COMP 3804 — Winter 2026 — Problem Set 3

Some useful facts:

1. $1 + 2 + 3 + \cdots + n = n(n+1)/2$.
2. for any real number $x > 0$, $x = 2^{\log x}$.
3. For any real number $x \neq 1$ and any integer $k \geq 1$,
$$1 + x + x^2 + \cdots + x^{k-1} = \frac{x^k - 1}{x - 1}.$$
4. For any real number $0 < \alpha < 1$,
$$\sum_{i=0}^{\infty} \alpha^i = \frac{1}{1-\alpha}.$$

Master Theorem:

1. Let $a \geq 1$, $b > 1$, $d \geq 0$, and
$$T(n) = \begin{cases} 1 & \text{if } n = 1, \\ a \cdot T(n/b) + \Theta(n^d) & \text{if } n \geq 2. \end{cases}$$
2. If $d > \log_b a$, then $T(n) = \Theta(n^d)$.
3. If $d = \log_b a$, then $T(n) = \Theta(n^d \log n)$.
4. If $d < \log_b a$, then $T(n) = \Theta(n^{\log_b a})$.

Question 1: You are given a min-heap $A[1 \ldots n]$ and a variable *largest* that stores the largest number in this min-heap.

In class, we have seen algorithms INSERT(A, x) (which adds the number x to the min-heap and restores the heap property) and EXTRACTMIN(A) (which removes the smallest number from the heap and restores the heap property).

Explain, in a few sentences, how these two algorithms can be modified such that the value of *largest* is correctly maintained. The running times of the two modified algorithms must still be $O(\log n)$.

Question 2: Let m be a large integer and consider m non-empty sorted lists $L_1, L_2, \ldots, L_m$. All numbers in these lists are integers. Let n be the total length of all these lists.

Describe an algorithm that computes, in $O(n \log m)$ time, two integers a and b, with $a \leq b$, such that

- each list L_i contains at least one number from the set $\{a, a+1, \ldots, b\}$ and
- the difference $b - a$ is minimum.

For example, if $m = 4$,

$$\begin{aligned} L_1 &= (2, 3, 4, 8, 10, 15) \\ L_2 &= (1, 5, 12) \\ L_3 &= (7, 8, 15, 16) \\ L_4 &= (3, 6), \end{aligned}$$

then the output can be $(a, b) = (4, 7)$ or $(a, b) = (5, 8)$.

As always, justify the correctness of your algorithm and explain why the running time is $O(n \log m)$.

Hint: Use a min-heap of size m and use Question 1. For the example, draw it like this, then stare at it until you "see" the algorithm:

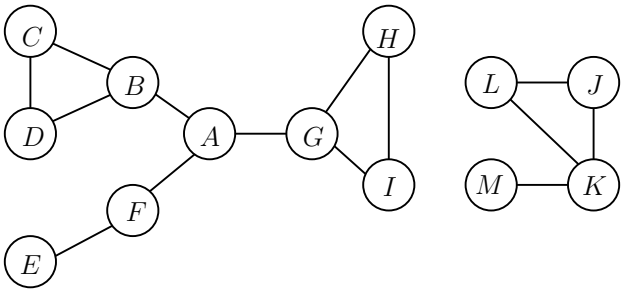
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	2	3	4				8		10					15	
1				5							12				
						7	8							15	16
		3			6										

Question 3: Let $G = (V, E)$ be an undirected graph. A *vertex coloring* of G is a function $f : V \to \{1, 2, \ldots, k\}$ such that for every edge $\{u, v\}$ in E, $f(u) \neq f(v)$. In words, each vertex u gets a "color" $f(u)$, from a set of k "colors", such that the two vertices of each edge have different colors.

Assume that the graph G has exactly one cycle with an odd number of vertices. (The graph may contain cycles with an even number of vertices.)

What is the smallest integer k such that a vertex coloring with k colors exists? As always, justify your answer.

Question 4: Consider the following undirected graph:



Question 4.1: Draw the DFS-forest obtained by running algorithm DFS on this graph. Recall that algorithm DFS uses algorithm EXPLORE as a subroutine.

In the forest, draw each tree edge as a solid edge, and draw each back edge as a dotted edge.

Whenever there is a choice of vertices (see the two lines labeled (*)), pick the one that is alphabetically first.

Question 4.2: Do the same, but now, whenever there is a choice of vertices (see the two lines labeled (*)), pick the one that is alphabetically last.

```
Algorithm DFS(G):
for each vertex u
do visited(u) = false
endfor;
cc = 0;
for each vertex v        (*)
do if visited(v) = false
    then cc = cc + 1
         EXPLORE(v)
    endif
endfor
```

```
Algorithm EXPLORE(v):
visited(v) = true;
ccnumber(v) = cc;
for each edge {v, u}     (*)
do if visited(u) = false
    then EXPLORE(u)
    endif
endfor
```

Question 5: Prove that an undirected graph $G = (V, E)$ is bipartite if and only if G does not contain any cycle having an odd number of edges.

Question 6: Since Katy Perry and Justin Trudeau miss each other very much, they decide to meet. Katy and Justin live in a connected, undirected, non-bipartite graph $G = (V, E)$. Katy lives at vertex k, whereas Justin lives at vertex j.

Katy and Justin move in steps. In each step, Katy must move from her current vertex to a neighboring vertex, and Justin must move from his current vertex to a neighboring vertex.

Both Katy and Justin have complete knowledge of the graph. They are in constant communication so that each person knows where the other person is.

Prove that there exists a moving strategy such that Katy and Justin meet each other at the same vertex.

Hint: While moving around in the graph, each of Katy and Justin may visit the same vertex more than once, and may traverse the same edge more than once.

If the graph G consists of the single edge $\{k, j\}$, then they will never be at the same vertex. But in this case G is bipartite.

If the graph contains a path having 8 edges, Katy lives at one end-vertex, and Justin lives at the other end-vertex, will they ever be at the same vertex?

Question 5 is useful.