Question 3:

First we sort the array in increasing order, we add the first interval [array[0],array[0]+unitLength], we remove all elements in that interval from the array, until array[i] > array[0] + unitLength, then we add the interval [array[i],array[i]+unitLength], we continue to do that until the array is empty.

Algorithm :

```
intervalSets(Array, unit) {
      Solution ← Empty
      Sort(Array)
      k← 0
      While ( Array is not Empty) {
            Solution ← Solution U { [array[k] , array[k]+unit] }
            i ← 0
            While (Array[i]  <= Array[k]+unit) {
                  Array.remove(i)
                  i++
            }
            k ← i
      }
      return Solution
}
```

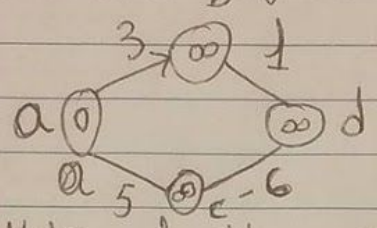
The greedy choice : The first interval of the solution starts with the smallest element in the array. Since there is no element less than array[0] (after sorting), that could be the the start of the first interval.

The optimal structure: we get the first interval, we remove all elements in that interval from that array, the next interval is determined by the remaining elements(i.e the first i < array.length S.T array[i] > array[0] + unit , array[i] is the start of the new interval and we keep doing that until the sorted array is empty.) -> optimal solution is the union of all the intervals then.

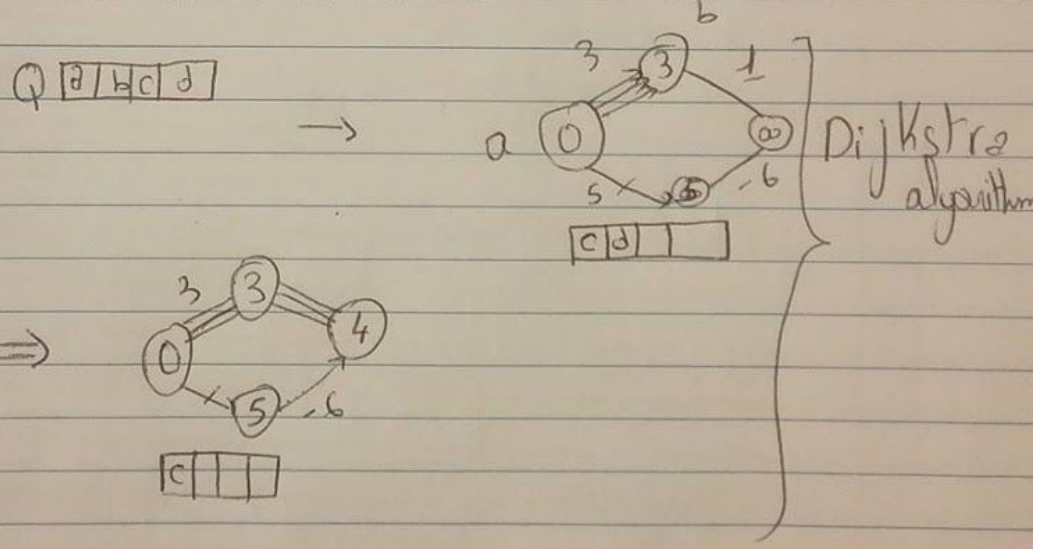
The worst case of running the algorithm is O (n*log(n)). The array sorting is O(n*log(n)). And removing the array element is O(n), even though there is two nested while loop, it will only go through the elements (n) of the sorted array once.

Question 4:

Let's consider the following graph:
find the shortest from a to d



the Dijkstra algorithm will not produce the shortest path which is in this case; $a \to c \to d$.
instead it will return $a \to b \to d$



Dijkstra algorithm is a greedy algorithm, it always seeks the lightest edge —, it is only valid with non-negative weights.

Question 5:

A graph is bipartite → contains no odd cycles. (first direction)

Say that the graph is bipartite and have two vertices Sets A and B.

We assume it contains an odd cycle (v1,v2,...,vn,v1) then v1 is in A, v2 in B .. etc → odd indexes are in A and even indexes are in B or n is odd because the cycle is odd. Then vn is in A but V1 is in A and that is a contradiction.

A graph contains no add cycles → A graph is bipartite. (second direction)

A graph doesn't contain any odd cycles. Suppose a vertex v is in the graph. We can divide the graph into :

- Let A be the set of vertices such that the shortest path from each element of A to v is of odd length.
- Let B be the set of vertices such that the shortest path from each element of B to v is of even length.

(v is in B in this case.)

Now suppose there exists two vertices a1 and a2 in A and are adjacent.

Then the cycle (v,...,a1,a2,....v) is odd: contradiction. (Same for set B).

So no two vertices in A can be adjacent.

Hence the graph is bipartite.

For the second part help from :
https://proofwiki.org/wiki/Graph_is_Bipartite_iff_No_Odd_Cycles