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.

```java
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];
```

### Example:

<table>
<thead>
<tr>
<th>i</th>
<th>a[i]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>d</td>
</tr>
<tr>
<td>1</td>
<td>a</td>
</tr>
<tr>
<td>2</td>
<td>c</td>
</tr>
<tr>
<td>3</td>
<td>f</td>
</tr>
<tr>
<td>4</td>
<td>f</td>
</tr>
<tr>
<td>5</td>
<td>b</td>
</tr>
<tr>
<td>6</td>
<td>d</td>
</tr>
<tr>
<td>7</td>
<td>b</td>
</tr>
<tr>
<td>8</td>
<td>f</td>
</tr>
<tr>
<td>9</td>
<td>b</td>
</tr>
<tr>
<td>10</td>
<td>e</td>
</tr>
<tr>
<td>11</td>
<td>a</td>
</tr>
</tbody>
</table>

- **RP**:
  - 0 for `d`
  - 1 for `a`
  - 2 for `c`
  - 3 for `f`
  - 4 for `b`
  - 5 for `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.

```java
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];
```

<table>
<thead>
<tr>
<th>i</th>
<th>a[i]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>d</td>
</tr>
<tr>
<td>1</td>
<td>a</td>
</tr>
<tr>
<td>2</td>
<td>c</td>
</tr>
<tr>
<td>3</td>
<td>f</td>
</tr>
<tr>
<td>4</td>
<td>f</td>
</tr>
<tr>
<td>5</td>
<td>b</td>
</tr>
<tr>
<td>6</td>
<td>d</td>
</tr>
<tr>
<td>7</td>
<td>b</td>
</tr>
<tr>
<td>8</td>
<td>f</td>
</tr>
<tr>
<td>9</td>
<td>b</td>
</tr>
<tr>
<td>10</td>
<td>e</td>
</tr>
<tr>
<td>11</td>
<td>a</td>
</tr>
</tbody>
</table>

offset by 1
[stay tuned]
**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.

```java
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];
```

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.

```java
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];
```

<table>
<thead>
<tr>
<th></th>
<th>i</th>
<th>a[i]</th>
<th></th>
<th>i</th>
<th>aux[i]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>d</td>
<td></td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>a</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>c</td>
<td></td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>f</td>
<td></td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>f</td>
<td></td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>b</td>
<td></td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>d</td>
<td></td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>b</td>
<td></td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>f</td>
<td></td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>b</td>
<td></td>
<td>9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>e</td>
<td></td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>a</td>
<td></td>
<td>11</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

```java
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];
```
**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];
```
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.

```java
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];
```
**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.

```java
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];
```

<table>
<thead>
<tr>
<th>( i )</th>
<th>( a[i] )</th>
<th>( \text{aux}[i] )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>d</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>a</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>c</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>f</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>f</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>b</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>d</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>b</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>f</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>b</td>
<td>9</td>
</tr>
<tr>
<td>10</td>
<td>e</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>a</td>
<td></td>
</tr>
</tbody>
</table>

Move items

### Frequency Cumulates

- \( a \rightarrow 1 \)
- \( b \rightarrow 2 \)
- \( c \rightarrow 6 \)
- \( d \rightarrow 7 \)
- \( e \rightarrow 8 \)
- \( f \rightarrow 10 \)
- \( g \rightarrow 12 \)
**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.

```java
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];
```

<table>
<thead>
<tr>
<th>i</th>
<th>a[i]</th>
<th>i</th>
<th>aux[i]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>d</td>
<td>0</td>
<td>a</td>
</tr>
<tr>
<td>1</td>
<td>a</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>c</td>
<td>2</td>
<td>c</td>
</tr>
<tr>
<td>3</td>
<td>f</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>f</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>b</td>
<td>5</td>
<td>c</td>
</tr>
<tr>
<td>6</td>
<td>d</td>
<td>6</td>
<td>d</td>
</tr>
<tr>
<td>7</td>
<td>b</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>f</td>
<td>8</td>
<td>f</td>
</tr>
<tr>
<td>9</td>
<td>b</td>
<td>9</td>
<td>f</td>
</tr>
<tr>
<td>10</td>
<td>e</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>a</td>
<td>11</td>
<td></td>
</tr>
</tbody>
</table>

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.

```java
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];
```

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.

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

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

r count[r]
a 1
b 3
c 6
d 8
e 8
f 11
g 12

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];
```

<table>
<thead>
<tr>
<th>i</th>
<th>aux[i]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>a</td>
</tr>
<tr>
<td>1</td>
<td>b</td>
</tr>
<tr>
<td>2</td>
<td>c</td>
</tr>
<tr>
<td>3</td>
<td>d</td>
</tr>
<tr>
<td>4</td>
<td>e</td>
</tr>
<tr>
<td>5</td>
<td>f</td>
</tr>
<tr>
<td>6</td>
<td>g</td>
</tr>
<tr>
<td>7</td>
<td>h</td>
</tr>
<tr>
<td>8</td>
<td>i</td>
</tr>
<tr>
<td>9</td>
<td>j</td>
</tr>
<tr>
<td>10</td>
<td>k</td>
</tr>
<tr>
<td>11</td>
<td>l</td>
</tr>
<tr>
<td>12</td>
<td>m</td>
</tr>
</tbody>
</table>
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.

```java
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
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.

```java
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

<table>
<thead>
<tr>
<th>i</th>
<th>a[i]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>d</td>
</tr>
<tr>
<td>1</td>
<td>a</td>
</tr>
<tr>
<td>2</td>
<td>c</td>
</tr>
<tr>
<td>3</td>
<td>f</td>
</tr>
<tr>
<td>4</td>
<td>f</td>
</tr>
<tr>
<td>5</td>
<td>b</td>
</tr>
<tr>
<td>6</td>
<td>d</td>
</tr>
<tr>
<td>7</td>
<td>b</td>
</tr>
<tr>
<td>8</td>
<td>f</td>
</tr>
<tr>
<td>9</td>
<td>b</td>
</tr>
<tr>
<td>10</td>
<td>e</td>
</tr>
<tr>
<td>11</td>
<td>a</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>i</th>
<th>aux[i]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>a</td>
</tr>
<tr>
<td>1</td>
<td>b</td>
</tr>
<tr>
<td>2</td>
<td>b</td>
</tr>
<tr>
<td>3</td>
<td>b</td>
</tr>
<tr>
<td>4</td>
<td>a</td>
</tr>
<tr>
<td>5</td>
<td>c</td>
</tr>
<tr>
<td>6</td>
<td>d</td>
</tr>
<tr>
<td>7</td>
<td>d</td>
</tr>
<tr>
<td>8</td>
<td>f</td>
</tr>
<tr>
<td>9</td>
<td>f</td>
</tr>
<tr>
<td>10</td>
<td>f</td>
</tr>
<tr>
<td>11</td>
<td>f</td>
</tr>
</tbody>
</table>
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.

```java
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];
```

<table>
<thead>
<tr>
<th>i</th>
<th>a[i]</th>
<th>i</th>
<th>aux[i]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>d</td>
<td>0</td>
<td>a</td>
</tr>
<tr>
<td>1</td>
<td>a</td>
<td>1</td>
<td>a</td>
</tr>
<tr>
<td>2</td>
<td>c</td>
<td>2</td>
<td>b</td>
</tr>
<tr>
<td>3</td>
<td>f</td>
<td>3</td>
<td>b</td>
</tr>
<tr>
<td>4</td>
<td>f</td>
<td>4</td>
<td>b</td>
</tr>
<tr>
<td>5</td>
<td>b</td>
<td>5</td>
<td>c</td>
</tr>
<tr>
<td>6</td>
<td>d</td>
<td>6</td>
<td>d</td>
</tr>
<tr>
<td>7</td>
<td>b</td>
<td>7</td>
<td>d</td>
</tr>
<tr>
<td>8</td>
<td>f</td>
<td>8</td>
<td>f</td>
</tr>
<tr>
<td>9</td>
<td>b</td>
<td>9</td>
<td>f</td>
</tr>
<tr>
<td>10</td>
<td>e</td>
<td>10</td>
<td>f</td>
</tr>
<tr>
<td>11</td>
<td>a</td>
<td>11</td>
<td>f</td>
</tr>
</tbody>
</table>

move items

```
**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.

```java
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];
```
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.

```java
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];
```
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.

```java
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];
```

---

<table>
<thead>
<tr>
<th>i</th>
<th>a[i]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>d</td>
</tr>
<tr>
<td>1</td>
<td>a</td>
</tr>
<tr>
<td>2</td>
<td>c</td>
</tr>
<tr>
<td>3</td>
<td>f</td>
</tr>
<tr>
<td>4</td>
<td>f</td>
</tr>
<tr>
<td>5</td>
<td>b</td>
</tr>
<tr>
<td>6</td>
<td>d</td>
</tr>
<tr>
<td>7</td>
<td>b</td>
</tr>
<tr>
<td>8</td>
<td>f</td>
</tr>
<tr>
<td>9</td>
<td>b</td>
</tr>
<tr>
<td>10</td>
<td>e</td>
</tr>
<tr>
<td>11</td>
<td>a</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>i</th>
<th>aux[i]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>a</td>
</tr>
<tr>
<td>1</td>
<td>a</td>
</tr>
<tr>
<td>2</td>
<td>b</td>
</tr>
<tr>
<td>3</td>
<td>b</td>
</tr>
<tr>
<td>4</td>
<td>b</td>
</tr>
<tr>
<td>5</td>
<td>c</td>
</tr>
<tr>
<td>6</td>
<td>d</td>
</tr>
<tr>
<td>7</td>
<td>d</td>
</tr>
<tr>
<td>8</td>
<td>e</td>
</tr>
<tr>
<td>9</td>
<td>f</td>
</tr>
<tr>
<td>10</td>
<td>f</td>
</tr>
<tr>
<td>11</td>
<td>f</td>
</tr>
</tbody>
</table>

```

```
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.

```java
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];
```