



<http://algs4.cs.princeton.edu>

5.1 KEY-INDEXED COUNTING DEMO

Key-indexed counting demo

Goal. Sort an array $a[]$ of N integers between 0 and $R - 1$.

- Count frequencies of each letter using key as index.
- Compute frequency cumulates which specify destinations.
- Access cumulates using key as index to move items.
- Copy back into original array.

$R = 6$

```
int N = a.length;
int[] count = new int[R+1];

for (int i = 0; i < N; i++)
    count[a[i]+1]++;

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

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

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

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

Key-indexed counting demo

Goal. Sort an array $a[]$ of N integers between 0 and $R - 1$.

- Count frequencies of each letter using key as index.
- Compute frequency cumulates which specify destinations.
- Access cumulates using key as index to move items.
- Copy back into original array.

```
int N = a.length;
int[] count = new int[R+1];

for (int i = 0; i < N; i++)
    count[a[i]+1]++;

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

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

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

count
frequencies



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

r	count[r]
a	0
b	2
c	3
d	1
e	2
f	1
-	3

Key-indexed counting demo

Goal. Sort an array $a[]$ of N integers between 0 and $R - 1$.

- Count frequencies of each letter using key as index.
- Compute frequency cumulates which specify destinations.
- Access cumulates using key as index to move items.
- Copy back into original array.

```
int N = a.length;
int[] count = new int[R+1];

for (int i = 0; i < N; i++)
    count[a[i]+1]++;

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

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

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

compute
cumulates



i	a[i]	r	count[r]
0	d		
1	a		
2	c		
3	f	a	0
4	f	b	2
5	b	c	5
6	d	d	6
7	b	e	8
8	f	f	9
9	b	-	12
10	e		
11	a		

6 keys < d, 8 keys < e
so d's go in a[6] and a[7]

Key-indexed counting demo

Goal. Sort an array $a[]$ of N integers between 0 and $R - 1$.

- Count frequencies of each letter using key as index.
- Compute frequency cumulates which specify destinations.
- Access cumulates using key as index to move items.
- Copy back into original array.

```
int N = a.length;
int[] count = new int[R+1];
```

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

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

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

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

move
items



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

r	$count[r]$
a	0
b	2
c	5
d	6
e	8
f	9
-	12

Key-indexed counting demo

Goal. Sort an array $a[]$ of N integers between 0 and $R - 1$.

- Count frequencies of each letter using key as index.
- Compute frequency cumulates which specify destinations.
- Access cumulates using key as index to move items.
- Copy back into original array.

```
int N = a.length;
int[] count = new int[R+1];
```

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

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

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

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

move
items



i	$a[i]$	r	$count[r]$	i	$aux[i]$
0	d			0	
1	a			1	
2	c			2	
3	f	a	0	3	
4	f	b	2	4	
5	b	c	5	5	
6	d	d	7	6	d
7	b	e	8	7	
8	f	f	9	8	
9	b	-	12	9	
10	e			10	
11	a			11	

Key-indexed counting demo

Goal. Sort an array $a[]$ of N integers between 0 and $R - 1$.

- Count frequencies of each letter using key as index.
- Compute frequency cumulates which specify destinations.
- Access cumulates using key as index to move items.
- Copy back into original array.

```
int N = a.length;
int[] count = new int[R+1];
```

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

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

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

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

move
items



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

r	$count[r]$
a	1
b	2
c	5
d	7
e	8
f	9
-	12

Key-indexed counting demo

Goal. Sort an array $a[]$ of N integers between 0 and $R - 1$.

- Count frequencies of each letter using key as index.
- Compute frequency cumulates which specify destinations.
- Access cumulates using key as index to move items.
- Copy back into original array.

```
int N = a.length;
int[] count = new int[R+1];
```

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

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

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

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

move
items



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

r	$count[r]$
a	1
b	2
c	6
d	7
e	8
f	9
-	12

Key-indexed counting demo

Goal. Sort an array $a[]$ of N integers between 0 and $R - 1$.

- Count frequencies of each letter using key as index.
- Compute frequency cumulates which specify destinations.
- Access cumulates using key as index to move items.
- Copy back into original array.

```
int N = a.length;
int[] count = new int[R+1];

for (int i = 0; i < N; i++)
    count[a[i]+1]++;

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

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

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

move
items



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

r	$count[r]$
a	1
b	2
c	6
d	7
e	8
f	10
-	12

Key-indexed counting demo

Goal. Sort an array $a[]$ of N integers between 0 and $R - 1$.

- Count frequencies of each letter using key as index.
- Compute frequency cumulates which specify destinations.
- Access cumulates using key as index to move items.
- Copy back into original array.

```
int N = a.length;
int[] count = new int[R+1];

for (int i = 0; i < N; i++)
    count[a[i]+1]++;

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

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

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

move
items



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

r	$count[r]$
a	1
b	2
c	6
d	7
e	8
f	11
-	12

Key-indexed counting demo

Goal. Sort an array $a[]$ of N integers between 0 and $R - 1$.

- Count frequencies of each letter using key as index.
- Compute frequency cumulates which specify destinations.
- Access cumulates using key as index to move items.
- Copy back into original array.

```
int N = a.length;
int[] count = new int[R+1];
```

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

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

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

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

move
items



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

r count[r]

a	1
b	3
c	6
d	7
e	8
f	11
-	12

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

Key-indexed counting demo

Goal. Sort an array $a[]$ of N integers between 0 and $R - 1$.

- Count frequencies of each letter using key as index.
- Compute frequency cumulates which specify destinations.
- Access cumulates using key as index to move items.
- Copy back into original array.

```
int N = a.length;
int[] count = new int[R+1];
```

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

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

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

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

move
items



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

r count[r]

a	1
b	3
c	6
d	8
e	8
f	11
-	12

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

Key-indexed counting demo

Goal. Sort an array $a[]$ of N integers between 0 and $R - 1$.

- Count frequencies of each letter using key as index.
- Compute frequency cumulates which specify destinations.
- Access cumulates using key as index to move items.
- Copy back into original array.

```
int N = a.length;
int[] count = new int[R+1];
```

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

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

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

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

move
items



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

r count[r]

a	1
b	4
c	6
d	8
e	8
f	11
-	12

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

Key-indexed counting demo

Goal. Sort an array $a[]$ of N integers between 0 and $R - 1$.

- Count frequencies of each letter using key as index.
- Compute frequency cumulates which specify destinations.
- Access cumulates using key as index to move items.
- Copy back into original array.

```
int N = a.length;
int[] count = new int[R+1];
```

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

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

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

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

move
items



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

r count[r]

a	1
b	4
c	6
d	8
e	8
f	12
-	12

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

Key-indexed counting demo

Goal. Sort an array $a[]$ of N integers between 0 and $R - 1$.

- Count frequencies of each letter using key as index.
- Compute frequency cumulates which specify destinations.
- Access cumulates using key as index to move items.
- Copy back into original array.

```
int N = a.length;
int[] count = new int[R+1];
```

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

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

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

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

move
items



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

r	$count[r]$
a	1
b	5
c	6
d	8
e	8
f	12
-	12

Key-indexed counting demo

Goal. Sort an array $a[]$ of N integers between 0 and $R - 1$.

- Count frequencies of each letter using key as index.
- Compute frequency cumulates which specify destinations.
- Access cumulates using key as index to move items.
- Copy back into original array.

```
int N = a.length;
int[] count = new int[R+1];
```

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

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

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

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

move
items



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

r count[r]

a	1
b	5
c	6
d	8
e	9
f	12
-	12

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

Key-indexed counting demo

Goal. Sort an array $a[]$ of N integers between 0 and $R - 1$.

- Count frequencies of each letter using key as index.
- Compute frequency cumulates which specify destinations.
- Access cumulates using key as index to move items.
- Copy back into original array.

```
int N = a.length;
int[] count = new int[R+1];

for (int i = 0; i < N; i++)
    count[a[i]+1]++;

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

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

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

move
items



i	a[i]	r	count[r]	i	aux[i]
0	d			0	a
1	a			1	a
2	c			2	b
3	f	a	2	3	b
4	f	b	5	4	b
5	b	c	6	5	c
6	d	d	8	6	d
7	b	e	9	7	d
8	f	f	12	8	e
9	b	-	12	9	f
10	e			10	f
11	a			11	f

Key-indexed counting demo

Goal. Sort an array $a[]$ of N integers between 0 and $R - 1$.

- Count frequencies of each letter using key as index.
- Compute frequency cumulates which specify destinations.
- Access cumulates using key as index to move items.
- Copy back into original array.

```
int N = a.length;
int[] count = new int[R+1];
```

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

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

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

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

move
items



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

r count[r]

a	2
b	5
c	6
d	8
e	9
f	12
-	12

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

Key-indexed counting demo

Goal. Sort an array $a[]$ of N integers between 0 and $R - 1$.

- Count frequencies of each letter using key as index.
- Compute frequency cumulates which specify destinations.
- Access cumulates using key as index to move items.
- Copy back into original array.

```
int N = a.length;
int[] count = new int[R+1];

for (int i = 0; i < N; i++)
    count[a[i]+1]++;

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

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

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

copy
back



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

r	$count[r]$
a	2
b	5
c	6
d	8
e	9
f	12
-	12