External Sorting

- Mergesort.

```
Mergesort(A,S,n,B)
      {sort the n elements of A starting from
       position s using B as work place}
  if n < some fixed value
     then use insertion sort
     else
        Mergesort(A,S,n/2,B)
        Mergesort(A,S+n/2,n/2,B)
        merge(A,S,n/2,n/2,B}    {merge to B}
        copy (B,A)            {copy back to A}
```

Runtime analysis:

$$\begin{aligned} T(n) &= 2T(\frac{n}{2}) + \text{ merge cost: n-1 comparisons} \\ &= n \log n - n + O(1) \end{aligned}$$

```
merge(A,S,p,q,B)
  i:=S; j:=S+p; k:=S;
  iend:=j; jend:=j+q;
  while i < iend and j < jend do
     if A[i] < A[j];
         then  B[k]:= A[i];
               i:=i+1; k:=k+1
         else  B[k]:= A[j];
               j:=j+1; k:=k+1;
  Copy remaing values to the end of B
```

Cost: $p+q-1$ comparisons (or less) and $p+q$ moves.

Bucketsort (Radix distribution sort)

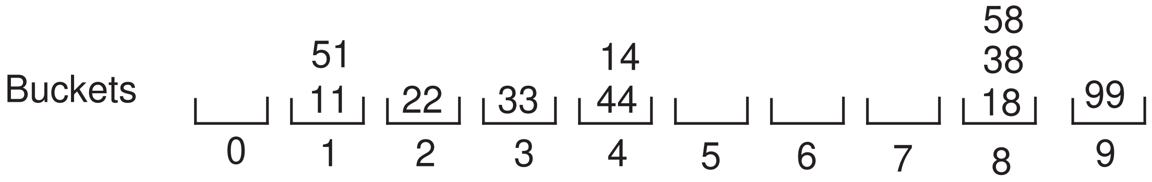
If key is a digit string (or character string, we can sort digit by digit.

```
Bucketsort:
   for each position from least to
       most significant digit do
          add each key to correponding
          linear list (Bucket)
```

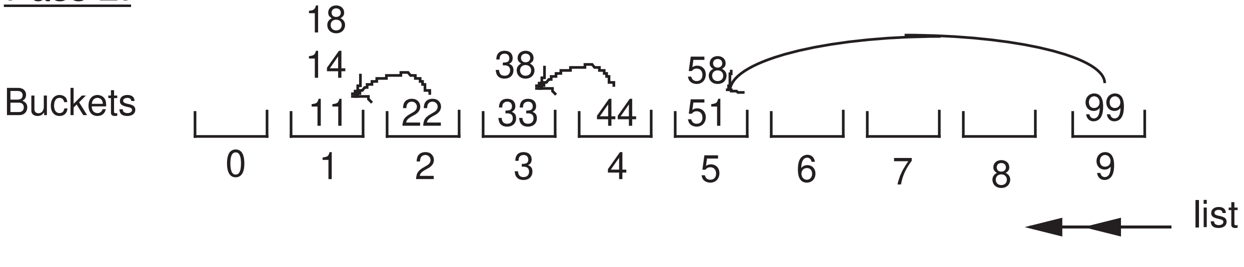
Runtime $\Theta(n \times \#\text{digits})$

Example: Sort 18, 11, 38, 99, 58, 22, 44, 51, 33, 14.

Pass 1:



Pass 2:



11 14 18 22 33 38 44 51 58 99