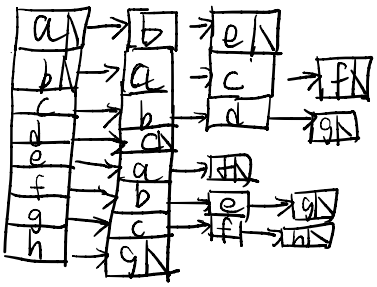
INT102 Algorithmic Foundations
Problem Session 2, Week 6
Group1: 09:00-11:00, 31/03/2023, Friday
Group2: 17:00-19:00, 31/03/2023, Friday
Location: SC176/Online

---

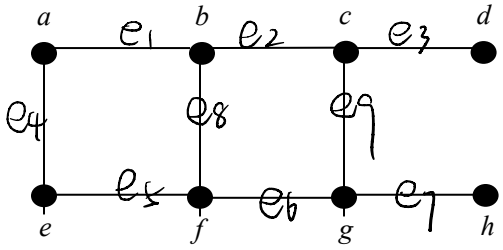
adjacency matrix:

	a	b	c	d	e	f	g	h
a	0	1	0	0	1	0	0	0
b	1	0	1	0	0	1	0	0
c	0	1	0	1	0	0	1	0
d	0	0	1	0	0	0	0	0
e	1	0	0	0	0	1	0	0
f	0	1	0	0	1	0	1	0
g	0	0	1	0	0	1	0	1
h	0	0	0	0	0	0	1	0



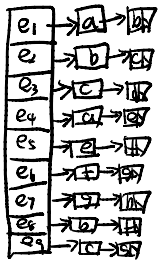
Question 1

Consider the following graph G.



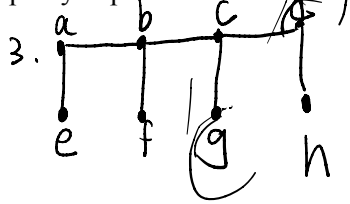
2. let $e_1=(a,b)$ $e_2=(b,c)$ $e_3=(c,d)$ $e_4=(a,e)$ $e_5=(e,f)$ $e_6=(f,g)$ $e_7=(g,h)$ $e_8=(b,f)$ $e_9=(c,g)$

	a	b	c	d	e	f	g	h
e_1	1	1	0	0	0	0	0	0
e_2	0	1	1	0	0	0	0	0
e_3	0	0	1	1	0	0	0	0
e_4	1	0	0	0	1	0	0	0
e_5	0	0	0	0	1	1	0	0
e_6	0	0	0	0	0	1	1	0
e_7	0	0	0	0	0	0	1	1
e_8	0	1	0	0	0	1	0	0
e_9	0	0	1	0	0	0	1	0

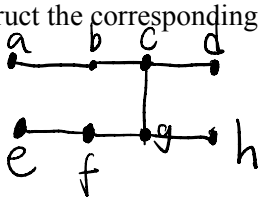


1. Give the adjacency matrix and adjacency list of the graph *G*.
2. Give the incidence matrix and incidence list of the graph *G*.
3. Starting at the vertex ***a*** and resolving ties by the vertex alphabetical order traverse the graph by breadth-first-search (BFS) and construct the corresponding BFS tree.
4. Starting at the vertex ***a*** and resolving ties by the vertex alphabetical order traverse the graph by depth-first-search (DFS) and construct the corresponding BFS tree.

3.



4.



Question 2

Consider the following recursive function

$$f(n)=\begin{cases}1 & if \quad 0 \le n \le 3 \\ f(n-1)+f(n-2)+f(n-3)+f(n-4) & if \quad n>4\end{cases}$$

1. Write a recursive (top-down) algorithm to compute it.
2. What is the complexity of your algorithm (in big-O notation)?
3. Design and write the pseudo code of a faster nonrecursive (bottom-up) algorithm using the concept of dynamic programming.
4. What is the time complexity of the faster algorithm (in big-O notation)?

1. Procedure f(n)
if n==0 or n==1 or n==2 or n==3
return 1
else
return f(n)=f(n-1)+f(n-2)+f(n-3)+f(n-4)

∴ $F(n)>4F(n-4)$
$F(n)>4F(n-4-4)=4(4F(n-4-4))$
$>4^2F(n-8-4))=4^3F(n-12)$
$>4^kF(n-4k)$
$F(n)>4^{\frac{n}{4}}=2^{\frac{n}{2}}$
$\therefore O(n)=2^n$

3. Procedure f(n)
Set A[0]=A[1]=A[2]=A[3]=1
for i=4 to n do
A[i]=A[i-1]+A[i-2]+A[i-3]+A[i-4]
return A[n]

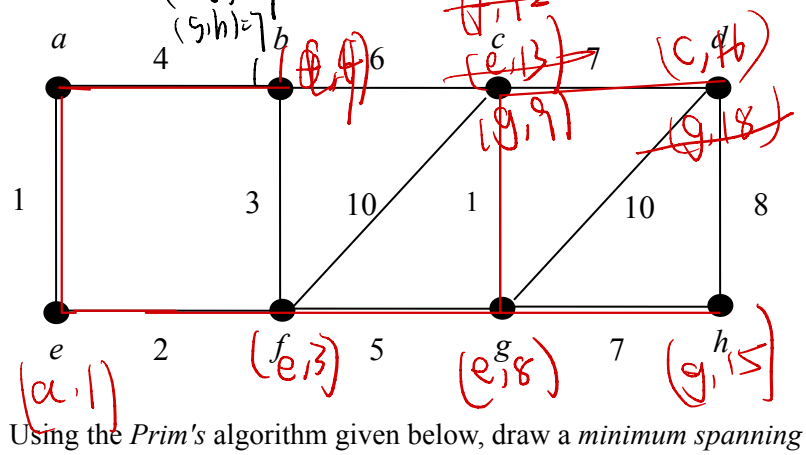
$F(n)=(F(n-2)+F(n-3)+F(n-4)+F(n-5)+1)$
$+(F(n-3)+F(n-4)+F(n-5)+F(n-6)+1)$
$+F(n-4)+F(n-5)+F(n-6)+F(n-7)+1)$
$+F(n-4)+1$
$=F(n-2)+2F(n-3)+4F(n-4)+3F(n-5)+F(n-6)+F(n-7)+4$

$F(n-3)=F(n-4)+F(n-5)+F(n-6)+F(n-7)+1$
$F(n-2)=F(n-3)+F(n-4)+F(n-5)+F(n-6)+1$

time complexity Fn
$F(n)=F(n-1)+F(n-2)+F(n-3)+F(n-4)+1$
$F(n-1)=F(n-2)+F(n-3)+F(n-4)+F(n-5)+1$

Question 3

Consider the following graph G. The label of an edge is the cost of the edge.



1. Using the *Prim's* algorithm given below, draw a *minimum spanning tree* (MST) of the graph Also write down the change of the priority queue step by step and the order in which the vertices are selected. Is the MST drawn unique? (i.e., is it the one and only MST for the graph?)

```
MST-Prim-Algorithm(G, w, root)
// Given a weighted graph G=(V,E) and a root vertex
for each u in V do
  begin
     key[u] = ∞
     π[u] = NIL
  end
key(root) = 0
V_T = ∅
while V - V_T ≠ ∅ do
   begin
     choose the vertex u in V - V_T with minimum key(u)
          V_T = V_T ∪ { u }
          for every vertex v in V - V_T that is a neighbour of u do
             if w(u,v) < d(v) then    // w(u,v) is the weight of edge (u,v)
                   d(v) = w(u,v)
                   π[v] = u
  end
```

2. Using *Kruskal's* algorithm, draw a *minimum spanning tree* (MST) of the graph G. Write down the order in which the edges are selected. Is the MST drawn unique? (i.e., is it the one and only MST for the graph?)

3. Referring to the same graph above, find the shortest paths from the vertex ***a*** to *all* other vertices in the graph G using *Dijkstra's* algorithm given below. Show the changes of the priority queue step by step and give the order in which edges are selected.

```
Dijkstra-Algoritm(G, w, root)
// Given a weighted graph G=(V,E) and a source vertex s
for each u in V  do
    begin
      d(v) = ∞
      p(v) = null
    end
d(s) = 0
V_T = ∅
while V - V_T ≠ ∅ do        // there is still some vertex left
begin
   choose the vertex u in V - V_T with minimum d(u)
  V_T  = V_T ∪ { u }
  for every vertex v in V - V_T  that is a neighbour of u  do
      if d(u) + w(u,v) < d(v)  then   //w(u,v) is the weight of edge (u,v)
         d(v) = d(u) + w(u,v)
         p(v) = u
end
```

N.B. There may be more than one solution. You only need to give one of the solutions.