SORTING ALGORITHMS

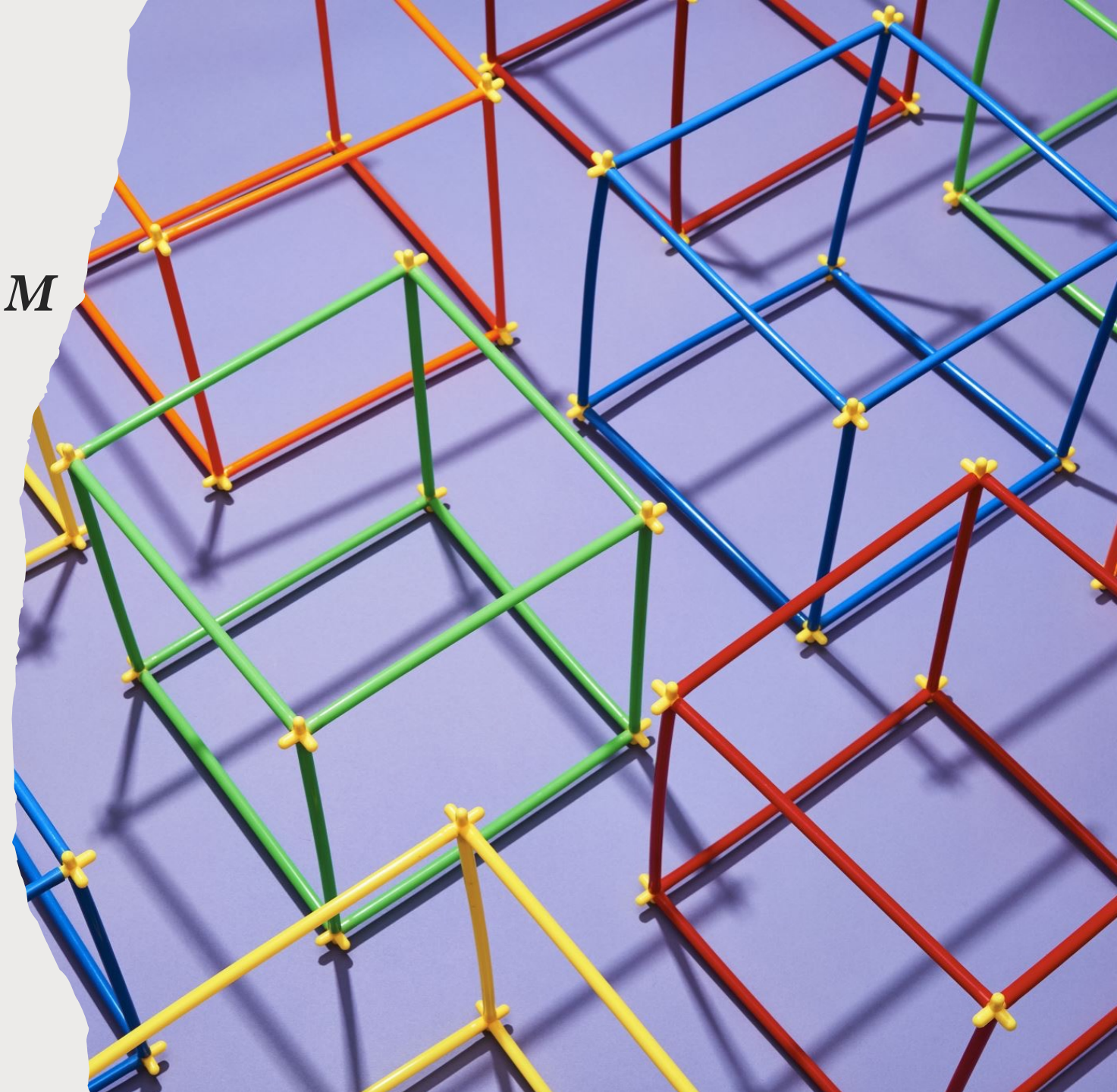
Martzel P Baste

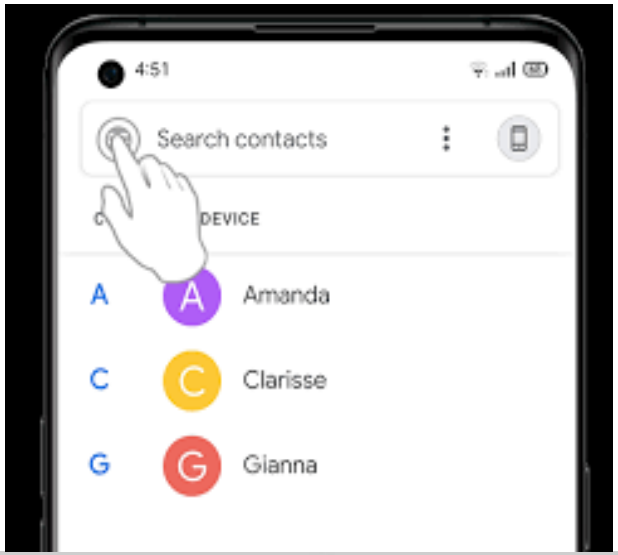
INTENDED LEARNING OUTCOMES

Discuss	Discuss the different Sorting algorithms/techniques (bubble, selection, insertion)
Evaluate	Evaluate the performance of each sorting technique
Prove	Prove the most efficient sorting technique based on time and space complexity
Build	Build and ADT out of the different sorting techniques

SORTING ALGORITHM

is the process of putting data in order; either numerically or alphabetically.





Name	Date Modified	Size	Kind
Week 1	Today at 12:13 PM	--	Folder
Week 2	Jul 31, 2021 at 9:58 PM	--	Folder
Week 3	Today at 12:29 PM	--	Folder
Week 4	Today at 12:29 PM	--	Folder
Week 5	Today at 12:31 PM	--	Folder
Week 6	Today at 12:31 PM	--	Folder

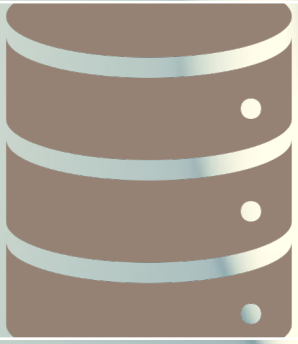
Tagalog Song Booklet

SMART SELECTION	SONG TITLE	SINGER
A	ABOVEANDBEYOND	BARBIE DUMLAO
A	AGOS	
A	AKAP	IMAGO
A	ALAALA	FREDDIE AGUILAR
A	ALAY	MARIUS VILLAROMAN
A	ALELUYA	MANUEL V
A	ALINLANGAN	SUGAR FREE
A	ALITAPTAP	
A	ALLELUIA	DON FISHEL
A	AMBISYOSO	KAMIKAZEE
A	ANINO	IMAGO
A	ANTIPOLO	
A	APOY	GREYHOUNDZ
A	ARAY	MAE RIVERA
A	ASA	SOUTH BORDER
A	AZUCENA	
AA	AALOG ALOG	
AA	ALIPIN AKO	
AAA	AKO AY AAWIT	
AAALY	AY AY AY,I LOVE YOU	MAX SURBAN
AAIK	AKO ANG IYONG KAPWA	GERRY M. DE LEON
AAK	AKO AY KAKANTA	ALLAN K
AAL	AKO AY LALAPIT	
AAM	AKO AY MASAYA	
AAN	AKO ANG NAGTANIM	
AAP	AWIT AT PAG	AEGIS
AB	AKONG BIRTDI	MAX SURBAN
ABAN	ANO BA ANG NANGYARI	APRIL BOYS
ABKB	ANG BOYPREN KONG BADUY	CINDERELLA
ABM	ANGELS BROUGHT ME	GUYSEBASTIAN
ABN	AND BY NOW	MARK BAUTISTA
ABOS	AKO BA O SIYA	HANNAH VILLAME
AC	ANOTHER CHANCE	
ACNP	ANG CUTE NG POKEMON	WILLIE REVILLAME
AD	AMBI DEXTROSE	
AD	ANOTHER DAY	
ADP	ANG DALAGANG PILIPINA	
AF	A FRIEND	KENO
AFY	AWAY FROM YOU	CHRISTIAN BAUTISTA
AG	ALALAHANIN, GUNITAIN	O. PAGSANGHAN
AG	ASCRIBE GREATNESS	UNKNOWN

SMART SELECTION	SONG TITLE	SINGER
A	ADIOS	SLAPSHOCK
A	AISHITRU,ILOVEYOU	
A	AKO	RJ JIMENEZ
A	ALAALA	
A	ALE	THE BLOOMFIELDS
A	ALELUYA	MICH DUMLAO
A	ALIPIN	
A	ALLELUIA	
A	ALWAYS	SHERYN REGIS
A	AMNESIA	LUKAS
A	ANONE	
A	APOLOGIZE	ONE REPUBLIC
A	ARAW	
A	AREGADENG	AWIT PAMBATA
A	ATAT	RIVER MAYA
A	A.I.D.S.	KAMIKAZEE
AA	AKING AWIT	
AA	ANG AKING	DASAL
AAA	ANG AKING AWITIN	
AAD	AS A DOE	GEORGE MISULIA
AAIR	AKO AY IKAW RIN	
AAKA	ANG ALAGA KONG ASO	AWIT PAMBATA
AAM	AKO AY MAGHIHINTAY	CENON LAGMAN
AAMP	AKO AY MAYROONG PUSA	AWIT PAMBATA
AAP	AKO AY PILIPINO	VARIOUS ARTISTS
AATR	AN AFFAIR TO REMEMBER	GRETCHEN BARRETTO
AB	ANNIE BATUNGBAKAL	
ABH	ANYWHERE BUT HERE	TOP SUZARA
ABKS	ANG BAYAN KONG SINILANGAN	ASIN
ABM	A BETTER MAN	OGIE ALCASID
ABO	ALPHA BETA OMEGA	BAMBOO
AC	ANIMA CRISTI	JANDI ARBOLEDA
ACNIM	ANG CUTE NG INA MO	MAKISIG MORALES
ACS	A CERTAIN SMILE	INTROVOYS
AD	ANG DIYOS	OPM HIT
ADAN	ANG DIYOS AND NAGLILIGTAS	CHITO LAZO
ADS	ANONG DALING SABIHIN	
AFM	ANDREW FORD MEDINA	
AG	ALABANG GIRLS	
AG	ARAW'T GABI	TRUE FAITH
AGGK	ANG GUGMANG GIBATI KO	VISAYAN SONG

SORTING APPLICATIONS

Searching algorithms



Database algorithms

Divide and conquer methods

Data structure algorithms

WHY IS IT IMPORTANT?

SORTING PROCESS

price[1]=85 and price[2]=100

price[1]=price[2]; // 100

price[2]=price[1]; // 100

hold = price[2]; // 100

price[2]=price[1]; // 85

price[1]= hold; // 100

price[1]= 100;

price[2]= 85;

The process of sorting an array requires the exchanging of values. While this seems to be a simple process, a computer must be careful that no values are lost during this exchange.

SORTING TECHNIQUES

- Bubble Sort
- Exchange Sort
- Selection Sort
- Insertion Sort
- Shell Sort
- Quick Sort
- Merge Sort
- Many more...

BUBBLE SORT

- As elements are sorted, they gradually "bubble" to their proper location in the array
- It repeatedly compares ***adjacent elements*** of an array.
- The first and second elements are compared and swapped if out of order. Then the second and third elements are compared and swapped if out of order.
- This sorting process continues until the last two elements of the array are compared and swapped if out of order.
- When this first pass through the array is complete, the bubble sort returns to elements one and two and starts the process all over again.

"pass" is defined as one full trip through the array comparing and if necessary, swapping ***elements***.

Array at beginning:	23	10	-50	70	34	5
After Pass #1:	10	-50	23	34	5	70
After Pass #2:	-50	10	23	5	34	70
After Pass #3:	-50	10	5	23	34	70
After Pass #4:	-50	5	10	23	34	70
After Pass #5:	-50	5	10	23	34	70
After Pass #6 (done):	-50	5	10	23	34	70

The bubble sort knows that it is finished when it examines the entire array and no "swaps" are needed.

BUBBLE SORT

	0	1	2	3	4	5
Array at beginning:	23	10	-50	70	34	5
1st	10	-50	23	34	5	70
2nd	-50	10	23	5	34	70
3rd	-50	10	5	23	34	70
4th	-50	5	10	23	34	70
5th	-50	5	10	23	34	70
6th	-50	5	10	23	34	70

len → 6

23, 10, -50, 70, 34, 5

> T T F T T

10 23 50 23 70 34 5 70 — 1st ✓

10 T -50 23 34 5 70

-50 10 F 23 F 34 F 5 70 — 2nd ✓

-50 10 23 5 34 70

-50 10 23 F 23 F 34 F 5 70 — 3rd ✓

-50 10 5 23 34 70

-50 F 10 T 5 23 34 70

-50 10 T 10 F 23 F 34 F 70 — 4th ✓

IMPLEMENTATION

```
int num[]={23,10,-50,70,34,5}; //Array Elements
int hold=0; //Temporary holder
for (int x = 0; x < num.length; x++) {
    for (int y = 0; y < num.length-1; y++) {
        if(num[y]>num[y+1]){
            hold=num[y+1];
            num[y+1]=num[y];
            num[y]=hold;
        } //End of if
    }//End of inner loop
}//End of outer loop
```

EXCHANGE SORT

- It compares elements of the array and swaps those that are not in their proper positions.
- ***The exchange sort compares the first element with each following element of the array, making any necessary swaps.***
- When the first pass through the array is complete, the exchange sort then takes the second element and compares it with each following element of the array swapping elements that are out of order.
- This sorting process continues until the entire array is ordered.

Array at beginning:	23	10	-50	70	34	5
After Pass #1:	-50	23	10	70	34	5
After Pass #2:	-50	5	23	70	34	10
After Pass #3:	-50	5	10	70	34	23
After Pass #4:	-50	5	10	23	70	34
After Pass #5 (done):	-50	5	10	23	34	70

EXCHANGE SORT

Array at beginning:	23	10	-50	70	34	5

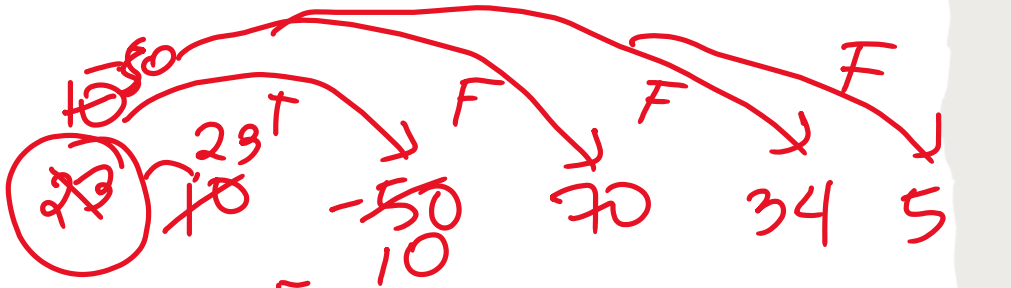
1st loop: -50 | 23 10 70 34 5

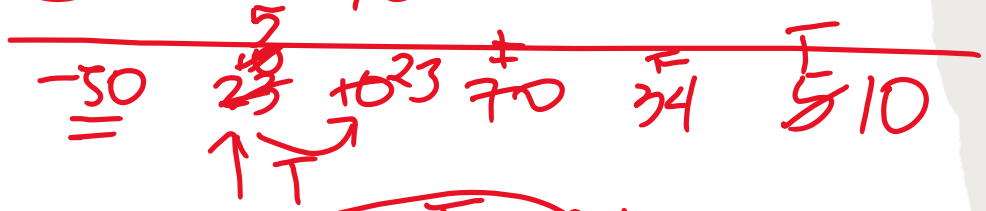
2nd loop: -50 5 | 23 70 34 10

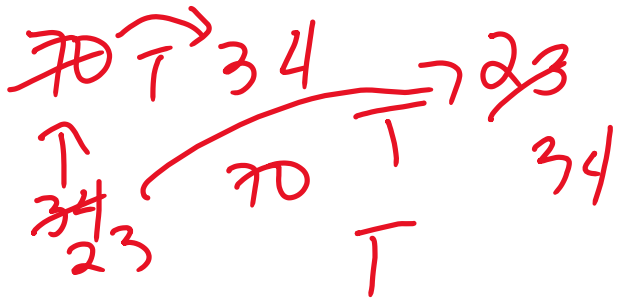
3rd loop: -50 5 10 | 70 34 23

4th loop: -50 5 10 23 | 70 34

5th loop: -50 5 10 23 34 70







Bubble Sort = outer → 6 × inner → 5 = 30 times looping

Exchange Sort = outer → 5 × inner → 5 → 25 times

IMPLEMENTATION

```
int num[]={23,10,-50,70,34,5};//Array Elements
int hold=0; //Temporary holder
for (int i = 0; i < num.length-1; i++) {
    for (int j =(i+1); j < num.length; j++){
        if(num[i]>num[j]){
            hold=num[j];
            num[j]=num[i];
            num[i]=hold;
        } //End of if
    } //End of inner loop
} //End of outer loop
```

SELECTION SORT

- The ***selection sort*** is a combination of searching and sorting.
- ***During each pass, the unsorted element with the smallest (or largest) value is moved to its proper position in the array.***
- The number of times the sort passes through the array is one less than the number of items in the array. In the selection sort, the inner loop finds the next smallest (or largest) value and the outer loop places that value into its proper location.

Array at beginning:	23	10	-50	70	34	5
After Pass #1:	-50	10	23	70	34	5
After Pass #2:	-50	5	23	70	34	10
After Pass #3:	-50	5	10	70	34	23
After Pass #4:	-50	5	10	23	34	70
After Pass #5:	-50	5	10	23	34	70
After Pass #6 (done):	-50	5	10	23	34	70

SELECTION SORT

Array at beginning:	23	10	-50	70	34	5

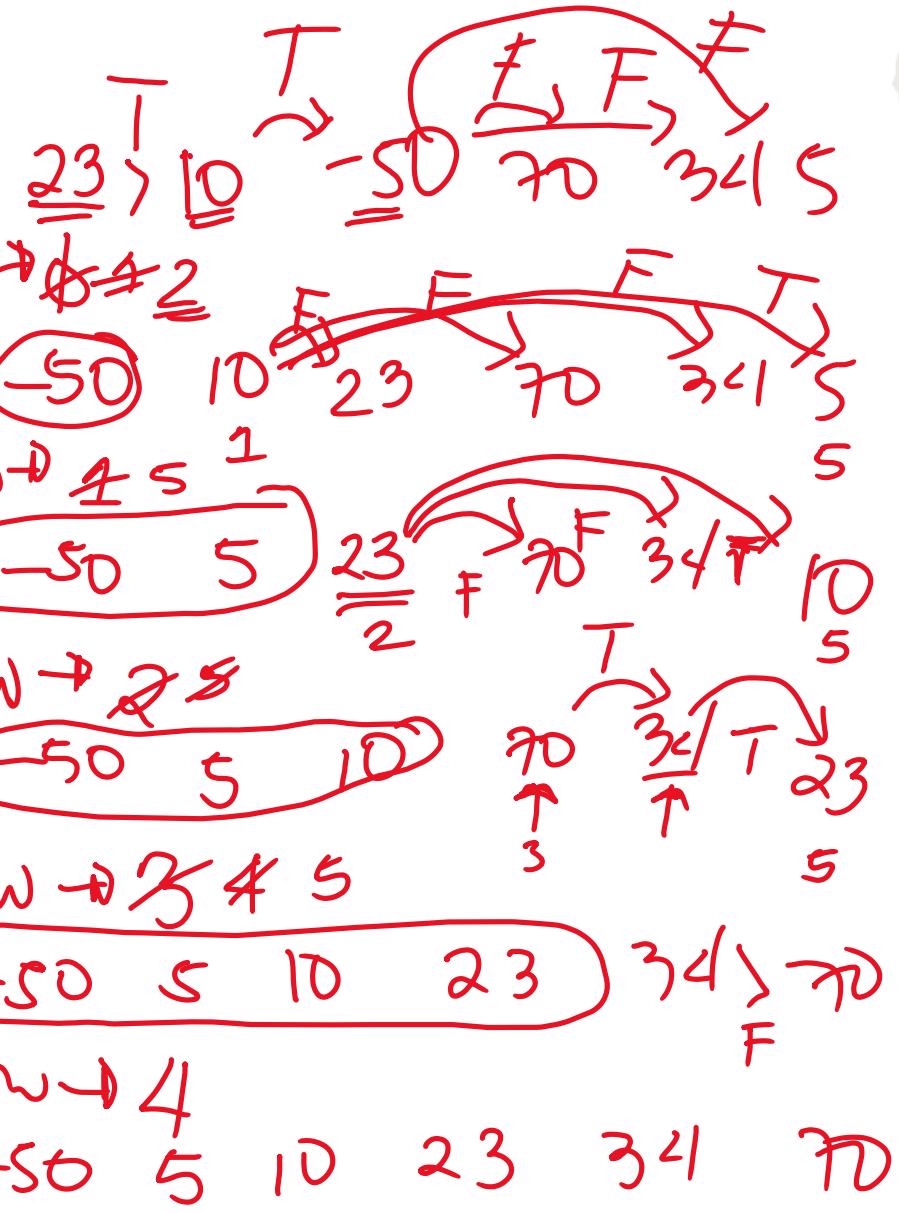
1st loop -50 | 10 23 70 34 5

2nd loop -50 5 | 23 70 34 10

3rd loop -50 5 10 | 70 34 23

4th loop -50 5 10 23 | 34 70

5th loop -50 5 10 23 34 | 70



IMPLEMENTATION

```
int num[]={23,10,-50,70,34,5}; //Array Elements
String all=""; //String object
int hold=0,lowest=0, inner=0;
for (int outer = 0; outer < num.length; outer++) {
    lowest=outer;
    for (inner=outer+1; inner<num.length; inner++) {
        if(num[inner]<num[lowest]) {
            lowest=inner;
        } //end of if

    }//end of inner loop
    hold=num[lowest];
    num[lowest]=num[outer];
    num[outer]=hold;
} //end of outer loop
```

INSERTION SORT

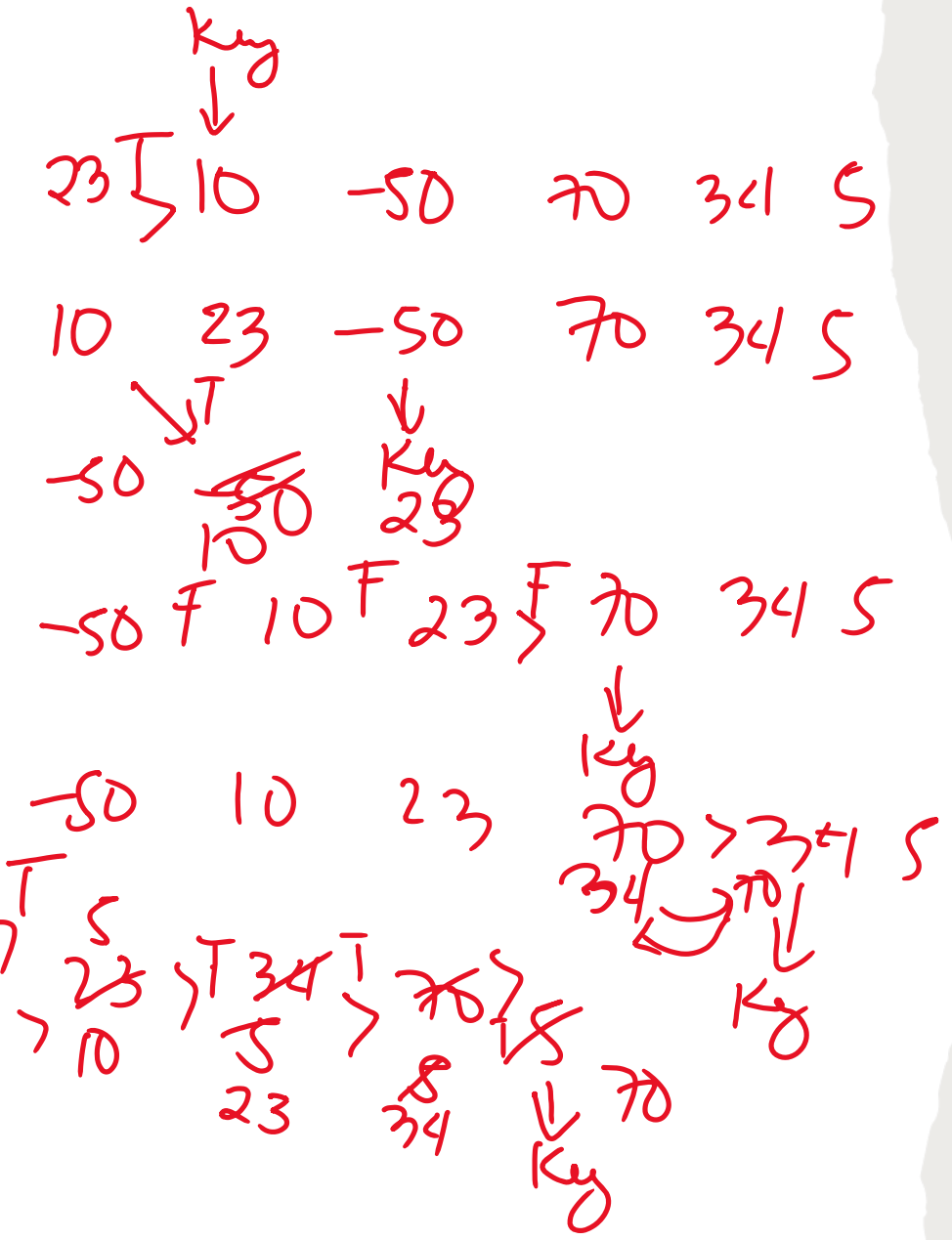
- The ***insertion sort***, unlike the other sorts, ***passes through the array only once.*** The insertion sort is commonly compared to organizing a handful of playing cards
- It splits an array into two sub-arrays (sorted and unsorted

Array at beginning:	23	10	-50	70	34	5
1st sub-array(sorted)	23	10	-50	70	34	5
2nd sub-array (unsorted)	10	23	-50	70	34	5
	-50	10	23	70	34	5
	-50	10	23	70	34	5
	-50	10	23	34	70	5
2nd sub-array **empty**	-50	5	10	23	34	70

The insertion sort maintains the two sub-arrays within the same array. At the beginning of the sort, the first element in the first sub-array is considered the "sorted array". With each pass through the loop, the next element in the unsorted second sub-array is placed into its proper position in the first sorted sub-array.

INSERTION SORT

Array at beginning:	23	10	-50	70	34	5
1st loop	10	23	-50	70	34	5
2nd loop	-50	10	23	70	34	5
3rd loop	-50	10	23	70	34	5
4th loop	-50	10	23	34	70	5
5th loop	-50	5	10	23	34	70



IMPLEMENTATION

```
int num[]={23,10,-50,70,34,5}; //Array Elements
String all=""; //String object
int key=0;
for (int outer = 1; outer < num.length; outer++) {
    key=num[outer];
    int inner=outer-1;

    while ((inner > -1) && (key<num[inner]) ) {
        num[inner+1] = num[inner];
        inner--;
    }
    num[inner+1] = key;
} //end of outer loop
```