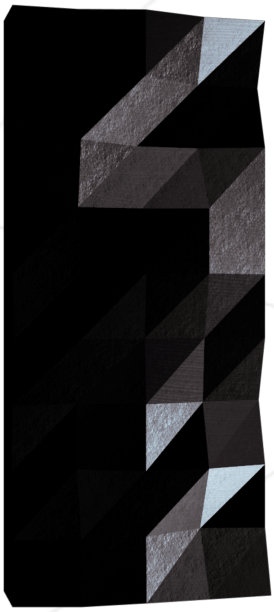

Starfleet

Interview

Staff 42 pedago@42.fr

Summary: This document is an interview question for the Starfleet Piscine.

Contents

I General rules **2**
I.1 During the interview 3

II Subset sequence **4**
II.1 Interview question 4
II.2 Acceptable answers (no constraint) 4
II.2.1 Sort the array 4
II.3 Follow up question 5
II.3.1 Hints 5
II.4 Best solution 6
II.4.1 Solution with hash table 6

Chapter I

General rules

- The interview should last between `45 minutes`.
- Both the interviewer and the interviewed student must be present.
- The interviewed student should write his code using a `whiteboard`, with the language of her/his choice.
- At the end of the interview, the interviewer evaluates the student based on the provided criteria.

Read carefully the interview question and solutions, and make sure you `understand` them before the interview. You can't share this document with other students, as they might be interviewed on the same question. Giving them the answer would prevent them from having to solve an unknown question during an interview.

I.1 During the interview

During the interview, we ask you to :

- Make sure the interviewed student `understands` the question.
- Give her/him any `clarification` on the subject that she/he might need.
- Let her/him come up with a solution before you guide her/him to the best solution given the constraints (time and space).
- Ask the student what is the `complexity` of her/his algorithm ? Can it be improved and how ?
- `Guide` her/him to the best solution without giving the answer. You may refer to the `hints` for that.
- You want to evaluate how the interviewed student thinks, so ask her/him to `explain everything` that she/he thinks or writes (there should be no silences).
- If you see a mistake in the code, wait untill the end and give her/him a chance to correct it by her/himself.
- Ask the student to show how the algorithm works on an `example`.
- Ask the student to explain how `limit cases` are handled.
- Bring out to the student any mistake she/he might have done.
- Give `feedback` on her/his performances after the interview.
- Be `fair` in your evaluation.

As always, stay mannerly, polite, respectful and constructive during the interview. If the interview is carried out smoothly, you will both benefit from it !

Chapter II

Subset sequence

II.1 Interview question

`Given an array integers which might contain duplicates, find the largest subset which form a sequence.`

EXAMPLE :

```
Input : {1,6,10,4,7,9,5}
Output : 4 5 6 7
```

II.2 Acceptable answers (no constraint)

II.2.1 Sort the array

- Sort the array (which can be done in-place in O(nlogn) time).
- Go through the sorted array and search for the longest sequence of successive numbers. This is done in O(n) time, but since you have to sort the array first, the algorithm is still in O(nlogn).

```
O(nlogn) time , O(1) space
```

Code:

```
void largestSequence(int *arr, int n) {
        int start, count;
        int current_start, current_count;

        // sorting takes O(nlogn) time
        sort(arr);

        // find the longest sequence O(n) time
        start = 0;
        count = 0;
        current_start = 0;
        current_count = 0;
        for (int i = 1; i <= n; i++) {
                if (i < n && arr[i - 1] == arr[i]) {
                        current_count++;
                } else {
                        if (current_count > count) {
                                start = current_start;
                                count = current_count;
                        }
                        current_start = 0;
                        current_count = 0;
                }
        }

        // print the sequence
        for (int i = 0; i < count; i++) {
                printf("%d ", start + i);
        }
        printf("\n");
}
```

II.3 Follow up question

Provide a solution with best possible runtime O(n). You may use extra space.

II.3.1 Hints

- What not use a hash table?
- Use the hash table to keep record of sequences boundaries.

II.4 Best solution

II.4.1 Solution with hash table

- Create a hash table.
- For each element in the array :

1. If the element has already been processed, continue.

2. Otherwise, add the new element to the hash table.

3. Check if element-1 and element+1 exist in the hash table. If one of them exists, the new element will be a new boundary for the sequence of consecutive number. If both of them exist, it will join the 2 sequences.

4. In both cases, update the hash table with higher bound to the lowest key and lower bound to highest key. That way, hashTable will always give the lower bound of the sequence right below i and hashTable will always give the upper bound of the sequence right above i (if they exist).

5. If the new found sequence is the longest yet, keep a record of it by updating the first and last indexes.

```
O(n) time , O(n) space
```

```
EXAMPLE : {1,6,10,4,7,9,5}

i = 0, arr[0] = 1,  hashTable = {1=1}, first = 0, last = 0

i = 1, arr[1] = 6,  hashTable = {1=1, 6=6}, first = 0, last = 0

i = 2, arr[2] = 10, hashTable = {1=1, 6=6, 10=10}, first = 0, last = 0

i = 3, arr[3] = 4,  hashTable = {1=1, 4=4, 6=6, 10=10}, first = 0, last = 0

i = 4, arr[4] = 7,  hashTable = {1=1, 4=4, 6=7, 7=6, 10=10}, first = 6, last = 7

i = 5, arr[5] = 9,  hashTable = {1=1, 4=4, 6=7, 7=6, 9=10, 10=9}, first = 6, last = 7

i = 6, arr[6] = 5,  hashTable = {1=1, 4=7, 5=5, 6=7, 7=4, 9=10, 10=9}, first = 4, last = 7
```

Code:

```
void largestSequence(int *arr, int n) {

        hashTable<int,int> hashTable; // declare a new hash table
        int first, last;
        int start, end;

        first = 0;
        last = 0;
        for (int i = 0; i < n; i++) {

                if (arr[i] in hashTable) // if key arr[i] exists in hash table
                        continue ;

                hashTable[arr[i]] = arr[i]; // set (key, value) in hash table

                start = i;
                end = i;
                if (arr[i]-1 in hashTable)
                        start = hashTable[arr[i] - 1];
                if (arr[i]+1 in hashTable)
                        end = hashTable[arr[i] + 1];

                // update the hash table
                hashTable[start] = end;
                hashTable[end] = start;

                // update first and last
                if (end - start > last - fist) {
                        first = start;
                        last = end;
                }
        }

        // print sequence
        for (int i = first; i <= last; i++) {
                printf("%d ", i);
        }
        printf("\n");
}
```