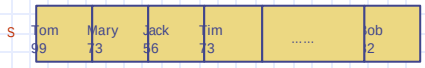
Address Calculating Sort

A sorting algorithm not based on the use of comparisons and achieve O(n) time complexity.

Distribution Counting Sort

- Suppose we have an array of student records:



- Question: sort the array with respect to s[i].grade

Algorithm:

```
Function Distribution_counting_sort(S, n){
Input: a student array S of n records
Output: a sorted array (wrt grade) NS

int count[101]; /*init to 0s */
/* counting */
for (i = 0; i < n; i++) count[S[i].grade]++;
/* accumulating */
count[0]--;
for (i = 1; i < 101; i++) count[i] = count[i -1] + count[i];
/* distribution */
for (i = 0; i < n; i++) NS[count[S[i].grade]--] = S[i];
```


Proximity Map Sort

- Map a key to an array index
- Compute hit counts
- Convert hit counts to a proxmap
- Compute insertion location
- Rearrange input array into ascending sorted order

Working with an Example (1):

i	0	1	2	3	4	5	6	7	8	9	10	11	12
A[i]	6.7	5.9	8.4	1.2	7.3	3.7	11.5	1.1	4.8	0.4	10.5	6.1	1.8

```
/* Compute hit counts */
for (i = 0; i < 13; i++)
    j = Mapkey(A[i]);
    H[j]++;
 }
```

i	0	1	2	3	4	5	6	7	8	9	10	11	12
A[i]	6.7	5.9	8.4	1.2	7.3	3.7	11.5	1.1	4.8	0.4	10.5	6.1	1.8
H[i]	1	3	0	1	1	1	2	1	1	0	1	1	0

Working with an Example (2):

i	0	1	2	3	4	5	6	7	8	9	10	11	12
A[i]	6.7	5.9	8.4	1.2	7.3	3.7	11.5	1.1	4.8	0.4	10.5	6.1	1.8
H[i]	1	3	0	1	1	1	2	1	1	0	1	1	0

```
/* Convert hit counts to proxmap */
RunningTotal = 0;
for (i = 0; i < 13; i++)
    if (H[i] > 0){
        P[i]= RunningTotal;
        RunningTotal + = H[i];
    }
}
```

i	0	1	2	3	4	5	6	7	8	9	10	11	12
A[i]	6.7	5.9	8.4	1.2	7.3	3.7	11.5	1.1	4.8	0.4	10.5	6.1	1.8
H[i]	1	3	0	1	1	1	2	1	1	0	1	1	0
P[i]	0	1	0	4	5	6	7	9	10	0	11	12	0

Working with an Example (3):

i	0	1	2	3	4	5	6	7	8	9	10	11	12
A[i]	6.7	5.9	8.4	1.2	7.3	3.7	11.5	1.1	4.8	0.4	10.5	6.1	1.8
H[i]	1	3	0	1	1	1	2	1	1	0	1	1	0
P[i]	0	1	0	4	5	6	7	9	10	0	11	12	0

```
/* Compute insertion locations */
for (i = 0; i < 13; i++)
      L[i]=  P[MapKey(A[i])];
 }
```

i	0	1	2	3	4	5	6	7	8	9	10	11	12
A[i]	6.7	5.9	8.4	1.2	7.3	3.7	11.5	1.1	4.8	0.4	10.5	6.1	1.8
P[i]	0	1	0	4	5	6	7	9	10	0	11	12	0
L[i]	7	6	10	1	9	4	12	1	5	0	11	7	1

Working with an Example (4):

i	0	1	2	3	4	5	6	7	8	9	10	11	12
A[i]	6.7	5.9	8.4	1.2	7.3	3.7	11.5	1.1	4.8	0.4	10.5	6.1	1.8
L[i]	7	6	10	1	9	4	12	1	5	0	11	7	1

```
/* Distribute A to B, B init 0 */
for (i = 0; i < 13; i++){
  v = A[i];
  k = L[i];
  while(1){
     if (B[k] == 0){
           B[k] = v;
           break;
     }
     elseif (B[k] <= v) k++;
     else {
           temp = v;
           v = B[k];
           B[k] = temp;
           k++;
     }
  }
}
```

i	0	1	2	3	4	5	6	7	8	9	10	11	12
A[i]	6.7	5.9	8.4	1.2	7.3	3.7	11.5	1.1	4.8	0.4	10.5	6.1	1.8
L[i]	7	6	10	1	9	4	12	1	5	0	11	7	1
B[i]	0.4	1.1	1.2	1.8	3.7	4.8	5.9	6.1	6.7	7.3	8.4	10.5	11.5