COMP26120 - January 2013 - Answers

Todd Davies

January 7, 2015

Please don't assume these answers are right. This is me attempting a past paper for revision purposes; I could have got it all wrong ;)

I chose to answer questions 1 & 2.

Question 1

Part a

```
public static boolean hasProductPair(int[] input, int k) {
  for(int i = 0; i < input.length; i++) {
    int divResult = k / input[i];
    if(BinarySearch.binarySearch(input, divResult, i + 1, input.length - 1) >= 0) {
      return true;
    }
  }
  return false;
}

public static void main(String[] args) {
```

Listing 1: for each iteration, and uses a binary search to see if the result is in the list. The binary search index increases by one on every iteration so that the square of items isn't counted.]Iterates through the list, divides k by input[i] for each iteration, and uses a binary search to see if the result is in the list. The binary search index increases by one on every iteration so that the square of items isn't counted.

Runs in $O(n \log n)$ (we do a binary search ($O(\log n)$) n times), but assumes that input is sorted first.

Part b

```java
public static boolean hasMajorityElement(int[] input) {
  HashMap<Integer, Integer> count = new HashMap<Integer, Integer>();
  final int desiredCount = input.length / 2;
  for(int i : input) {
    if(!count.containsKey(i)) count.put(i, 0);
    int c = count.get(i);
    if(++c == desiredCount) return true;
    else count.put(i, c);
  }
  return false;
}
```

Listing 2: Iterates through the list, keeps a count in a HashSet of how many times it's seen each integer. If the count equals $\frac{|input|}{2}$, then it returns true.

Runs in $O(n)$ time, providing the HashMap does lookups in $O(1)$ time, which for Java ints, it should.

Part c

Question 2

Part a