

Be sure to prove correctness and analyze runtime for all algorithms you provide. We expect proofs at a similar level of detail to Kruskal's proof in Chapter 2 of Anupam's notes (MST) (linked on course website).

**Please solve problem #1 by yourself, you are allowed to collaborate for the other problems.** Please limit yourself to groups of two (or at most three) people. You can use Piazza to find group mates. We adhere to a "whiteboard" collaboration policy: you should discuss with your partner on whiteboards (or equivalents) but then should go away and write down your solutions yourself. You must not share written work. **You must write down the names of the person(s) you collaborate with.** In general, you should solve the homework problems using only to material we refer you to (or put on the course page), and not books or other online resources. **Consulting an LLM such as ChatGPT is not allowed.** If you have reason to use any resources we point you to (except the course notes), please cite them.

1. **Conference Talks** (25 pts)

**Please solve this problem by yourself.** You are at a conference and there are  $n$  talks given. Each has a starting time  $s_i$  and ending time  $e_i$ . If you attend a talk you must sit through the entire talk (from start to finish). Naturally, you can only attend one talk at a time (i.e., you can't attend a pair of talks if their intervals intersect). Give an algorithm that, in  $O(n \log(n))$  time, finds the maximum number of talks you can attend. Prove correctness.

2. **Snipping Trees** (25 pts)

Consider a data structure that maintains trees. It starts with a single node  $r$  and support the following operations:

- **Add**( $v, c$ ): adds a new child  $c$  to the node  $v$ .
- **Snip**( $v$ ): makes  $v$  and all its ancestors except the root (i.e. all nodes on the path from  $v$  to the root, excluding the root) children of the root. (**Snip**( $r$ ) will do nothing and is not allowed). See Figure 1 for an example.

The cost of an **Add** operation is 1. The cost of a **Snip**( $v$ ) is the length of the path (number of edges in the path) from  $v$  to the root.

- After any sequence of operations, what can you say about the maximum depth of the tree? Prove your bound is tight.
- Show via **banker's method** that any interleaved sequence of **Adds** and **Snips** has an amortized cost at most 2 per operation.
- Show the same via **potential function**. Your potential function doesn't need intuitive meaning - just needs to work!

3. **Colorful Spanning Tree** (25 pts)

Given a graph  $G = (V, E)$  with  $n$  vertices and  $m$  edges, where each edge is colored red or blue.

- Give an algorithm in  $O(m \log n)$  time to find a spanning tree of  $G$  with at least  $k$  red edges, or report **None** if no such spanning tree exists. Can you find a spanning tree with the maximum number of red edges?

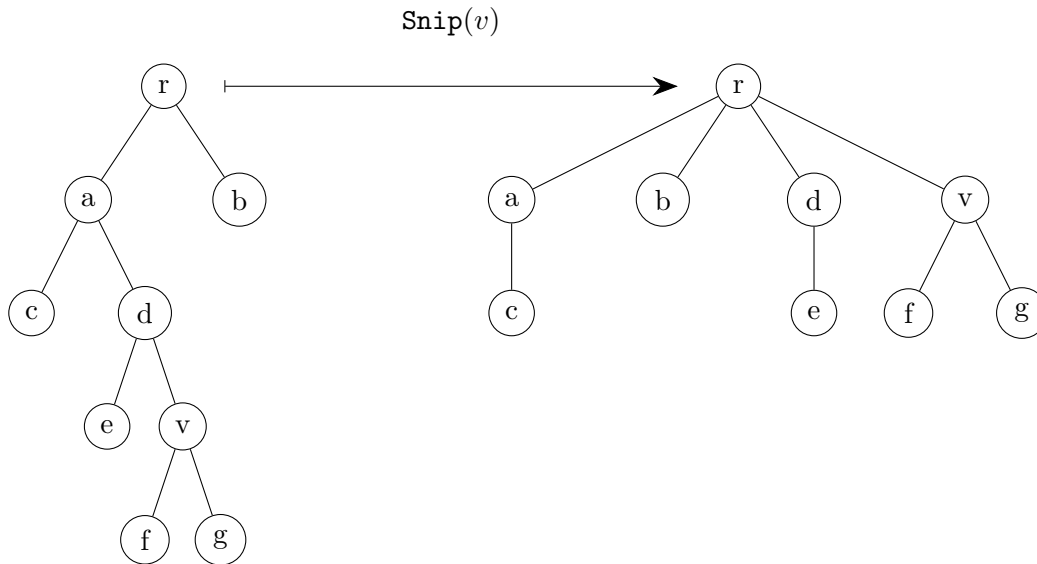


Figure 1: Example of  $\text{Snip}(v)$  operation

- (b) **Exchange Theorem:** Prove that if spanning trees  $T_1$  and  $T_2$  exist with  $R(T_1) \leq k \leq R(T_2)$ , where  $R(T)$  = number of red edges in  $T$ , then there exists a spanning tree with exactly  $k$  red edges.
- (c) Give an algorithm to output a spanning tree of  $G$  that contains *exactly*  $k$  red edges and  $n - k - 1$  blue edges. Again, report **None** if no such spanning tree exists. Your algorithm should run in  $O(mn)$  time (or less, this is a generous bound).

#### 4. Fast Multiplication (25 pts)

Given two  $n$ -bit strings, you are asked to output their product in the form of a binary string. For instance, if you received 101 and 11 as inputs, you should output 1111. The time complexity of the algorithm in this problem is the number of basic bitwise operations, such as AND, OR, XOR. You may also assume that you have access to primitive operations of addition and subtraction of  $n$ -bit strings which use  $O(n)$  basic bitwise operations.

- (a) Show the naïve "grade school" multiplication algorithm for  $n$ -bit integers takes  $\Theta(n^2)$  time.
- (b) Show a faster algorithm with time complexity  $O(n^{\log_2 3}) \approx O(n^{1.585})$  time.

**Hint 1:** Use the same trick as we did in class to solve matrix multiplication recursively.

**Hint 2:** Multiplying two 2-bit numbers  $ab$  and  $cd$  can be done with 4 multiplications. Can you do it with just 3? (Additional hint: expand  $(a - b)(c - d)$ .)