CS2040S: Data Structures and Algorithms

Contest 1 – Treasure Island

1 Background

Explorers have discovered an island with a centuries-old treasure chest with K locks on it, and a huge chain of N keys. They also have found a device, a TreasureExtractor, that takes in any number of keys and unlocks the chest if all the correct keys for the locks are present. You've been tasked to identify the set of K correct keys.

In each attempt, you can choose a set of $\geq K$ keys to try unlocking the chest with. If all K of the correct keys are in the set, the TreasureExtractor unlocks the chest. For each attempt, you will learn if the chest was unlocked or not, and nothing else.

As discussed in lecture, it just so happens that 122 is the minimum number of attempts needed for $N = 1024, K = 17$. (Can you prove that?)

2 Code Briefing

2.1 Interface Descriptions

Within `code.zip`, you should see the `ITreasureExtractor` interface:

```
public interface ITreasureExtractor {

    public boolean tryUnlockChest(int[] keys);

}
```

The interface provides the following method for you to use in your solution:

- `boolean tryUnlockChest(int[] keys)`:
  Passes a set of keys to the TreasureExtractor, and returns true if the TreasureExtractor unlocks the chest and false otherwise.
  `keys` should be an array of length N of 0's and 1's (represented as integers), where a 1 at index i means that key i is passed to the TreasureExtractor to try to unlock the chest. As per the description above, the array needs to have $\geq K$ 1's.
  For example, for $N = 5, K = 2$, `keys` $= \{1, 0, 1, 0, 0\}$ means that keys 0 and 2 are included in the set.

In the tasks below, your job is to implement the `IFindKeys` interface:

```
public interface IFindKeys {

    public int[] findKeys(int N, int k, ITreasureExtractor treasureExtractor);

}
```

The interface requires you to support the following method:

- `int[] findKeys(int N, int k, ITreasureExtractor treasureExtractor)`:
  Returns an array of length N of 0's and 1's (represented as integers), where a 1 at index i means that key i is in the set of correct keys (i.e. i unlocks one of the K locks). As per the description above, the array should have K 1's.
  For example, for $N = 5, K = 2$, `keys` $\{1, 0, 1, 0, 0\}$ means that keys 0 and 2 are in the set of correct keys.
  You are given `N`: the total number of keys, `k`: the total number of locks, and `treasureExtractor` as described above for you to call `tryUnlockChest`.

2.2 Testing

To test your implementation, you can run the `main` function in `RunAttempt.java`. If your implementation is correct (returns the correct keys bitmap), you should see in your console relevant information about the performance of your algorithm. Otherwise, you should see an Exception being thrown.

Feel free to change `bitmap` to experiment with different N and K.

3 Tasks

Problem 1: For any N keys and K locks, what is the minimum number of attempts that we need? Implement an algorithm in `FindKeysMinimumAttempts.java` that attempts to achieve this minimum number of attempts.

Bonus Optional: What is the asymptotic upper bound? Can you prove it?

Problem 2: Turns out there are natives on the island who charge $1 for each key the TreasureExtractor takes in on each attempt. What is the minimum amount that we need to spend? Implement an algorithm in `FindKeysLowestCost.java` that attempts to achieve the minimum cost required to identify all K correct keys.

4 Grading

For both problems, we will run your algorithm on different N and K and take note of
1) The number of successful cases (return correct keys bitmap) and
2) The total number of attempts sent or cost incurred for successful runs.