Key Indexed Counting

Sonam Wangmo

Gyalpozhing College of Information Technology

October 14, 2020

Key Indexed Counting

- A simple method for sorting that is effective whenever the keys are small integers.
- There are mainly four steps in processing key-indexed counting sort.
  1. Compute frequency counts.
  2. Transform counts to indices.
  3. Distribute the data.
  4. Copy back.

Example: Inputs are given below

Table 1: array a[]

Anderson	2
Brown	3
Davis	3
Garcia	4
Harris	1
Jackson	3
Johnson	4
Jones	3
Martin	1
Martinez	2
Miller	2
Moore	1
Robinson	2
Smith	4
Taylor	3

1. Compute frequency counts

The first step is to count the frequency of occurrence of each key value, using an int array count[]. For each item, we use the key to access an entry in count[] and increment that entry. If the key value is r, we increment count[r+1].

```

        for (i = 0; i < N; i++)
        count[a[i].key() + 1]++;
```

2.Transform counts to indices

Next,we use count[] to compute, for each key value, the starting index positions in the sorted order of items with that key.

```

        for (int r = 0; r < R; r++)
        count[r+1] += count[r];
```

3. Distribute the data

With the count[] array transformed into an index table, we accomplish the actual sort by moving the items to an auxiliary array aux[].

```

        for (int i = 0; i < N; i++)
        aux[count[a[i].key()]++] = a[i]
```

4. Copy back

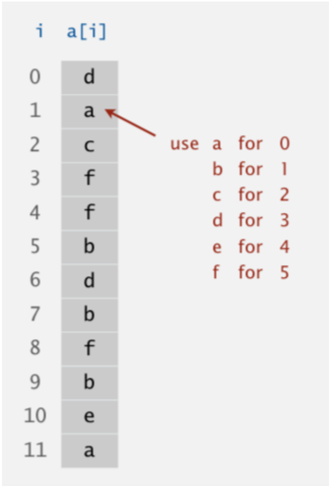
Since we accomplished the sort by moving the items to an auxiliary array, the last step is to copy the sorted result back to the original array.

```
        for (int i = 0; i < N; i++)
        a[i] = aux[i]
```

Key-indexed counting (a[].key is an int in [0, R).

```
int N = a.length;
String[] aux = new String[N];
int[] count = new int[R+1];
// Compute frequency counts.
for (int i = 0; i < N; i++)
count[a[i].key() + 1]++;
// Transform counts to indices.
for (int r = 0; r < R; r++)
count[r+1] += count[r];
// Distribute the records.
for (int i = 0; i < N; i++)
aux[count[a[i].key()]++] = a[i];
// Copy back.
for (int i = 0; i < N; i++)
a[i] = aux[i];
```

Q. Do key indexing counting sort for the given array.



i	a[i]
0	d
1	a
2	c
3	f
4	f
5	b
6	d
7	b
8	f
9	b
10	e
11	a

use a for 0
b for 1
c for 2
d for 3
e for 4
f for 5

Figure 1: a[i]

Running Time of Key Indexed Counting

- key-indexed counting does no compares (it accesses data only through key()). When R is within a constant factor of N, we have a linear-time sort.
- O(N+ R) R can be characters of alphabets, Hence we ignore R.
- O(N)