```
void selectSort(int a[], int size){
int i, j, min, temp;
For(i=0; i<size-1; i++)
{ min=i;
  for(j=i+1; j<size; j++){
    if(a[j] < a[min]) //if item < min
    { min=j //item=new min
    }
  }
temp=a[i];
a[i]=a[min];
a[min]=temp;
}
}
```

Selection Sorting - conceptually most simple

- Finds smallest element in array & swap w/ element in 1st pos.
- then 2nd smallest ↔ 2 pos ... etc

Original Array	1st pass	2nd Pass	3rd pass
3	1	1	1 sorted!
6	6	3	3
1	3	6	6
8	8	8	8

COMPLEXITY = worst/best/average time comp: $O(n^2)$

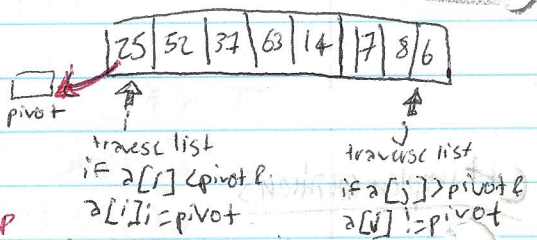
QUICK SORT

- not stable search
- fast, requires less additional space
- DIVIDE & CONQUER (ie. "partition-exchange sort")
- space compl: $O(1)$

- Divides list into 3 parts:
  1) Element s < pivot element
  2) pivot element
  3) elements > pivot element

COMPLEXITY =
worst time: $O(n^2)$
best time: $O(n \log n)$
average: $O(n \log n)$

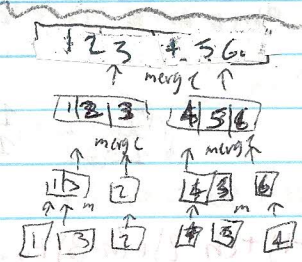
space comp $O(n \log n)$



*if BOTH sides we find the elements !satisfying conditions, we swap & repeat

MERGE SORT

- divide & conquer into N sublists, each w/ 1 element, then merges into sorted lists until 1 is produced



- best for sorting LinkedLists
- $O(n \log n)$ time comp

HEAPSORT