HW5 - hd2366 - Written

1). Weiss 7.19 array a:

QuickSort: 3 1 4 1 5 9 2 6 5 3 5

index 0 1 2 3 4 5 6 7 8 9 10

median of 3 9 5 is 5 ← pivot

Swap w/ first element

5 1 4 1 5 9 2 6 5 3 3

i → ← j

5 1 4 1 5 9 2 6 5 3 3

x x x i j

After swap 3 & 9

5 1 4 1 5 3 2 6 5 3 9

i → ← j

5 1 4 1 3 3 2 6 5 3 9

x x x i j

After Swap 3 & 6

5 1 4 1 5 3 2 3 5 6 9 (crossed)

i j

After Swap 5 & 5

(5 1 4 1 5 3 2 3) (5) (6 9)

less than cut off

Sort left Partion: 5 1 4 1 5 3 2 (3) (median of 5, 1, 3)

Swap w/ 1st element

3 1 4 1 5 3 2 5

i → ← j

3 1 4 1 5 3 2 5

i j

After Swapping 4 & 2

3 1 2 1 5 3 4 5

i → ← j

3 1 2 1 5 3 4 5 Cross

i j

Swap 3 & 1

(1 1 2) (3) (5 3 4)

Both partions meet cutoff => Insertion sort

(1 1 2) sorted

(5 3 4)

↳ 5 > 3

↳ 5 > 4 (3 4 5)

Combine to get her

1 1 2 3 3 4 5 5 6 9.

2) Weiss 7.23

No, because in an unsorted array, it's maybe as well picking a random number

If the number happen to be a max or a min, quicksort will take a long time

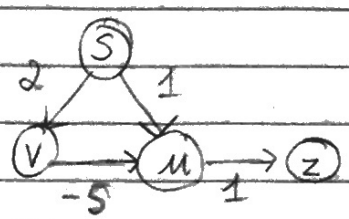
3) Weiss 7.28 a

```
int i = left;
int j = right - 1;
int leftPivot = 0
int rightPivot = 0

for (;;) {
    while (a[++i] < pivot) { }
    while (a[--j] > pivot) { }

    if (a[j] == pivot) {
        Swap (a[j], left + 1 + leftPivot);
    leftPivot++; }
```

```
If (a[i] == pivot) {
    Swap(a[i], right - a - rightPivot)
    rightPivot++; }
If (i<j) {      Swap (i, j) };
else {break;}
// return right element + pivot
For (int k = 0; k < rightPivot+1, k++) {
        Swap ( i+k, right -(k+1)), }
// return left element
For (int k = 0 , k < leftPivot ; k++){
        Swap ( j-k, left+(k+1)); }
```

5) Weiss 9.7a

Start at s



v	known	d_v	p_v
s	~~F~~ T	~~0~~	–
u	~~F~~ T	~~∞~~ ~~-3~~	s → minimum distance
v	~~F~~ T	~~∞~~ 2	s
z	~~F~~ T	~~∞~~ 2	u

Initial: $d(s) = 0$

$d(u) = 1$

① expand u ⇒ $d(z) = 2$

$d(v) = 2$

② expand v ⇒ $d(u) = -3$

⇒ Finally, $d(z) = 2$ when the actual shortest path is $d(z) = -2$

4/6) Weiss 9.1

	1	2	3	4	5	6	7	8	9	10	11
S	0	0	0	0	0	0	0	0	0	0	0
A	2	1	1	0	0	0	0	0	0	0	0
B	1	1	1	1	1	0	0	0	0	0	0
C	3	3	3	3	3	3	2	1	1	0	0
D	2	1	0	0	0	0	0	0	0	0	0
E	4	4	3	2	1	0	0	0	0	0	0
F	2	2	2	2	2	2	2	1	0	0	0
G	1	0	0	0	0	0	0	0	0	0	0
H	1	1	0	0	0	0	0	0	0	0	0
I	2	2	2	2	1	1	1	0	0	0	0
t	3	3	3	3	3	3	3	3	2	1	0
Enqueu	S	G	D, H	A		B, E		I	F	C	t
Dequeu	S	G	D	H	A	B	E	I	F	C	t

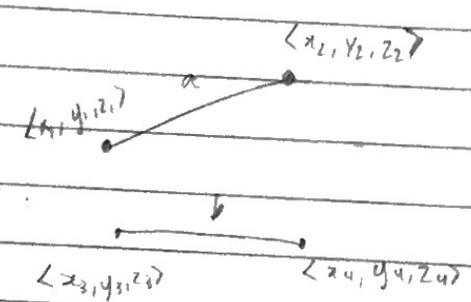
Final order:

S, G, D, H, A, B, E, I, F, C, t

6) Weiss 9.38.

a) Stick a coordinates $\langle x_1, y_1, z_1 \rangle$ and $\langle x_2, y_2, z_2 \rangle$

Stick b coordinates. $\langle x_3, y_3, z_3 \rangle$ $\langle x_4, y_4, z_4 \rangle$



Calculate vector range:

$\text{Range}(x_1, x_2) = R_{ax}$ $R(y_1, y_2) = R_{ay}$

$R(x_3, x_4) = R_{bx}$ $R(y_3, y_4) = R_{by}$

- If the intersection of the x ranges or y ranges of a & b are zero, the two sticks are unrelated
- Otherwise, we can calculate the cross-section of the two lines on the xy plane. The stick w/ higher Z-value is on top

6) Assuming the we know which stick are on top / below each other, we can pick up all the stick by using the graph algorithm

- Treat each stick as a vertex
- Arrow pointing from B → A indicates A is on top of B

- Use topological sort to order the stick in way it can be pick up one by one
- However, if there is a cycle in the graph, no vertex has indegree = 0 ⇒ can't pick up all the sticks