

Your Name

September 25, 2026

**6.5240 Problem Set 2**

Collaborators: *Firstname Lastname*

Due: **10pm**, October 7, 2026

1. **Sublinear testing of connectedness.** Given a graph  $G$  of max degree  $d$ , and a parameter  $\epsilon$ , give an algorithm which has the following behavior: if  $G$  is connected, then the algorithm should pass with probability 1, and if  $G$  is  $\epsilon$ -far from connected (at least  $\epsilon dn$  edges must be added to connect  $G$ ), then the algorithm should fail with probability at least  $3/4$ . If  $G$  is neither connected nor  $\epsilon$ -far from connected, the algorithm is allowed to output anything. Your algorithm should look at a number of edges that is independent of  $n$ , and polynomial in  $d, 1/\epsilon$ .

For this homework set, when proving the correctness of your algorithm, it is acceptable to show that if the input graph  $G$  is likely to be passed, then it is  $\epsilon$ -close to a graph  $G_0$  which is connected, without requiring that  $G_0$  has degree at most  $d$ .

2. **Sublinear test of diameter.** The diameter of an unweighted graph is the maximum distance between any pair of nodes. Give an algorithm for graphs with degree at most  $d$  (where  $d$  is a constant and the graph is represented in the adjacency list model) that have low diameter.

The algorithm should have the following specific behavior:

- (a) Graphs with diameter at most  $D$  are always accepted.
- (b) Graphs which are  $\epsilon$ -far (that is, at least  $\epsilon dn$  edges must be added) from having diameter  $4D + 2$  are failed with probability at least  $2/3$ .
- (c) Any output is acceptable for graphs with diameter larger than  $D$  but not  $\epsilon$ -far from having diameter  $4D + 2$ .
- (d) The query complexity of the algorithm should be  $O(1/\epsilon^c)$  for some constant  $c \geq 1$ .

For this homework set, when proving the correctness of your algorithm, it is acceptable to show that if the input graph  $G$  is likely to be passed, then it is  $\epsilon$ -close to a graph  $G_0$  which has diameter  $4D + 2$ , without requiring that  $G_0$  has degree at most  $d$ .

3. **MST Edge Weight Sampling** In this problem, we use Prim's algorithm to sample edge weights from an MST of a connected graph  $G = (V, E)$  with  $n$  vertices and  $m$  edges. Assume that all edge weights are positive integers in  $\{1, 2, \dots, W\}$ . The graph is given in the adjacency-list model, with the following three available queries:

- Vertex query: Return a uniform random vertex  $v \in V$ ;
- Degree query: Give a vertex  $v \in V$ , return  $\deg(v)$ ;

- Neighborhood query: Give a vertex  $v \in V$  and  $i \in [n]$ , return the  $i$ -th neighbor of  $v$  and the weight of the corresponding edge or return *fail* if  $\deg(v) < i$ .

We also assume that  $m$  is known to the algorithm.

Recall the description of Prim's algorithm: We construct a sequence of trees  $T_1, T_2, \dots, T_n$ , where  $T_1$  contains a single starting vertex  $s$  chosen arbitrarily. For  $1 \leq i \leq n-1$ ,  $T_{i+1}$  is obtained from  $T_i$  by adding a minimum-weight edge that connects a vertex in  $T_i$  and a vertex outside  $T_i$ . The final tree  $T_n$  is an MST.

Our algorithm is the following:

1. Sample  $s$  uniformly at random from  $V$ . Independently, sample a uniform real  $Y \in (0, 1)$ .
2. If  $Y < 1/n$ , the algorithm returns *fail*. Otherwise, we simulate Prim's algorithm with an early termination: let  $k = \lceil 1/Y \rceil$ . After  $T_k$  is constructed, we terminate the algorithm and output the *highest edge weight* in  $T_k$ .

Next, we fix  $w \in \{1, \dots, W\}$  and analyze the probability that the algorithm outputs  $w$ . We define  $E^{(i)}$  to be the set of edges of weight at most  $i$  in  $E$ , and  $G^{(i)} = (V, E^{(i)})$ . For a vertex  $v \in V$ , define  $C^{(i)}(v)$  to be the set of vertices in the connected component of  $G^{(i)}$  containing  $v$ .

In a), b) and c), we focus on the step 2 in the algorithm, so we also assume that  $s$  is fixed.

- a) In the Prim's algorithm, what relation between  $i$  and  $|C^{(w-1)}(s)|$  ensures that the maximum edge weight in  $T_i$  is smaller than  $w$ ? What relation between  $i$  and  $|C^{(w)}(s)|$  ensures that the maximum edge weight in  $T_i$  is larger than  $w$ ? Directly write your answer without explanation.
- b) Give a concise (at most one line) expression describing the condition that step 2 in the algorithm outputs  $w$ , in terms of  $Y$ ,  $|C^{(w-1)}(s)|$  and  $|C^{(w)}(s)|$ .
- c) Based on b), determine the probability that the algorithm outputs  $w$ .

Now we consider the full algorithm, where  $s$  is chosen randomly among  $V$ .

- d) By averaging your solution in c) for every  $s \in V$ , argue that the algorithm outputs  $w$  with probability  $c_w/n$ , where  $c_w$  is the number of edges of weight  $w$  in any MST of the graph  $G$ .<sup>1</sup>
- e) Show that, when  $s$  and  $Y$  are fixed, we can implement step 2 in  $O(d_{\max}/Y)$  queries, where  $d_{\max}$  is the maximum degree of  $G$ . Conclude that the algorithm can be implemented in  $O(d_{\max} \log n)$  expected number of queries.

---

<sup>1</sup>It is proved in Lemma 5 of Scribe Note 2 that every MST in a graph  $G$  has the same number of edges of weight  $w$ , so  $c_w$  is well-defined.