Homework 5

Due 10/10/2024

October 3, 2024

1. Give pseudocode for an algorithm to compute the number of *quadratic nonresidues* modulo n, where a quadratic nonresidue mod n is an integer r in the range of 0 to $n-1$ such that there is no integer x in this range where $x^2 \bmod n = r$.

   For example, the quadratic nonresidues of $n = 6$ are 2 and 5, as shown by the table below:

x	$x^2 \bmod 6$
0	0
1	1
2	4
3	3
4	4
5	1

   Thus, the solution for $n = 6$ is 2.

   Your algorithm should be as efficient as possible. Be sure to describe which data structure(s) you're using and why.

2. Dr. Snevets is working on an algorithm that uses a Union-Find to maintain a set partition, but her algorithm also needs to efficiently determine the smallest element in each partition. Answer the following about developing an improved Union-Find that efficiently supports a FindMin(x) operation that can return the minimum element in the same partition as x.

   (a) One way to implement this improved data structure is to store a third array *min* such that *min*[r] is the minimum element in the tree rooted at r, for every root element r. (*min* can equal any value at indexes that are not roots.) Describe how to modify the pseudocode for the Union-Find initialize operation (below) to also initialize *min* appropriately.

```
Input: n: size of the Union-Find to initialize
Output: a Union-Find of size n where every element
        points to itself
1 Algorithm: Union-Find.Initialize
2 uf = Array(n)
3 Initialize uf to 1..n
4 size = Array(n)
5 Initialize size to 1
6 return (uf, size)
```

(b) What is the worst-case time complexity for your modified Initialize algorithm? Justify your answer.

(c) Describe how to modify the Union operation (below) so that *min* always stores the minimum element of the tree at every root (*min* can store anything at an index that is not a root of the Union-Find.)

```
Input: (uf, size): the Union-Find to modify
Input: a: index of one element to union
Input: b: index of another element to union
Output: modified uf that efficiently merges the trees
        containing a and b
1 Algorithm: uf.Union
2 ra = uf.Find(a)
3 rb = uf.Find(b)
4 if size[ra] > size[rb] then
5 |  Swap ra and rb
6 end
7 uf[ra] = rb
8 size[rb] = size[ra] + size[rb]
```

(d) What is the amortized time complexity of your modified Union algorithm? Show your work. Note that the Find operation does not need to be changed from the optimized Union-Find presented in class.

3. Find the worst-case complexity of the hierarchical clustering algorithm below. You may assume that the distance function takes $\Theta(1)$ time to compute.

```
   Input: data: set of data points
   Input: n: size of data
   Input: distance: distance function that takes two data points and
          returns a nonnegative real number
   Input: c: desired number of clusters; must be an integer between 1 and
          n
   Output: single-linkage hierarchical clusters for data
 1 Algorithm: SingleHClust
 2 heap = MinHeap()
 3 for i = 1 to n − 1 do
 4 |   for j = i + 1 to n do
 5 |   |   Insert distance(data[i], data[j]) into heap, along with the
   |   |     corresponding indexes i and j
 6 |   end
 7 end
 8 uf = UnionFind(n)
 9 count = n
10 while count > c do
11 |   (i, j, dist) = heap.DeleteMin()
12 |   if uf.Find(i) ≠ uf.Find(j) then
13 |   |   uf.Union(i, j)
14 |   |   count = count − 1
15 |   end
16 end
17 return uf
```