COMP3821 - Assignment 1

z5119666 Shan Wang

1. (a)

```
Algorithm Sort array by Merge-Sort and then search for sum = x by binary search
  Procedure Find Sum (S, x)
    Merge-Sort (S)
    for each element S[i] in array S
      Binary-Search (S, x-S[i])
      if Binary-Search (S, x-S[i]) is true then return true
    endfor
    return false
```

The complexity of Merge-Sort is O(nlogn) and binary search for every element in the array cost O(nlogn). Therefore, the total complexity is O(nlogn)

(b)

```
Algorithm Sort array by using hashtable and then search for sum
  Procedure Find Sum (S, x)
    for each element in S
      Make a hashtable to store the array S, key is the modulus of S[i] by x
    endfor
    for each element in hashtable
      Search for key directly in the hashtable
      if found then return true
    endfor
    return false
```

The complexity of inserting every element to hashtable is O(n), and search for every element in the hashtable cost O(n). Therefore, the total complexity is O(n)

2.

```
Algorithm Sort array by Merge-Sort and then search for #elements between L and R by binary search
  Procedure howManyElementsBetween (S)
    Merge-Sort (S)
    for n queries provide L and R
      Lp <- Binary-Search (S, L) get the position of L
      Rp <- Binary-Search (S, R) get the position of R
      howMany <- Rp - Lp
    endfor
```

The complexity of Merge-Sort is O(nlogn) and binary search for every query in the cost O(nlogn). Therefore, the total complexity is O(nlogn)

3.

```
Algorithm Cantor's diagonalization, for each friend there is 1 different choice
Procedure support Teams (N teams, N friends) // teams and friends in 2 arrays
  for i = 0; all team[i]; i++
    ask friend[i] does friend[i+1] support team[i+1], while for the last friend ask friend[0]'s preference
    if yes then I don't support team[i+1]
    else then I support team[i+1]
    save my choice for team[i+1]
  endfor
```

The complexity of it is O(n), and ask N questions.

4.

There are (n-1) difference between n numbers. Separate the interval into k 'buckets', so the absolute value of subtraction between buckets are (k-1). Therefore, there are (n-k) ($\Leftrightarrow$ (n-1) – (k-1)) difference within buckets. Within the buckets, the largest difference is (1/k), and between buckets, the largest difference is (2/k). Thus, the total is $(n-k)/k + 2(k-1)/k < 2$, which is $k > n-2$.

The algorithm of this problem is a not fully sorted bucket-sort. The algorithm is more like hashtable; the key is determined by the number of buckets. And put the numbers in the same range into one buckets. So this runs linear time.

5. (a)

```
Algorithm linear algorithm and ask 3 times to (n-1) people
  Procedure Find celebrity (n people)
    for n people and set i = 0
      if person[i] knows person[i+1]
      then
        person[i] is not celebrity
        potential celebrity = person[i+1]
      else
        person[i+1] is not celebrity
        potential celebrity = person[i]
    endfor
    for n people
      ask them if they knows potential celebrity
      if no then return no celebrity
    endfor
    for n people
      ask potential celebrity if he/she knows others
      if yes then return no celebrity
    endfor
    return potential celebrity
```

Therefore, this task can be accomplished by asking at most 3(n-1) questions.

(b)

In the elimination phase, we use tournament tree to get potential celebrity. And then when we checking if potential celebrity is true, we don't need to ask the same question to the people we already asked. Therefore, logn questions can be omit in the final check. (n -1) + 2n – logn -> 3n – floor(logn) – 2

6. (a)

Since every student write down their name on the paper, everyone's name appears at least once. So consider everyone votes for themselves and a classmate. Therefore, calculate both name on the paper, and minus one for everyone because that one is they vote for themselves.

(b)

Each paper has 2 names, if a student did not receive any votes, then his name will only show on his paper. Therefore, the student did not receive any votes is whose name only appears once. And the other name on that paper is the person he votes for.

(c)

Since there are n students and n votes. Everyone received at least one vote, which also means everyone receive exactly one vote.
i.e. n student * (≥ 1 votes) ⇔ there are ≥n votes in total. However, there are exactly n votes. Therefore, the maximum vote everyone can get is 1.

(d)

Algorithm Graph with direction
  Procedure vote for (n student, n paper)
    go through all paper, consider every paper as name pairs like <name1, name2>, and calculate how many votes each student received.
    sorted the votes by ascending order array
    for each student in the array
      if the student received 0 vote
      then he votes for the other name on that pair
      else he votes for the highest votes student
      delete 1 vote of the student he votes for
      next student is who remains the least votes
    endfor

The time complexity is O(n^2). From the student who received the least votes, if he received 0 vote, then there is only one possibility, else consider he vote for the

person who received more votes. Repeat, always consider from the student with the least votes, and check the pair with him/her.