24/02/2023 Selection Sort First of all we need to understand what is sorting. Sorting is basically rearranging elements in either increasing or decreasing order. 1,3,5,4,7,2 - These elements are not in sorted order as initially it was increasing but 5 to 4 is of decreasing order & hence we can say it is not sorted. Madrilminison el- I me dola

\$	
Steb-1	In selection sort, an element is picked at then is placed at correct place. In this we have to place the minimum element at the right place. Algorithm of Selection Sort 10 1 4 8 5 7 Find the minimum element from the
	avray & place at i = 0.
	min = 1 Swap (aur [o], aur [min Index]); 1 10 4 8 5 7
Step-2	Find minimum element in subarray start from index = 1 to n-1 & place at 1st index.
- 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	min = 4 Swap (aur [1], aur [min Index]); 1 4 10 8 5 7
	Find minimum in sub-array start from index = 2 to n-1 4 place at 2nd index min = 5 Swap (arr [2], arr [min Index]);

Scarined with Jam

	Page
	1 4 5 8 10 7
Steb	index = 3 to n-1 & place at 3rd index
	min = 7
	swap (aur [3], aur [minIndex]);
	1 4 5
	5 7 10 8
Stel	from index = 4 to n-1 & place at 4th index
	min = 8 Swap (avv [4], avv [min Index]);
	1 4 5 7 8 10
	Now only I element remaining, so need to check it as there is no element on its right and hence we get the sorted away.
Mote	+ Every step we do is termed as round or -
	Step-1 = Round-1 = Parse-1
	We can observe that for 6 elements, 5
	Founds are required to sort the array. Hence 1 or n elements in -1 rounds are
	Hence for n elements, n-1 rounds are required to sort the away via selection
	Sort.
	Scarineu with Ca

Scarined with Cam

~_	
	Code
	C + () solo solo solo solo solo solo solo so
MINE.	void selection Sort (vector <int> aur){</int>
201	
	int n = arr size()
	for (int i=0; i <n-1; i++)="" th="" {<=""></n-1;>
	// Assuming minimum element to be current
	int min Index = 1
	for (int $j = i + 1$) $j < n : j + +$) {
	// Comparing with minimum Index.
1	if (ov [min Index] > ov [j])
7.8.1	min Index = j; 3 //Updating minimum
	3 //Updating minimum
	3
	// Placing min element at correct position
	swap (avr [min Index], avr [i]);
	3
Ju = 1	5
- Wote	We have started inner loop from i+1
	$\frac{1}{1}$
	La coes not make sense to compage
	with itself.
	To an analytic state of the sta
	Time complexity
	time (omblexity gets many)
	time complexity gets multiplied.
	Space complexity
	O(1) as only variables
	O(1) as only variables have been created 4 we don't have to consider
	space of the input.
H	Scarnieu with Ca

Scarined with Cam

	Bubble Sort
	The logic behind this sorting technique is that in the ith round, ith largest element will be placed at the right position.
	that in the ith round ith largest element
	will be blaced at the right position.
	1 19 19 POSITIOIL
	Algorithm for bubble sort
	10 1 7 6 14 9
	10 1 7 6 14 9
	Paulad
	Round - 1
*	10>1 → swab
	1 10 7 6 14 9
*	1077 → SWab
	1 7 6 14 9
*	
	1 7 6 10 14 9
*	10714 - no swap
	1479 - SWab
	1 7 6 10 9 14
	At right place I
	the state of the s
	Round-2
	1 7 6 10 9 14
*	Sorted -
	1 > 7 -> no swap
1	176 -) SWAP
*	1 6 7 10 9 14
*	1>10 -> no swap
	$\begin{array}{cccccccccccccccccccccccccccccccccccc$
	6

Scarined with Caril

	Classmate Date
***	208 Page
	Round - 3
	MOUTICE S
7 .	1 6 7 9 10 4
	Sorted boundary
*	176 7
*	6>7 No swap in any case.
*	779
	Round-4
	1 6 7 9 10 14
	Sorted boundary
*	176 ? No swap in any case
·*	6>7 5
-	
	Round - 5
-	
	1 6 7 9 410- 14
-	Sorted boundary
*	176 → no swap
	No. 1.10 dan/l for l
	Now we don't have to go in round - 6 as
	only one element is left & hence we can't
4	compare it with any element. Hence we get sorted array in n-1 rounds where n is the no. of elements.
	n is the no of elements.
	Obtimization in Bubble Sort
	Also we can observe that it
+	there was no swap done & hence the
	there was no swap done & hence the array gets already sorted & We don't have to go in further rounds.
	to go in further rounds.
*	
	m
	Scarifica With Cam

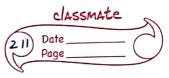
Scarineu with Caril

	Code
_	void bubble Sort (vector < int> 4 aur) {
	VOICE BUILDING COLOUR CITIES COLOUR C
	int n = aur. size();
	for (int i=0; i< n-1; i++) {
	bool swapped = false i
	for (int $j = 1$) $j < n - \lambda$) $j + +$) t
	: C C
	if (ovr[j-1] > ovr[j]) {
	Swapped = true; Swap (avr[j-1], avr[j]);
	ξ
	2
	if (Iswapped) {
	break
	3
_	3
_	3
_	
_	Best case lime Complexity
	Best case occurs when the time complexity
	Best case Time Complexity Best case occurs when the away is already sorted. Here the time complexity is O(n) as only n=1 comparasions will be
	done.
	Normal case Time complexity O(D2)
_	Worst case Time complexity
_	Worst case Time complexity Worst case occurs when the array is reverse sorted & hence time complexity her occurs
_	reverse sorted & hence time complexity
_	here is O(n2).
-	D I I I I I I I I I I I I I I I I I I I
-	By optimization, best case time complexity

	Classmate
210	Date Page

	reduces to O(n).
-	Space complexity O(1) as only variables are created.
Note-	Use case of selection sort is incase of small anyons:
je I	Use case of bubble sort is when ith largest elements to be put at correct place.
	Insertion Sort This means to insert the element at right blace.
	2 1 5 4
	1 2 5 Insert 4
	1 2 4 5 → Sorted array
<u> </u>	Algorithm for insertion sort
. 1)	
<u></u> ع)_	10 is already at right place or will automatically be placed at right place. Pick element at index = 1
	1<10, shift lo & place I there. There is no swapping involved.

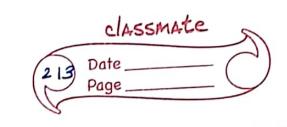
Scarineu with Cam



	No. of the control of
	1 10 7 6 14 9
	Sorted
3)	Pick element at index = 2
	7<10 & 7 should come in blue I & 10.
	7>1 J
	So shift 10 & copy 7 at the empty place.
	1 7 10 6 14 9
	Sorted
4)	Pick element at index = 3
	6 < 10
	6 < 7
	6 > 1
	Shift 10, then shift 7 & then copy 6 at
	the empty place
	\(\frac{1}{2}\)
	1 6 7 10 14 9
	Sorted
5)	Pick element at index = 4
	14>10 → nothing to do.
	ų.
	1 6 7 10 14 9
	Sorted
6)	Pick element at index = 5
1	who splings in our standards and
	9 < 14
	9 < 10
	9>7
	Shift 14, then shift lo & then copy 9 to
	" Scarineu with call

	the empty place
	1 6 7 9 10 14
	Hence the array is sorted now.
1	
	Code
<u> </u>	void insertion Sort (vector <int>arr) {</int>
·	
	int n = arr size();
ļ	for (int i=1) i <n; i++)="" th="" {<=""></n;>
<u> </u>	int val= avr [i] : // Pick element
Tx	$\inf_{j=1,\ldots,n} \frac{1}{j} = \frac{1}{j} = \frac{1}{j}$
1.	for ('j = 0 ; j = -) {
·	//Compare element
·	if (avr [j] > val) {
⁻	//shifting operation arr [j+1] = vali
[·	else {
[<u>-</u>	breaki
	3 - value 10 tage
	3
4.	// Copy Step avr [j+1] = vali
	3
L-	3
	Time complexity
- 	O(n²) in normal & worst case.
	O(1) It The Best case i'e already
- 	sorted. Space complexity
	O(1) as only variables are created.
7-1	oreated.

Scarineu with Cam



Use case of insertion sorted is in case of small arrays or when array is partially sorted.

Inbuilt sort function

sort (aur. begin(), aur. end()) i is used to

sort the vector

we need to include algorithm header file

to use this inbuilt function.

Scanned with Cam