Kayvaun Khoshkhou

Professor: Duc Ta

Due Date: 11/20/2020

# Assignment 5

# Sorting

### 1. Selection Sort

L.	Sel	lecti	on Si	ort:	Ran	dom	12	int	egers.	21	-10	3	in as	unair	ng
-	L	17	8	72	34	63	29	35	2	10	80	99	93		
-	7.	2	8	72	34	63	29	35	17	10	80	99	93	1 11	
2	3.	2	8	10	34	63	29	35	17	72	80	99	93	12	
	4.	2	8	10	17	63	29	35	34	72	80	99	93	1.05	
	5.	2	8	10	17	29	63	35	34	72	80	99	93	177	
	6.	2	8	10	17	29	34	35	63			99	93	12	
	7.	2	8	10	17	29	34	35	63	72	80	13	99	1.50	
1	-	4	-	1.0	17	7	a	34	25	- 6	3	72	80	93	9971

Selection sort algorithm sorts an array by finding the minimum element over and over again from an unsorted part and placing it at the beginning of the array.

#### 2. Insertion Sort

65 80	
2. Insertion Sort: Random 12 integers & 1-1003 ascending	
2. MISCIETON SOIT. KANNOM 12 MITGETS E.	
1.[17] 8 72 34 63 29 35 2 10 80 99 93	
2 [17 87 72 34 63 29 35 2 10 80 99 93	
2. 11 0 12 37 32 3 10 50 99 93	
3. T8 17 72 34 63 29 35 2 10 80 99 93	
4. [8 11 12 37 03 2 3 2 3 2 9 9 9 9 9 9 9 9 9 9 9 9 9 9	
J. L 8 11 31 12 3 12 3 10 80 99 93	
6.68 11 34 63 12 21 33 2 10 80 99 93	
1.18 17 29 34 63 12 35 2 10 00 17	
8. 8 17 29 34 33 03 12 2 107 00 00 02	
7. 2 8 11 29 34 33 63 12 103 00 12	
10 17 8 10 1/ 21 39 33 63 12 00 1	
11.52 8 10 17 29 3435 63 72 80 99	
17. (28 10 17 29 3435 63 72 80 99 93	
12 17 8 10 17 29 34 35 63 72 86 93 99 ]	6
10 1 2 2 4 4 2 4 4 6 11 41 8 5 11	
2 8 10 17 29 34 35 63 72 86 93 99	) /
1 3 7 10 17 34 34 35 44 70 50 8	

Insertion Sort works in a way where you take the value from an unsorted part and place it in front of a desired position. In this case I am sorting in ascending and you can see for example on step 6; I mark 34 and 29, this indicates that I am going to take 29 and move it in front of 34 and place it there, while also pushing back all the other positions of the array back one in order to make up for the empty position that was left behind when I withdrew 29.

# 3. Shell Sort

3.	Shel	1 500	<u>t</u> :	Rand	om_	12 int	egers	51	-100	3 9	scend	ing	100	
١.	۱٦	8	72	34	63	29	35	2	10	80	99	93	12/	2 = 4
J.	17						35			7 11		Bu L	V 7 9 1	
7.		ż		-	Y .			8						
2. 3. 4.		-	10				-		72	- 0				
5.	_	V.P.		34	-				- 1	80	0.0		1 1	
6		A. F			63	29					99	9 2	1	
-	-					21	-	200	4 74			13	11	7
	17	2	10	34	63	29	35	8	72	80	99 9	73	6/-	
	~					1	-	0.	10		1 ( /		6/2 :	3
1.	17		100	34			35		(Same	80		1	0/2	
2	,	ż			63			8		9	9	4	47	
3	-		(0	7.11		29	A .		72	a et a	9	3	5.3.16	
	17	2	10	34	12	20	33.)	-	7.	74 A	A (12	38		111
	111	, _	10	51	63	29	35	8	72	80 9	9 93			
_	-		-			3 1			4 1		- 1		3/2=	1
	1. [17	1	1				15					W.T.L.	B L B -	
2		1 2	1	- 4	-	150	18V2	-	-		-		Ch Tr	100
	3.12	•		10		, TO			10 mg			Services		7
-	+. 17	1	0 1	7 3	,47	1	391		1/10		13			
	5.5	2 1	0 1		34	637	Mar.	MY	5-5			1.17	13	
	6.[		10 1	7	34		297				14 9,35		Tares Van	
	-			7	29	34	63	35]	1 1				9	
- 3	8.5				29		35	63	87			GE H		
	9, C	2	8	10	17			35	63	72		7		
100	0/		8	0	17			35	63	72	80		7	
		2	8 1	0	17		34	35	63	72	80	99		
1	2.	2		0	17		34 34	3 <i>5</i> 3 <i>5</i>	63	72		99	93	_
	ノー	4	8 I	0	11	21	)	رر	63	72	80	93	97]	
		2	8 1	0	17	29	34	35	63	72	- 80	93	99 1	/

Shell sort is good at optimizing sorting when the exchange of values are far apart on the sorting list. The formula n/2 slowly minimalizes the sort until n=1 and each position is quickly and efficiently exchanged with its neighbor to get sorted. In this case it happened ascendingly.

4. Bubble Sort

	1. Bubble	Sort	Random	12	integ	1e13	21.	100	3	asun.	0	
	17 8	72	34 63	3 29	35	2	10	80	99	93	3	1:5
	1	12	31 0							[]	N	+
Pass 1:	1, 17 8		11					-	1 2	1		1.
	8 17						C	1 1	- 1			1 41
3	1.					36	3	_				1.7
4		72	34		1 00	19	1	1				100
5		34	72	- (		- 13						8
6			72 63	-	S 1	-7_		-				13
/			63 72		( 1) =			- 1	700			100
8		-	72		A							111
9.				72	35			1.1.	- 17	*/		11
10,		5.5	50	72	572				10	45		12
(1,	30	5.4.	4. 4.9	5.	72	2				37	4.7	13.
/2. /3.	-1C	-11			2.	72						14
19.	0.0	7.9	377 7	JE 163		72	10	33	10	43	23.0	SIK
15.	V		200			10	72					
16.	72: 80	- 5.	01	. 5-	25		77	8	2	1 8		
17.	4	E						80		9		1
18.		1111							90		3	
- 0.						_				3 99	1	
19									9:	5 //		
19.	8 17 2	4 /	2 79	2 5	7		10	72	93			79
	8 17 3			35			10	72	80		3 8	79
s /t									80	9	3 8	
s /t	8-								80	9	3 8	
ass 2:									80	9	3 8	
ass 2:	8 17	17 34	34 63	29				10	80	9	3 8	
ass 2: 1. 2. 3. 4.	8 17	17 34	63	29				10	80	9	3 8	
ass 2: 1. 2. 3. 4. 5.	8 17	17 34	63	29	35		2 1	10	80	9	3 8	
ass 2: 2. 3. 4. 5.	8 17	17 34	63	29	35		2 1	10	80	9	3 7 93 1	
ass 2: 2. 3. 4. 5. 6. 7.	8 17	17 34	34 63	29 3 3 3 5 6 3	35		2 1	10	80	9	3 7 93 1	
255 2: 2. 3. 4. 5. 6. 7.	8 17	17 34	63 63 29 63 29 63	29 3 3 5 63	<i>35</i>		2 1	10	80	9	3 7	
ass 2: 2. 3. 4. 5. 6. 7. 9.	8 17	17 34	63	29 3 3 3 5 6 3	<i>35</i>		2:1	10	80	9	3 93	
255 2: 2. 3. 4. 5. 6. 7.	8 17	17 34	63 63 29 63 29 63	29 3 3 5 63	35 5 2 63 63	10	2 1 1	10	80	9	3 7	
255 2: 2. 3. 4. 5. 6. 7. 9. 10.	8 17	17 34	63 63 29 63 29 63	29 3 3 5 63	<i>35</i>	10	2 1	10	80	9	3 7 93 9	
255 2: 2. 3. 4. 5. 6. 7. 9. 10. 11. 12.	8 17	17 34	63 63 29 63 29 63	29 3 3 5 63	35 5 2 63 63	10	2 1	2	72 0	9	3 93	
ass 2:  2. 2. 3. 4. 5. 6. 7. 9. 10. 11. 12. 13.	8 17	17 34	63 63 29 63 29 63	29 3 3 5 63	35 5 2 63 63	10	2 1	2 2	80 72 d	9 9	3 7 93 9	
ass 2:  2.  3.  4,  5.  6.  7,  9,  10  11.  12.	8 17	17 34	63 63 29 63 29 63	29 3 3 5 63	35 5 2 63 63	10	2 1	2 2 2	72 0	9	3 7 7 3 1	

6.	8 17 34 29 35 2 10 63 72 80 93	99
Pass 3: 1	8 17	1 :
330	17 34	18
Z. 3.	34 29	
4.	29 34	-,1
5.	29 34 34 35	. "
6.	35 2	á
7	2 3 5	. 7
8.	35 10	9
9	20.35	t,
10.		117
11.	63 72	11
12.	72 80	12
13.	80 93	21
14.	93 99	-
Fig. 7. 7	8 17 29 34 2 10 35 63 72 80 93 99.	/
Pass 4: 1.	8.17	L . 38
	17 29	
Z. 3.	29 34	,
4.	34 2	v .
5.	2 34	4
	34 10	
6. 7.	10 34	
7	34 35 35 63	F
8.	35 63	33
F. 9.		1.57
٩.	63 72	
9.  0,	63 72	1
9.  0,	63 72 72 80 80 93	1
9.  0,	63 72	V.
9. 10.	63 72 72 80 80 93	

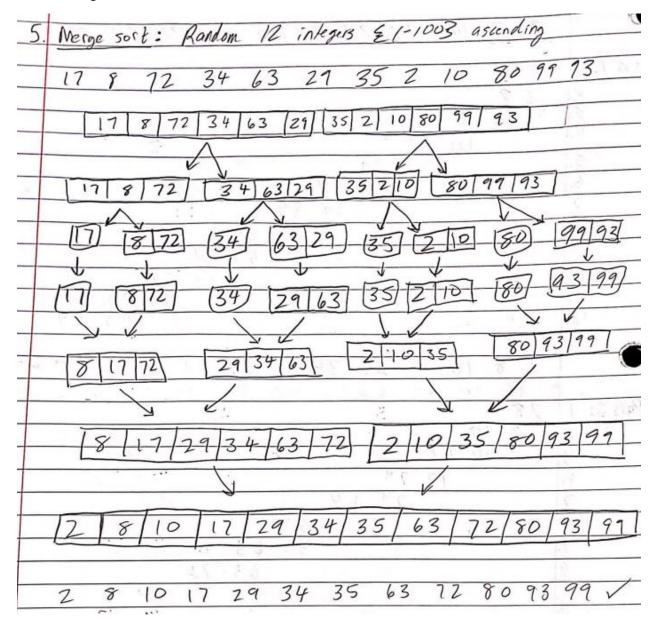
711	8 17	29	2	10	34.	35	63	72	- 8	0 9.	399	9
Dass 5: 1.	8 17	1							5	1 4		
2.	l T							15	- 1		15	
3		29	2				-	11 7	Ç i	ŕ	18	
4		2	29					3.0	0		I b	
5.			29	10			5.4	7	-		7	
6.			10	29		111						
7,				29	34	8	1				1	
8.				7	3.4	33					13	
9.			- (		1 2 5	35	63				17	
10,			er 1,	d	3:		63	72			165	
		3 1	77	6.6		-		72	80			
12.	XA	03	77			20			80	93		Y
13.	100 100	11.17						7	11 11	93	99	
	1	1 76		17 :	* 115	-	7 :	× × 3				1
		-	300									
1 <sub>055/0</sub> , 1.	8 1	7 2	10	29	34	-	63	72	80	13	99	V
Passb: 1:	8 17	0.4	10	16.30	3.6	-	15	1		93	99	V
2.	8 17 8 17	2	10	16.30	3.6	-	15	1	<u> </u>	1	1/4	V
2.	8 17	2		16.30	3.6	-	15	7 5	<u> </u>	1	\$ 4   \	V
2. 3, 4.	8 17 8 17	2 17 17	10	16.30	45	-	15	7 5	<u> </u>	1	\$ 1	V
2: 3, 4. 5.	8 17 8 17	2	10		45	7	15	7 5	<u> </u>	1	\$ 1	
2. 3, 4.	8 17 8 17	2 17 17	10	29:	45	7	9 1	7 5	<u> </u>	1	\$ 1	
2. 3, 4, 5. 6. 7	8 17 8 17	2 17 17	10	29	34	2	15	7 5	<u> </u>	1	\$ 4 \$	
2. 3, 4, 5. 6, 7, 8.	8 17 8 17	2 17 17	10	29:	34 34	35	2 2 3 4 3 4	7 5	<u> </u>	1	\$ 4 2 (*)	
2. 3, 4, 5. 6, 7, 8. 9,	8 17 8 17	2 17 17 10	10	29	34 34	2	63	118	<u> </u>	1	\$ 4 \$	
2. 3, 4, 5. 6, 7, 8.	8 17 8 17	2 17 17 10	10	29	34	35	2 2 3 4 3 4	72	P	1	% - I	
2. 3, 4. 5. 6. 7, 8. 9,	8 17 8 17	2 17 17 10	10	29	34	35	63	118	80		% - I	
2. 3, 4. 5. 6. 7, 8. 9, 10,	8 17 17 2	2 17 17 10	10	29	34	35	63	72	P	1		
2. 3, 4. 5. 6. 7, 8. 9,	8 17 17 2	2 17 17 10	10	29	34	35	63	72	80			
2. 3, 4. 5. 6. 7, 8. 9, 10,	8 17 17 2	2 17 17 10	10	29	34	3535	63	72 72	80	93	% - I	

	8 2 10 17 29 34 35 63 72 80 93 99
Pass 7: 1.	82
7,	28
3,	8 (0)
4.	10 17
5.	17 29
6.	29 34
7.	34 35
8	35 63
9	63 72
100	72 80
th	80 93
12	
ATO THE	- 0.7 00 07 00
(0	2 8 10 17 29 34 35 63 72 80 93 99
Pass 8: 1.	28
2.	8.10
3.	10 17
4.	17 29
5.	29 3 4
6.	34 35
٦.	35 63
8	63 72 72 80
9.	80 93
16.	
11,	
0	2.8 10 17 29 34 35 63 72 80 93 79.
	No change between pass 7 and 8 so bubble sort

<sup>&</sup>quot;No change between pass 7 and 8 indicated the bubble sort is complete"

Bubble sort works in a way that repeatedly swaps adjacent elements within the sort if they are not in the desired order. In this case I wanted to arrange the sort in an ascending format, so from left to right I processed each element side by side and arranged it ascending and repeated my process until I did not need to repeat anymore because each rendition would yield the same sort. This indicates completion.

### 5. Merge Sort



Merge sort divides itself into simpler segments so that quick adjustments can be made, and then once the adjustments have been made, a merge takes place along with another adjustment to arrange an order. In this case we wanted ascending order. 12 did not divide that evenly into 4, since it gave us groups of 3 elements, but it wasn't an issue because we split it into 2 and 1, arranged the 2, then merged the 1 back in accordingly, then once we had 3 again, another merge would take place and we would have 2 groups of 6, and then merge that accordingly to get the sorted group of 12 elements that we had unsorted at the beginning.

# 6. Quick Sort

6.	Quick Sort: Random 12 integers & 1-1003 ascending
	P THE RESERVE AND THE WAY AND THE PERSON OF
1.	17 8 72 34 63 29 35 2 10 80 99 93
	i R
	B 36 13 66 42 56-62 30 61 43 16 7
Z.	17 8 72 34 63 29 35 2 10 80 99 93
	L R S
	50 FE 18 62 62 63 55 45 56 15 51 55 55
3.	10 8 72 34 63 29 35 2 17 80 99 93
-).	L
	A CONTRACTOR OF THE PROPERTY O
-: 4,	10 8 72 34 63 29 35 2 47 80 99 93
1,	L R
- 8	The company of the second of t
5.	772 61 99 63
	10 8 17 34 65 29 35 2 12 00 11 13
F 11	THE APE, WALLE BY BY U.S. CHECK ST. S.
-	21 21 21 22 23
6.	10 8 17 34 63 29 35 2 72 80 99 93
51	We hower as he as to the state to
7	10 8 2 34 63 29 35 17 72 80 99 93
7,	10 8 2 34 63 29 35 17 72 80 99 95
	- 60 08 61 560 550 560 11 11 11 11 11 11 11 11 11 11 11 11 11
Q	10 8 2 34 63 29 35 17 72 80 99 93
8.	10 0 2 39 63 21 33 11 12 00 11 13
	THE RESERVE THE STATE OF THE ST
a	
-[.	1082176329353472809993
10	10 2 - 12 12 28 26 74 - 1 21 99 02
/0.	1- 0 1 1 0 5 2 1 3 - 2 1 1 0 - 1 1 1 3
	LR .
/1	P 10 7 17 12 28 25 21 72 51 96 01
(1.	10 8 2 17 63 29 35 34 72 80 99 93
	L = 2 12 12 14 11 12 17 R

	P					2.00			- 30	0		
12.	10	8	Z	17	63	29	35	34 7	2 80	99	93	
-	L	_	R			Ä. L			1 72	1	VI	
13.	2	8	10	17	63	29	7 (	- //	2 0	1 9/	92	
121	L		R	' '	63	21	35	34 /	28	0 19	93	1
		-	1	P	71			-				
14.	2	8	10	17	63	29	35	34 7	2 80	99	93	
1		_		LR	P		11 1	0.63		1		1
15.	2	8	10	17	63	2	9 30	5 34	72	80	99	9 93
		119	10.00	= 0	2			, ,,	, -		,	9 93 R
16.	7	0	10	17	P							
16.	2	8	10	17	63 L	29	3 5	1 100	72	80	99	93
					L			R	31.5	1		137
17.	2	8	10	17	34	29	35		72	80	99	93
4		4			L		100	R	1136	1 1	G.	
18.	2	8	10	17	P	2.0	/	, ,,				
10.	-	-	-	17	34 LR	29	3 5	63	72	80	99	93
			The same		P	7			1 100	1	1	
19.	2	8	10	17	34	29	35	63	72	80	99	93
		_	MISW.		L	113	F 19	186	59.9	- 2	8	R
20,	7	2	10	17	P	20		12	72	80	00	60
0,	-	8	10	17	34	R	35	63	12	00	99	93
						P				1		
21.	2	8	10	17	29	34	35	63	72	80	99	93
		-	181	-	L	R	18	11	71	EX. A		01110
12	-	_	- 1		20	P		1 -		0.4	99	0 -
22	12	8	0	17	29	34 LR	35	63	72	80	99	93

								P					
23	2	8	10	1	7 2	29	34	35	63	72	80	99	13 R
								P					
24.	2	8	10	17		9	34	35 LR	63	72	80	99	93
21	2	9	10	17			2 1/2		P	72	80	99	13
25.	2	8	10	17	2	7 =	34	35	63 L	10	0.5		R
26,	Z	8	10	17	29	3	4 3	35	P 63 LR	72	80	99	93
									CK	P			
27.	2	8	10	17	21	34	35	6	3	72 L	80	99	93 R
28.	Z	8	10	7 2	.9	34	35	6		P 72 LR	80	99	93
29	2	8	10	17	29	34	35	63		72	P 80 L	99	93 R
30.	2	8	10	17 2	9 3	4 3	35	63	72		P 80	99	93
				- 20			-v-				R	P	
1.	2	8	10	17	29	3	4 3:	5 67	> 7	12	80	99 L	93 R
2.	2	8	10	17	20	? 3	4 3	5 6	3 7	2	80	93	P 99
		_				_					_	L	R
3.	2	8	10	17	29	34	35	6	3 7	2	80	93	99
_	74				29				11.5	2 9	(50%)	93	RE

Quick sort is a very interesting sorting method and probably my favorite, it took a lot of work for this particular sort (33 lines) but once the final sort was received I was proud of it. There were many different explanations on how to go about conducting a quick sort online, but I mainly reference what was explained in the google drive

(https://drive.google.com/drive/folders/147oQRPpb6YFAwfBXz14i FKNRQfOclef) at around 27:00.