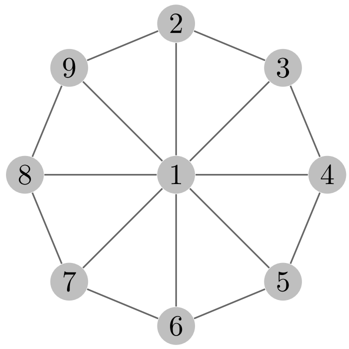
# Back-paper

***Write your roll number in the space provided on the top of each page.***
*Write your solutions clearly in the space provided after each problem. You may use additional sheets for working out your solutions; attach such sheets at the end of the question paper. You are not allowed to consult your notes, books or the internet.*
**Time: 3 hrs**
**Attempt all problems.**

Name and Roll Number: ____________________

| Problem | Points | Score |
|---|---|---|
| 1 | 10 | |
| 2 | 10 | |
| 3 | 20 | |
| 4 | 20 | |
| 5 | 20 | |
| 6 | 20 | |
| Total: | 100 | |

1. Consider the following undirected wheel graph.




Assume that the above wheel graph is given as adjacency lists, where in the list of vertex $v$, the neighbours of $v$ are listed in ascending order.

(a) Suppose a depth-first search (DFS) is performed on this graph with vertex 1 as root. Draw the DFS tree with the root on top; present the children of a node from left to right in the order in which they are explored; direct the tree edges from parent to child, use dotted lines for back edges and direct them from the node to the ancestor. $\boxed{5}$

(b) Suppose a breadth-first search (BFS) is performed on this graph, with vertex 2 as root. Draw the BFS tree with the root on top. Draw only the tree edges and direct them from parent to child; present the children of a node from left to right in the order in which they are visited. $\boxed{5}$

2. Recall the representation of a max-heap with $n$ elements as an array $A$ (the indices run from 0 to $n-1$): the value at the root is $A[0]$, the two children of $A[i]$ are $A[2i+1]$ and $A[2i+2]$. We considered the following operations on a heap: (i) $\mathsf{bubbleup}(i)$, which starts at $A[i]$ and repeatedly swaps the element with its parent whenever the parent's value is

smaller; (ii) bubbledown($i$), which starts at $A[i]$ and repeatedly swaps the element with the larger of its two children, until the element reaches a location where it is at least the value of its children.

(a) Suppose the array $A$ is turned into a max-heap by applying the operations bubbleup($i$), for $i = 0, 2 \ldots, n-1$. How many comparisons will be performed in the worst case? $\boxed{3}$

Answer: $\theta($____________$)$

(We say that $f = \theta(g)$ if $f = O(g)$ and $g = O(f)$; so the answer you write down should be both an upper bound and a lower bound, ignoring constant factors.)

(b) Consider the following array with ten characters. $\boxed{7}$

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| S | T | A | T | I | S | T | I | C | S |

Suppose this array is turned into a max-heap (letters that appear earlier in the alphabetical order are considered smaller) by applying the operations bubbledown($i$), for $i = 9, \ldots, 1$. What heap will be produced at the end?

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |

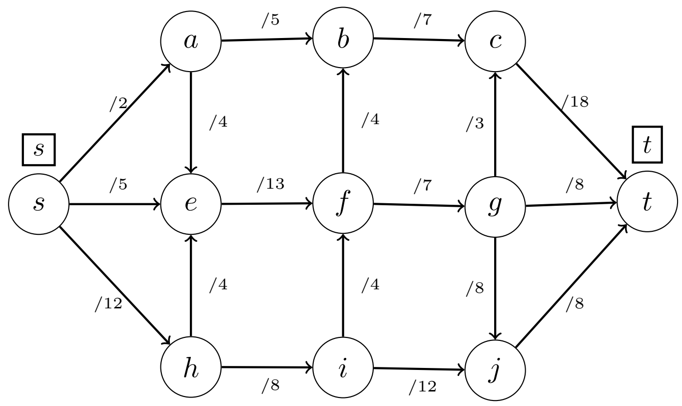
3. (a) Let $G = (V, E)$ be a directed weighted graph with vertex set $\{1, 2, \ldots, n\}$; suppose each edge $(u, v) \in E$ has a length, given by $\ell((u, v))$. We wish to fill a two-dimensional array $A$ indexed by $V \times V$, where $A[u, v]$ is the length of the shortest path from $u$ to $v$ (it is set to $\infty$, if there is no path from $u$ to $v$ in $G$). Assume that $G$ has no negative length cycles. Fill in the blanks in the recurrence below such that $A_{n-1} = A$. $\boxed{5}$

$$A_1[u, v] = \begin{cases} \ell((u, v)) & \text{if } (u, v) \in E \\ \underline{\hspace{3cm}} & \text{if } u = v \\ \underline{\hspace{3cm}} & \text{otherwise} \end{cases} ;$$

and for $i = 2, \ldots, n-1$,

$$A_i[u, v] = \min\{A_{i-1}[u, v], \min\{A_{i-1}[u, w] + \underline{\hspace{3.5cm}} : (w, v) \in E\}\}.$$

(b) In the following *directed* network with capacities, determine a maximum $(s, t)$-flow and a minimum $(s, t)$-cut. Write the flow on the edges; write $s$ or $t$ next to each vertex to indicate on which side of the cut the vertex falls (the vertices $s$ and $t$ already have $\boxed{\text{s}}$ and $\boxed{\text{t}}$ written next to them). Note that a label of $a/c$ on an edge indicates that the edge carries a flow of $a$ and its capacity is $c$. $\boxed{10}$




The value of the flow is __________. The capacity of the cut is __________.

(c) How would you assign binary prefix-free codewords to the letters a, b, c ,d ,e, f, g, h if the frequencies are as follows? 5

`a:2   b:1   c:1   d:7   e:5   f:8   g:13   h:21`

You may state the codeword over the alphabet $\{0, 1\}$ for each of the letters, or you may represent the codewords using a rooted binary tree. (Work the answer out on a separate sheet and only write the final solution below.)

4. Consider the following *heaviest path* problem.
**Input:** An undirected tree $T = (V, E)$, where $V = \{1, 2, \ldots, n\}$, and an array $W$ indexed by $V$, where $W[v]$ denotes the weight of $v$.
**Weight of a path:** For a path $p$ in $T$, let the weight of the path be the sum of the weights of the vertices in $p$, that is, $\sum_{v \in p} W[v]$.
**Output:** A path $p^*$ in $T$ of maximum weight.

We wish to find a path of maximum weight in the given tree $T$.

(a) Assume that all weights are non-negative. Describe precisely how such a path can be found in linear time (assume that the graph is given via adjacency lists). 15

(b) How would you modify the algorithm so that it works even if some of the weights are negative? Your algorithm should still run in linear time. $\boxed{5}$

5. Let $G = (V, W, E)$ be a bipartite graph. An independent set in $G$ is a subset $I$ of $V \cup W$ such that no edge has both end points in $I$.

   (a) Show that the size of the largest independent set in $G$ is $|V| + |W| - |M|$, where $M$ is a maximum matching in $G$. $\boxed{5}$

   (b) Describe an efficient algorithm to determine the largest independent set in a bipartite graph, when the graph is given in the form of adjacency lists. Note that your algorithm should output the independent set, not only its size. $\boxed{15}$

6. Consider the following $k$-COLOURING problem on undirected graphs.
   **Input:** A graph $G$ on $n$ vertices and a positive integer $k \leq n$.
   **Output:** 1 if $G$ can be properly coloured using $k$ colours, and 0 otherwise.

   (a) Show that $k$-COLOURING is in NP. $\boxed{5}$

   (b) Show a reduction from 3-COLOURING to SAT. That is, given a graph $G$, show how you will construct a Boolean expression $\phi(G)$ in polynomial time, such that $\boxed{15}$

   $$G \text{ is 3-colourable iff } \phi(G) \text{ is satisfiable.}$$