# Medians - solution

The algorithm presented below assumes that there exists a solution. Checking if a solution actually exists can be added pretty easily to the algorithm.

We will construct the permutation *A[1], …, A[2*N-1]* incrementally, in a greedy manner. Let *vmin* and *vmax* be two variables, initially set to *0* and *2*N*. Let *used* be a binary array with *2*N+1* elements, which is all set to *0*, except for *used[0]* and *used[2*N]*, which are set to *1*.

The function *updateVmin()* proceeds as follows: as long as *used[vmin]=1*, it increments *vmin*. The function *updateVmax()* proceeds as follows: as long as *used[vmax]=1*, it decrements *vmax*.

The overall algorithm proceeds as follows. First, we have *A[1]=B[1]* (and then we set *used[A[1]]=1*). Then:

```
for i=2 to N do {
        if (B[i] is equal to B[i-1]) then {
                // add to the permutation the smallest and the largest unused elements.
                updateVmin()
                A[2*i-2]=vmin; used[vmin]=1
                updateVmax()
                A[2*i-1]=vmax; used[vmax]=1
        }

        if (B[i]>B[i-1]) {
                if (used[B[i]] is equal to 0) {
                        // B[i] has not been added to the permutation, yet.
                        // Add B[i] to the permutation and the largest unused element.
                        A[2*i-2]=B[i]; used[B[i]]=1
                        updateVmax()
                        A[2*i-1]=vmax; used[vmax]=1
                } else {
                        // B[i] already exists in the permutation.
                        // Add the largest two unused elements.
                        updateVmax()
                        A[2*i-2]=vmax; used[vmax]=1
                        updateVmax()
                        A[2*i-1]=vmax; used[vmax]=1
                }
        }
```

```
        if (B[i]<B[i-1]) {
                if (used[B[i]] is equal to 0) {
                        // B[i] has not been added to the permutation, yet.
                        // Add B[i] to the permutation and the smallest unused element.
                        A[2*i-2]=B[i]; used[B[i]]=1
                        updateVmin()
                        A[2*i-1]=vmin; used[vmin]=1
                } else {
                        // B[i] already exists in the permutation.
                        // Add the smallest two unused elements.
                        updateVmin()
                        A[2*i-2]=vmin; used[vmin]=1
                        updateVmin()
                        A[2*i-1]=vmin; used[vmin]=1
                }
        }
}
```

The time complexity of the algorithm is ***O(N)***.

***Proof of correctness:***

We first define *lower(i)=i-1* and *upper(i)=2*N+1-i*. Note that, when a solution exists, we always have *B[i]>lower(i)* and *B[i]<upper(i)* (that means none of the numbers *1, …, i-1* or *2*N+1-i, …, 2*N-1* can be the $i^{th}$ median or any median after the $i^{th}$). We will prove that whenever we add *vmin* to the permutation, we have *vmin<=lower(i)* and whenever we add *vmax* to the permutation, we have *vmax>=upper(i)*. If that is the case, then it is obvious that our algorithm finds a correct solution. That's because adding *vmin* or *vmax* to the permutation will not interfere in any way with the values of the medians after the current index *i*.

A solution does not exist when *B[i]* does not obey the inequalities regarding *lower(i)* and *upper(i)* or when *B[i]* and *B[i-1]* are not adjacent in the sorted order of all the first *2*i-1* elements of the permutation. Note that, if our claim regarding *vmin* and *vmax* is true, then our algorithm cannot cause any of the previous conditions to occur (unless the *B[i]* values do not allow for any solution).

We will consider several cases (for *2<=i<=N*). We will always assume that we have *lower(i)<B[i]<upper(i)* (otherwise, we know from the start that there is no solution).

**Case 1: B[i]=B[i-1]**

We need to add *vmin* and *vmax* to the permutation.

Let's assume that all the numbers *1, …, lower(i)* are already in the permutation before considering the $i^{th}$ median (and thus, we will have *vmin>lower(i)*). In this case, the median *B[i-1]* of the first *2*(i-1)-1* elements of the permutation must be exactly *i-1* (because there would be exactly *i-2* elements smaller than it in the permutation). However, *B[i]* cannot be equal to *i-1*, because *i-1=lower(i)*. Thus, we cannot have *B[i]=B[i-1]*. This contradicts our initial assumptions, leading us to the conclusion that at least one of the numbers *1, …, lower(i)* was not used among the first 2*(i-1)-1 elements of the permutation. *vmin* will be equal to one of the numbers not used which are at most equal to *lower(i)* and, thus, we will have *vmin<=lower(i)*.

The proof is similar for *vmax* (actually, it is symmetrical).

**Case 2: B[i]<B[i-1]**

**Subcase 2.1:** *B[i]* does not appear as a median before the index *i*.

We assume our claim to be correct for the first *i-1* medians and we will prove it to be correct for the first *i* medians. This means that neither *vmin* or *vmax* were ever equal to *B[i]*. Thus, *B[i]* does not exist in the permutation so far and we can add it. As for *vmin* (which must also be added to the permutation), we will show that *vmin<=lower(i)*.

As before, let's assume that all the numbers *1,…,lower(i)* occur among the first *2*(i-1)-1* elements of the permutation. But then we must have *B[i-1]=i-1*. And, since *B[i]>lower(i)*, we would have *B[i]>B[i-1]* (but this contradicts our initial assumption!).

**Subcase 2.2:** *B[i]* appeared as a median at some previous index (possibly more than once).

As in the previous subcase, we assume that our claim is correct for the first *i-1* medians and then we will prove that it is also correct for the first *i* medians.

We will need to add *vmin* to the permutation two times. Thus, we have to prove that at least two elements among the set *1, …, lower(i)* have not been used, yet. Let's assume first that all the elements *1,…,lower(i)* were used among the first *2*(i-1)-1* elements of the permutation. This case is handled as in the previous subcase (the obtained contradiction is, again, that *B[i]>B[i-1]*).

Let's assume now that all the elements *1, …, lower(i)* have been used among the first *2*(i-1)-1* elements of the permutation, except one (whose value we denote by *X*). In this case, we have *B[i-1]>lower(i)=i-1*. Since *B[i]* appeared before as a median, it must a value adjacent to *B[i-1]* (in the sorted order of all the first *2*(i-1)-1* elements of the permutation). Note how the value just before *B[i-1]* in the sorted order is exactly *lower(i)=i-1* (because *lower(i)* is the $(i\text{-}2)^{nd}$ elements in the sorted order). Thus, we obtain *B[i]=lower(i)*. But this contradicts our general assumption that *B[i]>lower(i)*.

**Case 3: B[i]>B[i-1].**

This case is handled similarly to *Case 2*, but in a symmetrical manner (it also has two subcases).