This assignment is **due on April 28** and should be submitted on Gradescope. All submitted work must be *done individually* without consulting someone else's solutions in accordance with the University's "Academic Dishonesty and Plagiarism" policies.

As a first step go to the last page and read the section: Advice on how to do the assignment.

Problem 1. (20 points)
Consider the following algorithm for testing if a given binary tree has the binary search tree property.

```
1: function TEST-BST($T$)                                  ▷ Assumes $T$ has $n$ distinct keys
2:     for $u \in T$ do
3:         if $u.left \neq null$ and $u.key <$ RIGHTMOST-DESC$(u.left).key$ then
4:             return False
5:         if $u.right \neq null$ and $u.key >$ LEFTMOST-DESC$(u.right).key$ then
6:             return False
7:     return True
```

where RIGHTMOST-DESC(v) is v if $v.right = null$ and RIGHTMOST-DESC$(v.right)$ otherwise.

Your task is to either prove the correctness of this algorithm or provide a counter example where it fails to return the correct answer.

Problem 2. (40 points)
Suppose you have k sorted arrays $A_1, A_2, \dots, A_k$ storing $n = |A_1| + |A_2| + \cdots + |A_k|$ distinct keys. We want a data structure for maintaining a pinning pair (x, y), where $x < y$ are keys in the union of the arrays. The data structure must support the following operations:

- INITIALIZE(): where x is set to the smallest key and y is set to the largest key in the union
- LEFT-FORWARD(): where x is set to the key that comes after x in the union (in sorted order)
- RIGHT-BACKWARD(): where y is set to the key that comes before y in the union (in sorted order)

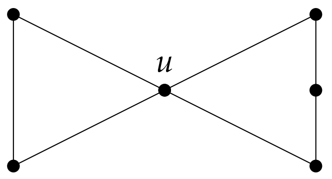
Your task is to design a data structure that uses $O(k)$ space (not counting the space used by the arrays), INITIALIZE takes $O(k)$ time and the other operations run in $O(\log k)$ time. Remember to

a) Design a data structure that supports the required operations in the required time and space.
b) Briefly argue the correctness of your data structure and operations.
c) Analyse the running time of your operations and space of your data structure.

Problem 3. (40 points)
Let $G = (V, E)$ be a connected undirected graph. We define the *risk factor* of a vertex u as the number of pairs of vertices that cannot reach one another after removing u from G; more formally,

$$\text{risk factor of } u = |\{(x, y) \in V - u \times V - u : \nexists \text{ } x\text{-}y \text{ path in } G[V - u]^{1}\}|.$$

For example, in the following graph u has risk factor 6 because removing u disconnects the graph into two connected components having 2 and 3 nodes respectively.



Design an algorithm that computes the risk factor of a given vertex u in $O(n + m)$ time. Remember to:

a) describe your algorithm in plain English,

b) prove it correctness, and

c) analyze its time complexity.

To get full marks your algorithm must run in $O(n + m)$ time. If you cannot make the stated bound, you can get partial credit by submitting a slower algorithm.

[1] Recall that given a graph $G = (V, E)$ and a subset $S \subseteq V$, the notation $G[S]$ denotes the subgraph of G whose vertex set is S and whose edge set is those edges in E having both endpoints in S.

Advice on how to do the assignment

- Assignments should be typed and submitted as pdf (no pdf containing text as images, no handwriting).
- Start by typing your student ID at the top of the first page of your submission. Do **not** type your name.
- Submit only your answers to the questions. Do **not** copy the questions.
- When asked to give a plain English description, describe your algorithm as you would to a friend over the phone, such that you completely and unambiguously describe your algorithm, including all the important (i.e., non-trivial) details. It often helps to give a very short (1-2 sentence) description of the overall idea, then to describe each step in detail. At the end you can also include pseudocode, but this is optional.
- In particular, when designing an algorithm or data structure, it might help you (and us) if you briefly describe your general idea, and after that you might want to develop and elaborate on details. If we don't see/understand your general idea, we cannot give you marks for it.
- Be careful with giving multiple or alternative answers. If you give multiple answers, then we will give you marks only for "your worst answer", as this indicates how well you understood the question.
- Some of the questions are very easy (with the help of the slides or book). You can use the material presented in the lecture or book without proving it. You do not need to write more than necessary (see comment above).
- When giving answers to questions, always prove/explain/motivate your answers.
- When giving an algorithm as an answer, the algorithm does not have to be given as (pseudo-)code.
- If you do give (pseudo-)code, then you still have to explain your code and your ideas in plain English.
- Unless otherwise stated, we always ask about worst-case analysis, worst case running times, etc.
- As done in the lecture, and as it is typical for an algorithms course, we are interested in the most efficient algorithms and data structures.
- If you use further resources (books, scientific papers, the internet,...) to formulate your answers, then add references to your sources and explain it in your own words. Only citing a source doesn't show your understanding and will thus get you very few (if any) marks. Copying from any source without reference is considered plagiarism.