How to sort a hash of hashes by value?

hash | keys | values | sort | cmp | <=> | $a | $b

Before attempting to sort a hash of hashes by values, one should make sure to be familiar with the question: How to sort a hash in Perl?

The problem

Given the following reference to a hash of hashes, how can we sort them based on the value of **Position**?

```
my $data = {
  'Gaur 3' => {
        'Max' => '85',
        'Type' => 'text',
        'Position' => '10',
        'IsAdditional' => 'Y',
        'Required' => 'Y',
        'Mandatory' => 'Y',
        'Min' => '40'
  },
  'Gaur 2' => {
        'Max' => '90',
        'Type' => 'text',
        'Position' => '11',
        'IsAdditional' => 'Y',
        'Required' => 'Y',
        'Mandatory' => 'Y',
        'Min' => '60'
  },
  'Gaur 1' => {
        'Max' => '80',
        'Type' => 'text',
        'Position' => '10',
        'IsAdditional' => 'Y',
        'Required' => 'Y',
        'Mandatory' => 'Y',
```

```
        'Min' => '40'
    },
};
```

Of course, we cannot really sort the hash, but we can sort the keys. As `$data` is a reference to a hash, first we need to de-reference it by putting a `%` in front of it. calling `keys` on this construct will return the list of keys in a random order.

```
1. my @keys = keys %$data;
   foreach my $k (@keys) {
       say $k;
   }
```

Result might look like this, but it can be in a different order on other runs:

```
Gaur 1
Gaur 3
Gaur 2
```

`$a` and `$b` are the standard place-holders of `sort`. For each comparison they will hold two keys from the list. In order to compare the **Position** value of two elements we need to access them using `$a` and `$b` and compare the numerical(!) values using the spaceship-operator (`<=>`).

```
1. my @positioned = sort { $data->{$a}{Position} <=> $data->{$b}{Position} }  ke

   foreach my $k (@positioned) {
       say $k;
5. }
```

The result is:

```
Gaur 3
Gaur 1
Gaur 2
```

This would be nice, but unfortunately this result is not going to be the same always. After all the **Position** values of 'Gaur 1' and 'Gaur 3' are both the same number (10), so the space-ship operator cannot really decide between the two.

Secondary sorting

If we would like to make sure the result of sort is more predictable, we will need to sort based on two (or even more) values.

For example we can sort the hash first by the **Position** value, and among the entries with the same Position value we can sort by the value of the **Max** field. In order to do this we will use the following expression:

```
my @maxed = sort {
    $data->{$a}{Position} <=> $data->{$b}{Position}
    or
    $data->{$a}{Max} <=> $data->{$b}{Max}
    }  keys %$data;

foreach my $k (@maxed) {
    say $k;
}
```

And the result is:

```
Gaur 1
Gaur 3
Gaur 2
```

Of course if, we encounter two entries where both the **Position** and the **Max** fields have the exact same value, then even this sort won't provide the same results on every call.