

**Question 1** *Silly Sort*

There is an algorithm, “Silly-sort,” inspired by Stooge-sort. If the input size,  $n$ , is 1 or 2, then the algorithm sorts the input immediately. Otherwise, it is called recursively on various ranges of elements as follows. To simplify matters, we assume that the input size is always a power of 2.

```

silly-sort( $A, i, j$ )                                /*  $0 \leq i \leq j \leq |A|$  */
   $n \leftarrow j - i$ 
  if  $n = 2$  then
    if  $A[i] > A[j - 1]$  then
      Swap  $A[i]$  and  $A[j - 1]$ 
  else if  $n > 2$  then
     $m \leftarrow \frac{n}{4}$ 
    silly-sort( $A, i, i + 2m$ )                        /* Sort the first half */
    silly-sort( $A, i + m, i + 3m$ )                    /* Sort the middle part */
    silly-sort( $A, i + 2m, j$ )                        /* Sort the last half */
    silly-sort( $A, i + m, i + 3m$ )                    /* Sort the middle part again */
    silly-sort( $A, i, i + 2m$ )                        /* Sort the first half again */

```

**1.1** What is the output of **silly-sort** on the following array:  $[9, 3, 5, 4]$


**1.2** Unfortunately, **silly-sort** does not always correctly sort its output. Find a permutation of  $[1, 2, 3, 4]$  such that the output of **silly-sort** is not sorted.

**1.3** Characterize the running time,  $T(n)$ , for **silly-sort**, using a recurrence equation, and use the Master Theorem to determine an asymptotic bound for  $T(n)$ .

**1.4** Define a new algorithm, **silly-sort-fixed**, that always correctly sorts its input (you can assume the size of the input is a power of 2). **silly-sort-fixed** needs to keep the spirit of **silly-sort**: it can only perform recursive calls to itself using ranges based on  $i, j$ , and multiples of  $m$ , and it cannot have any merging procedure. Provide the pseudo-code for **silly-sort-fixed**. You do not need to explain in English how and why the algorithm works.

**1.5** Characterize the running time,  $T(n)$ , for **silly-sort-fixed**, using a recurrence equation, and use the Master Theorem to determine an asymptotic bound for  $T(n)$ .

**Question 2** *Tiling  $T_n$  with  $T_2$  polyominoes*

The result proven in this exercise is that a tiled triangle of side  $n \geq 6$ , known as  $T_n$ , can be tiled with smaller triangles, , if and only if  $n \equiv 0 \pmod{3}$  or  $n \equiv 2 \pmod{3}$ .

For any subquestion, you may assume known the answer to the previous subquestions. For instance, you can use the fact that a tiling for  $n = 6$  exists in Question 2.3 even you did not find the tiling requested in Question 2.2. In my opinion, Question 2.3 is the hardest, so don't hesitate to try solving Question 2.4 even if you're out of luck with Question 2.3.

**2.1** Suppose that  $n = 3k + 1$  for some positive integer  $k \geq 1$ . Prove that no tiling is possible.

**2.2** Figure 1 gives a tiling for  $n = 9$ . Find a tiling for  $n = 6$ . You may describe your solution with a (clear) drawing or with a list of triples, up to you.

Figure 1: Tiling for  $T_9$

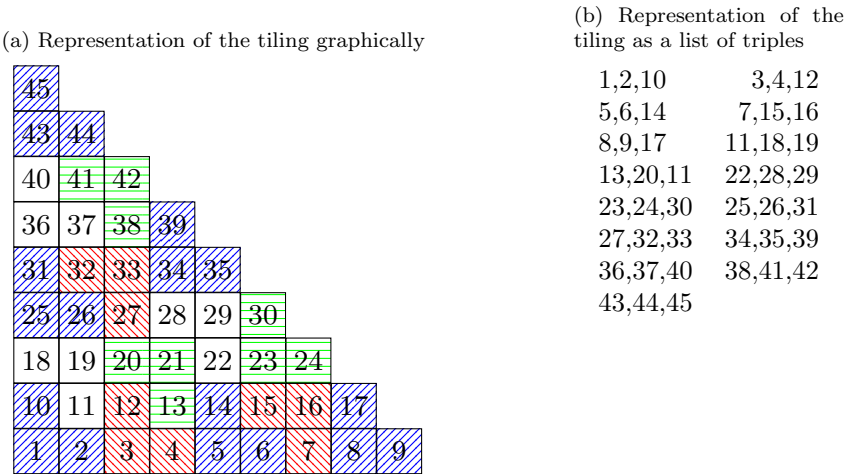
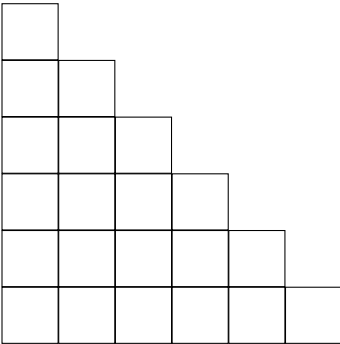


Figure 2:  $T_6$



**2.3** Suppose that  $n = 3k$  for some positive integer  $k \geq 2$ . Prove by induction that a tiling is possible.

**2.4** Suppose that  $n = 3k + 2$  for some positive integer  $k \geq 2$ . Prove that a tiling is possible.

### Question 3 *Rural rent-a-car*

Sarah is planning a trip to a rural area which consists of  $T$  towns separated by a network of several main roads. Due to the remoteness of the area, there aren't a lot of travel options between two towns. In particular, for any two towns there is exactly one possible path from one to the other (which may go over multiple roads).

Sarah is planning to rent a car with an effective range of  $D$  kms to travel between the towns and wants to evaluate how effective it would be as a mode of transportation. Every town has a petrol station so she would be able to travel along any path which consists of roads of length less than  $D$ , refueling at each successive town.

**3.1** We will model our problem as a graph problem. That is, we construct a graph  $G$  with vertices  $1, 2, \dots, T$  and edges representing our roads between towns whose weights are their total distance. What can you say about the topology of this graph?

**3.2** Design an algorithm that computes the total number of paths that Sarah may traverse in her desired rental car. For simplicity, for any two towns  $A$  and  $B$  we consider the path from town  $A$  to town  $B$  to be equivalent to the path from town  $B$  to town  $A$ . Describe your algorithm using pseudo-code and explain it in English.

**3.3** What is the worst-case time complexity of your algorithm? Explain your reasoning.

### Question 4 *Tree Search*

You are given a connected acyclic graph,  $G$ , (also known as a unrooted tree) with  $n$  nodes. One of these nodes contains a prize, but you don't know which node this is. However, you are able to query nodes – each query will tell you which edge from that node you must travel in order to get to the prize, or “Yes” if you've picked the prize node. Your task is to determine which node has the prize using at most  $\lg n$  queries.

**4.1** Design an algorithm to solve this problem if  $G$  is a line graph.

**4.2** Solve this problem for Figure 4, using at most 2 queries. It's sufficient to fill in the table provided; an example is given in Figure 3.

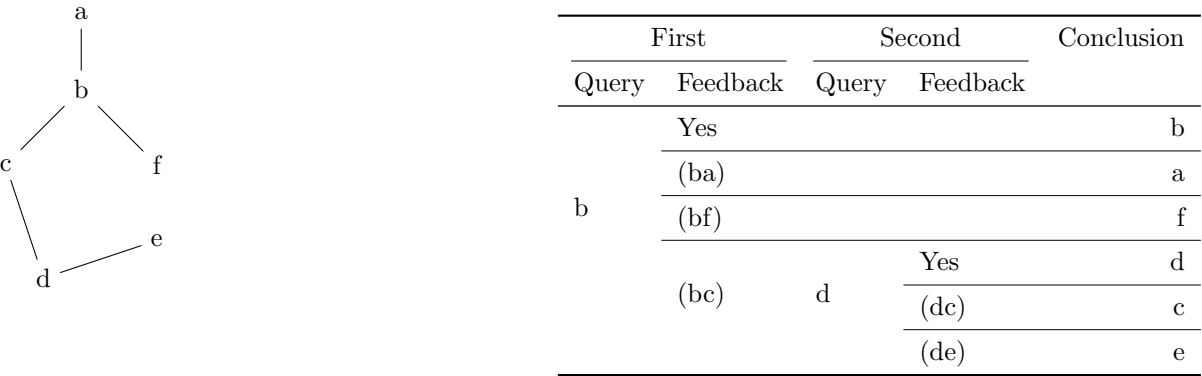


Figure 3: Example input and strategy for Question 4.2.

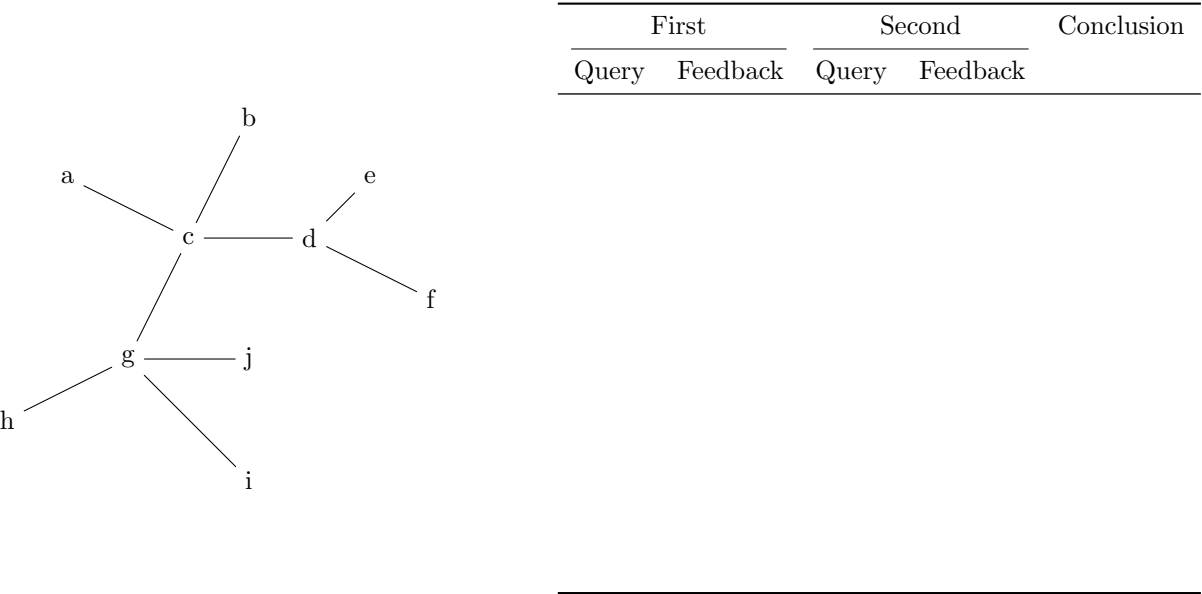


Figure 4: Target input and template strategy for Question 4.2.

**4.3** Design an algorithm to solve this problem for any connected acyclic graph  $G$ , that runs in at most  $O(n^2 \lg n)$ .

**4.4** Design an algorithm to solve this problem for any connected acyclic graph  $G$ , that runs in at most  $O(n \lg n)$ . If you are able to correctly solve this subproblem, you do not need to provide a separate solution for Question 4.3.