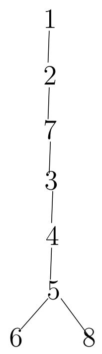
CSCI2100: Final Exam Solutions

Problem 1 (15%). F, T, F, T, F, T, F, T, F, T.

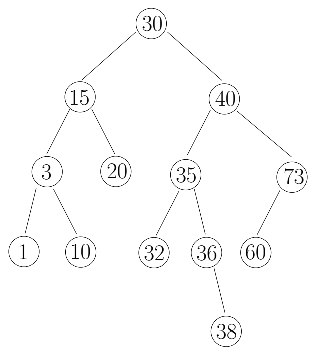
Problem 2 (5%).



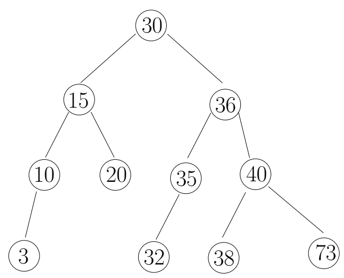
Vertices' time intervals:
Vertex 1: [1, 16]
Vertex 2: [2, 15]
Vertex 7: [3, 14]
Vertex 3: [4, 13]
Vertex 4: [5, 12]
Vertex 5: [6, 11]
Vertex 6: [7, 8]
Vertex 8: [9, 10]

Problem 3 (15%).

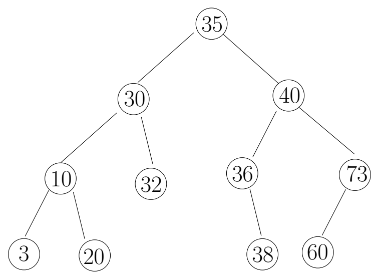
After inserting 1:



After deleting 60:



After deleting 15:



Problem 4 (8%). Initalize an empty min-heap H. When an element e needs to be added to S, check first whether H has at most $k-1$ elements. If so, insert e into H. Otherwise, let x_{min} be the key at the root of H. If $x_{min} > e$, ignore e. Otherwise, perform a del-min on H and then insert e into H.

Problem 5 (7%). Create a hash table on S_1 in $O(n)$ time. Then, for each element $e \in S_1$, probe the hash table to check whether $e \in S_1$; if so, report e.

Problem 6 (10%). We will first prove that a DAG G has a source vertex if and only if it has exactly one vertex with in-degree 0. Let us start with the "only if" direction ($\Rightarrow$). Suppose that s is a source vertex of G, but G has two distinct vertices u, v with in-degree 0. W.o.l.g., suppose that $s \neq u$. Then, s cannot have a path to u becauuse every such path must use at least one in-coming edge of u.

Next, we prove the "if" direction ($\Leftarrow$). Let s be the only vertex with an in-degree 0. For any other vertex u in G, we argue that s must have a path to u. Suppose that this is not true. Define $u_0 = u$. Inducitively, suppose that we have found a vertex u_i ($i \geq 0$) such that s has no path to u_i. Let v be any in-neighbor of u_i; note that v definitely exists because the in-degree of u_i is at least 1 (note that $s \neq u_i$ because s has a path to itself). As s has no path to u_i, it cannot have a path to v, either. Define $u_{i+1} = v$. As G is a DAG, repeating the above process will give distinct vertices $u_0, u_1, u_2, \ldots$. This gives a contradiction because G has only a finite number of vertices.

For an algorithm, first run DFS to decide whether G is acyclic. If so, check whether G has exactly one vertex with in-degree 0. These steps can be done in $O(|V| + |E|)$ time.

Problem 7 (10%). Maintain a BST T on S but at all times remember the smallest integer of S, denoted as x_{min}. We can find x_{min} by descending into the leftmost leaf of T in $O(\log n)$ time. Support the four operations as follows:

- `Insert`(e): insert e into T as in a normal BST. After the insertion, update x_{min}. The total cost is $O(\log n)$.
- `Delete`(e): delete e from T as in a normal BST. After the deletion, update x_{min}. The total cost is $O(\log n)$.
- `Delmin`: first delete x_{min} from T and then update x_{min} in the resulting tree. The total cost is $O(\log n)$.
- `Findmin`: simply return x_{min}.

Problem 8 (20%).

- There are $2|E|$ choices to set u_1, u_2 such that $\{u_1, u_2\} \in E$. Similarly, there are $2|E|$ choices to set u_3, u_4 such that $\{u_3, u_4\} \in E$. Once u_1, u_2, u_3, u_4 are chosen, whether $\{u_1, u_2\}, \{u_2, u_3\}, \{u_3, u_4\}, \{u_4, u_1\}$ make a 4-cycle is determined. Therefore, the number of 4-cycles in G cannot exceed $4|E|^2$.

- First, build a structure that, given any two vertices $u, v \in V$, can determine in constant expected time if $\{u, v\}$ is an edge in G. For this purpose, for each vertex $u \in V$, build a hash table on its adjacency list so that, give any vertex $v \in V$, we can check in $O(1)$ expected time whether v appears in the adjacency list. Constructing the hash tables for all $u \in V$ takes $O(|V| + |E|)$ time.

  To find all 4-cycles, apply the following algorithm.

  > **for** each edge $\{a, b\} \in E$ **do**
  >   **for** each edge $\{c, d\} \in E$ **do**
  >   **if** $\{b, c\} \in E$ and $\{d, a\} \in E$ **then**
  >   report cycle $\{a, b\}, \{b, c\}, \{c, d\}, \{d, a\}$
  >   **if** $\{a, c\} \in E$ and $\{d, b\} \in E$ **then**
  >   report cycle $\{b, a\}, \{a, c\}, \{c, d\}, \{d, b\}$
  >   **if** $\{b, d\} \in E$ and $\{c, a\} \in E$ **then**
  >   report cycle $\{a, b\}, \{b, d\}, \{d, c\}, \{c, a\}$
  >   **if** $\{a, d\} \in E$ and $\{c, b\} \in E$ **then**
  >   report cycle $\{b, a\}, \{a, d\}, \{d, c\}, \{c, b\}$

  Because each if-statement takes $O(1)$ time, the algorithm runs in $O(|E|^2)$ time overall.

- If G is a clique, it has $\Theta(|E|^2)$ 4-cycles. Simply outputting all the cycles incurs $\Omega(|E|^2)$ time.

Problem 9 (10%). Initialize a heap H with the top-left number of M as the only element. In general, if a number of M is inserted in H, mark it in M. Repeat the following k times. Perform a delete-min from H. Let x be the number returned. Output x, and insert in H the numbers on its right and below it respectively, provided that (i) they exist, and (ii) they are unmarked. As H can contain only $O(k)$ vertices, the total running time is $O(k \log k)$.