph has no edges then you can colour each graph the same colour but as soon as you have an edge you need to apply different colours to the end vertices of that edge.

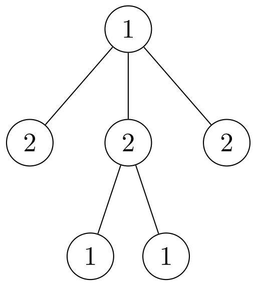
(b) What is the minimal number of colors needed to colour a tree with more than 1 node. Explain your answer.

Answer:

For a tree with more than one node, 2 colours is enough.

This is because we can fix the color of the root node, then all its children can have a different color, then all of its children can have the first color and so forth so the minimal number of colours required is 2.

For example we can consider this tree (where root is coloured 1 then second layer is 2 then 1 and so forth):



Problem 3 continued on next page. . .

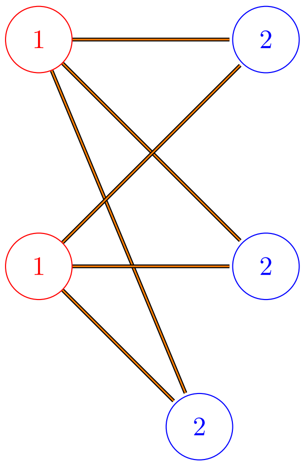
(c) Explain that the bipartite graph $K_{n,m}$ can be 2-colourable.

Answer:

For $K_{n,m}$ we have that it can be 2-colourable, why?

Because we can assign 1 to each vertex in the n independent set and 2 to each vertex in the m independent set.

For example consider the graph $K_{2,3}$ we can color all the m vertices in red and all the n vertices in blue so there are at most 2 colours.



Problem 4

Question:

Let k be the maximal among the degrees of all vertices of graph G. Write down a greedy and linear / polynomial time algorithm that colours G with at most $k + 1$ colours.

Answer:

Inputs: G the graph

Outputs: Coloured version of $G = G'$

We can do the following steps for a greedy algorithm:

- Colour in the first vertex with the first colour
- For $k = 2$ to $|V|$
  - For the current vertex V_k , colour it in with lowest numbered colour that has not been previously used on any **previously coloured vertices adjacent to it**.
  - If all previously used colours appear on vertices adjacent to V_k then assign a new colour to V_k.

This algorithm is polynomial since you need to loop through all vertices and check adjacent vertices however with the help of adjacency matrices it is possible to make this algorithm linear.

This algorithm also colours G with at most $k + 1$ colours.

Problem 5

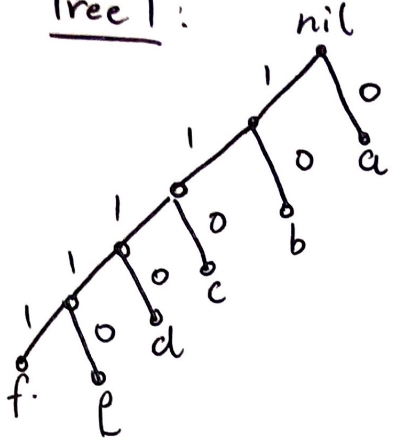
Question:

Suppose $S = \{a, b, c, d, e, f\}$. Give two examples of prefix codes for these letters. Present the prefix codes as binary trees.

Answer:

Note that $|S| = 6$ so there will be 6 prefix codes.

Tree 1 :



$\therefore$ $\gamma(a) = 0$

$\gamma(b) = 10$

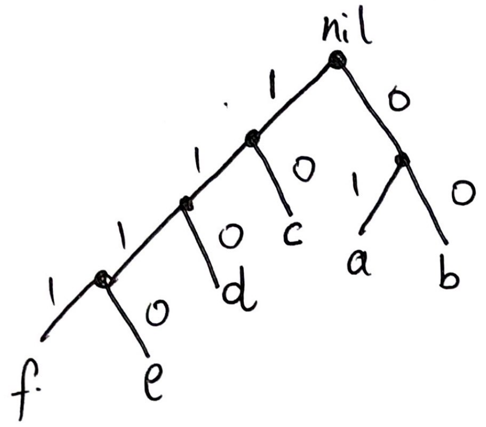
$\gamma(c) = 110$

$\gamma(d) = 1110$

$\gamma(e) = 11110$

$\gamma(f) = 11111$

Tree 2



$\therefore$ $\gamma(a) = 01$

$\gamma(b) = 00$

$\gamma(c) = 10$

$\gamma(d) = 110$

$\gamma(e) = 1110$

$\gamma(f) = 1111$