Sorting

2. Click the sequence of integers that represent the content of the array A=**[42, 8, 26, 35, 23, 32, 12, 39]** with n integers after 5 passes of this version of **Insertion** Sort.

Start A New Session

```
for (i = 1; i <= 5; i++) { // 5 passes
  e = A[i]; j = i;
  while (j > 0) {
    if (A[j-1] > e)
      A[j] = A[j-1];
    else
      break;
    j--;
  }
  A[j] = e;
}
```

Your answer is: **8 , 23 , 26 , 32 , 35 , 42 , 12 , 39**

You answered this question correctly! :) (Report Bug with Question)

First n+1 numbers will be sorted. Insertion sort is **STABLE**, so the rest of the elements is unaffected.

3. An array of 10 integers is being sorted in ascending order using **Quicksort**.
Suppose the algorithm has just finished the first pass of partitioning and pivot swapping thus changing the content of the original array into the following array:
[101, 104, 106, 109, 103, 106, 108, 115, 113, 107]
From the resulting array above, determine how many integers could have been the pivot?
Note: elements == pivot are partitioned to the right.

Start A New Session

1

You answered this question correctly! :) (Report Bug with Question)

Find number such that every number to the left is lesser than (or equal to) itself and every number to the right is greater than (or equal to) itself

3. An array of 10 integers is being sorted in ascending order using **Quicksort**.
Suppose the algorithm has just finished the first pass of partitioning and pivot swapping thus changing the content of the original array into the following array:
[101, 104, 106, 109, 103, 106, 108, 115, 113, 107]
From the resulting array above, determine how many integers could have been the pivot?
Note: elements == pivot are partitioned to the right.

Start A New Session

1

You answered this question correctly! :) (Report Bug with Question)

Find number such that every number to the left is lesser than (or equal to) itself and every number to the right is greater than (or equal to) itself

4. How many **swap(s)** is/are required to sort an array of n=6 integers: [16, 28, 9, 13, 26, 15] using this version of **Bubble** Sort?

Start A New Session

```
for (j = 0; j < n-1; j++)
  for (i = 0; i < n-j-1; i++)
    if (A[i] > A[i+1])
      swap(A[i], A[i+1]);
```

8

You answered this question correctly! :) (Report Bug with Question)

Brute force count swaps

5. Click the sequence of integers that represent the content of the array A=**[6, 38, 18, 22, 20, 19, 29, 34]** with n integers after 4 passes of this version of **Selection** Sort.

Start A New Session

```
for (i = 0; i < 4; i++) { // 4 passes
  cur_min = i;
  for (j = i+1; j < n; j++)
    if (A[j] < A[cur_min])
      cur_min = j;
  swap(A[i], A[cur_min]);
}
```

Your answer is: **6 , 18 , 19 , 20 , 38 , 22 , 29 , 34**

You answered this question wrongly.

The correct answer is: **6 , 18 , 19 , 20 , 22 , 38 , 29 , 34**

Selection sort is **UNSTABLE**. Keep track of where elements are swapped to.

11. Click the sequence of integers that represent the content of the array A=**[38, 4, 6, 5, 29, 34, 32, 26]** with n integers after 7 passes of this version of **Bubble** Sort.

Start A New Session

```
for (j = 0; j < 7; j++) // 7 passes
  for (i = 0; i < n-j-1; i++)
    if (A[i] > A[i+1])
      swap(A[i], A[i+1]);
```

Your answer is: **4 , 5 , 6 , 26 , 29 , 32 , 34 , 38**

You answered this question correctly! :) (Report Bug with Question)

Biggest 7 elements are bubbled right. So, this sequence of 8 integers will be fully sorted lol

12. How many **comparison(s)** is/are required to sort an array of n=5 integers: [15, 17, 14, 16, 13] using this version of **Insertion** Sort?

Start A New Session

```
for (i = 1; i < n; i++) {
  e = A[i]; j = i;
  while (j > 0) {
    if (A[j-1] > e) // each execution of this if-statement is counted as one comparison
      A[j] = A[j-1];
    else
      break;
    j--;
  }
  A[j] = e;
}
```

9

You answered this question correctly! :) (Report Bug with Question)

Become the algorithm

14. How many **pass(es)** is/are required to sort an array of n=5 integers: [35, 26, 37, 9, 7] using this version of **Bubble** Sort?

Start A New Session

```
j = 0;
do {
  swapped = false; j++;
  for (i = 0; i < n-j; i++) // each completion of this for-loop is counted as one pass
    if (A[i] > A[i+1]) { // each execution of this if-statement is counted as one comparison
      swap(A[i], A[i+1]);
      swapped = true;
    }
} while (swapped);
```

4

You answered this question wrongly.

The correct answer is: **5**

Become the algorithm. This is the “smart” version which breaks if no new swaps are made in the pass-through.

<u>Linked list</u>

2. In the pseudocode program below, `list` is an initially empty Singly Linked List.
The function `populateList()` adds the integers **[8, 7, 9, 8, 3, 5]** to the tail of `list` sequentially.
What is the output of the program?
Select 'No Answer' if the program results in an error.

Start A New Session

```
populateList();
int sum = 0;
Node n = list.head; // list.head/list.tail points to the first/last integer in list
n = list.head;
n = list.head;
n = n.next;
n = n.next;
sum += n.next.value;
n = n.next;
n = list.head;
sum += n.next.value;
print sum;
```

No answer

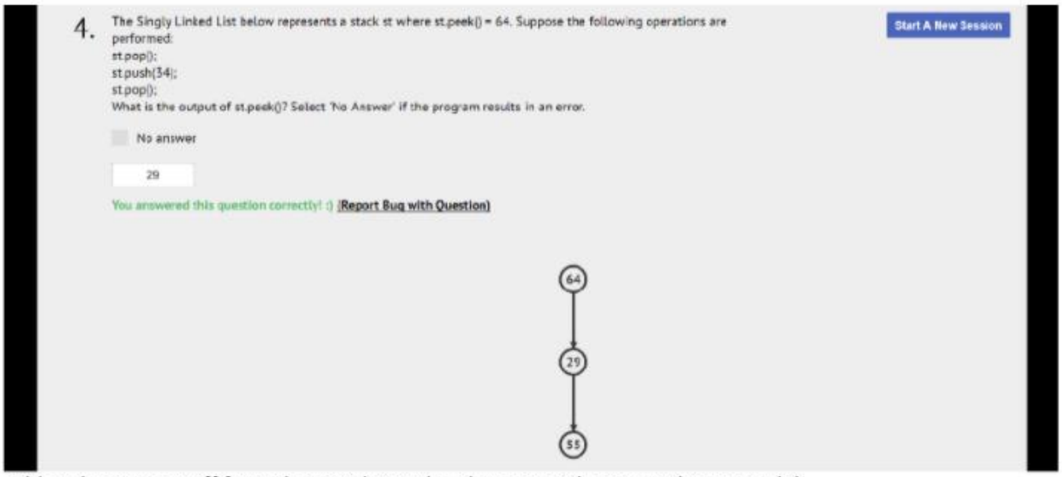
16

You answered this question wrongly.

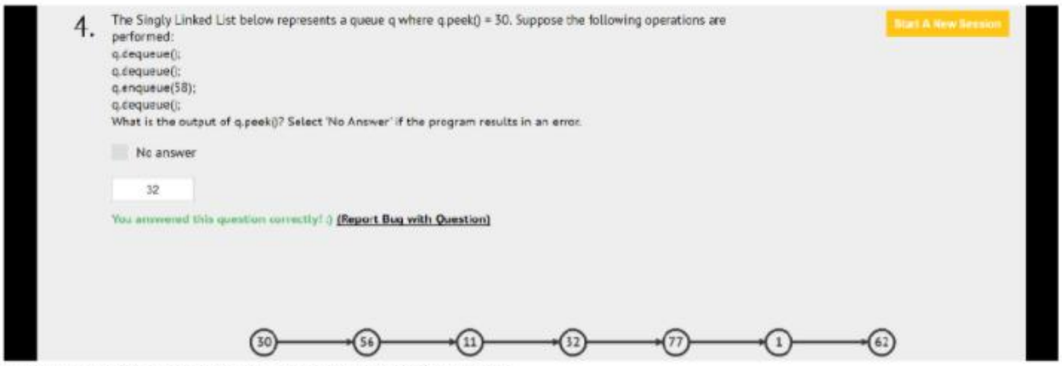
The correct answer is: **15** **(Report Bug with Question)**

Just be careful. Tail pointer is usually not updated. Final_element.next will lead to NULL (still valid), but attempting to NULL.next or dereference NULL will lead to compilation error -> NO ANSWER

Matthew's VA Cheat Sheet



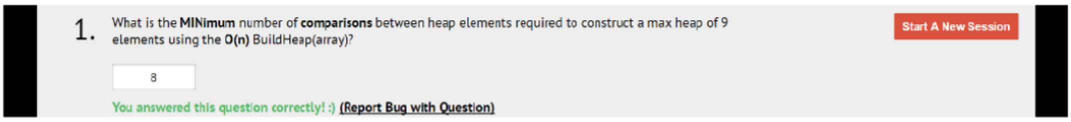
Add and remove stuff from the TOP (64 is the element at the top in this example)



Enqueue adds at the RIGHT, removes from the LEFT

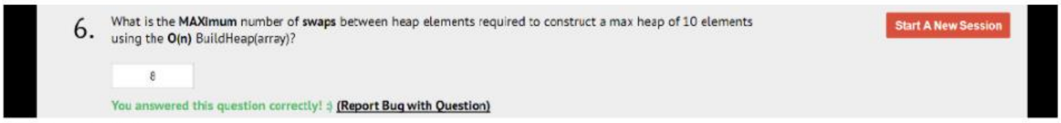
Binary Heap

Valid Max Heap: Make sure parent bigger than child

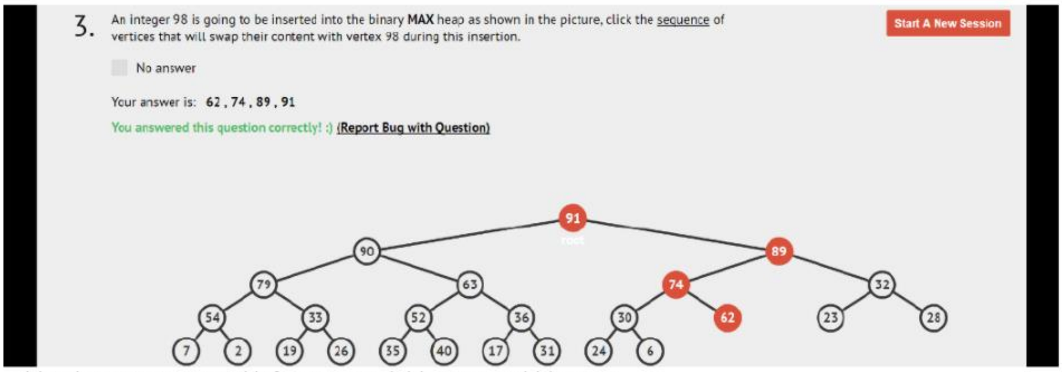


Refer to chart

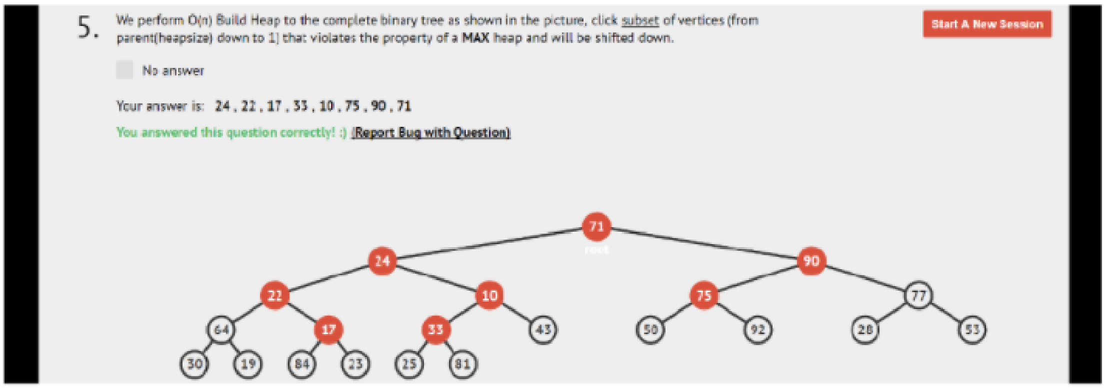
Refer to chart



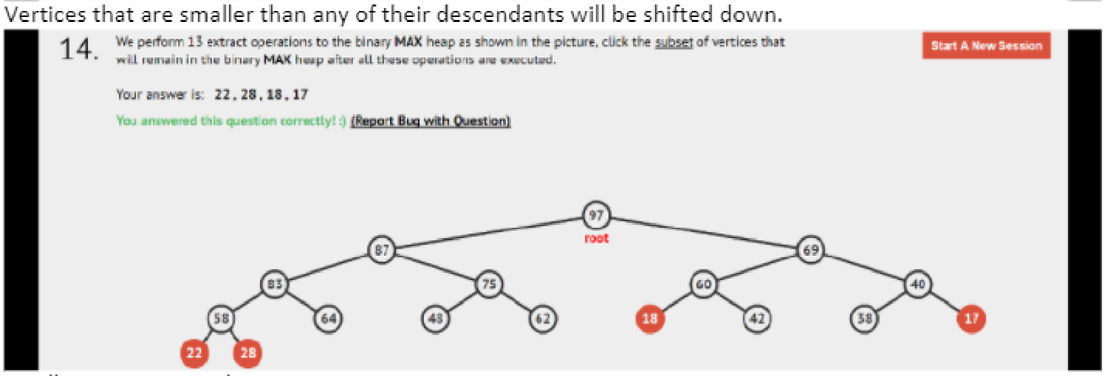
Refer to chart



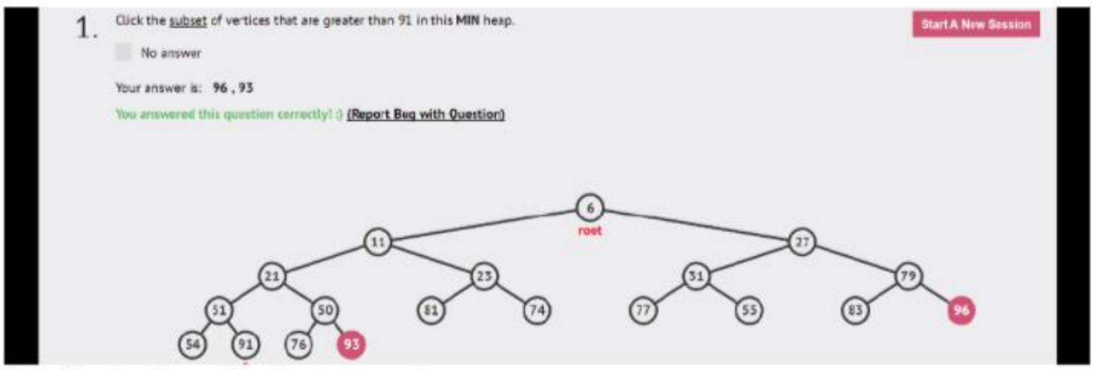
Add to bottom-most and left-most available spot. Bubble it up.



Vertices that are smaller than any of their descendants will be shifted down.



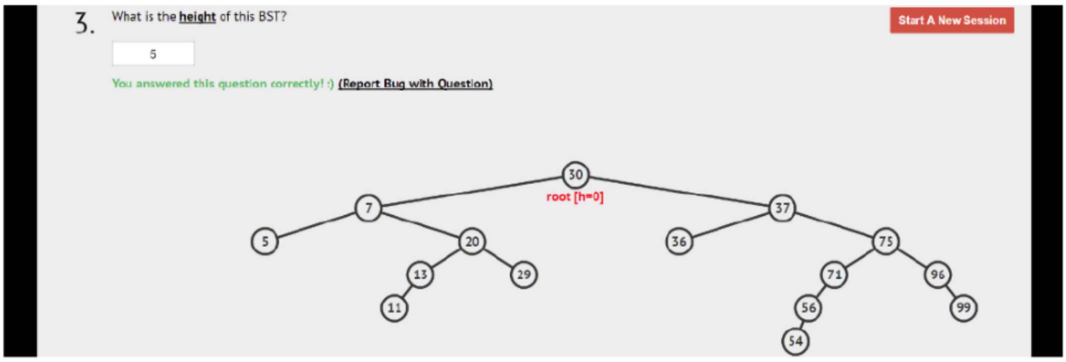
Smallest 17 – 13 = 4 elements are remaining



Find all vertex bigger than the query number

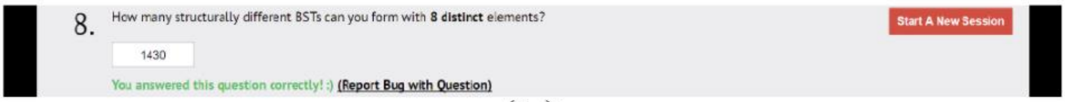
Binary Search Tree

BST: Left child < current < Right child.
Successor: Most direct way to the next element bigger than the node of interest
Predecessor: Most direct way to the next element smaller than the node of interest
Preorder: **Print first**, then check children
Inorder: Check left, **then print**, then check right
Postorder: Check children first, **then print.**

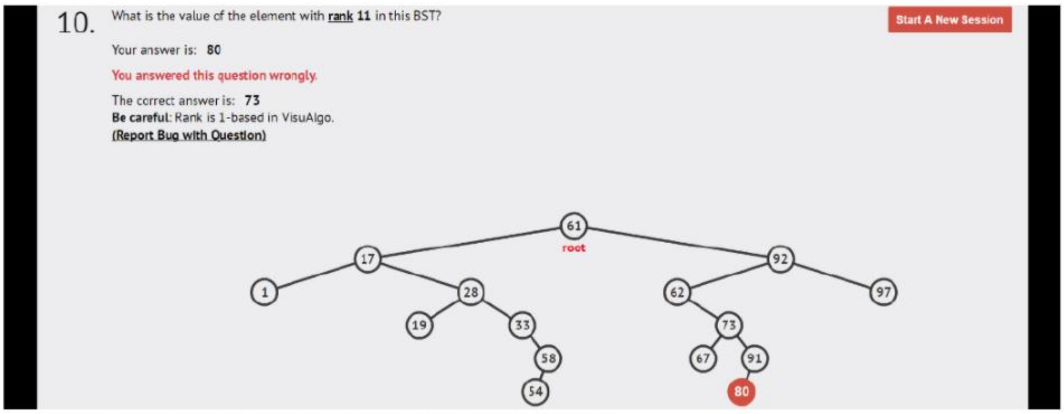


Largest number of edges from root to leaf

If target is bigger, next number should be bigger; if target smaller, next number should be smaller.

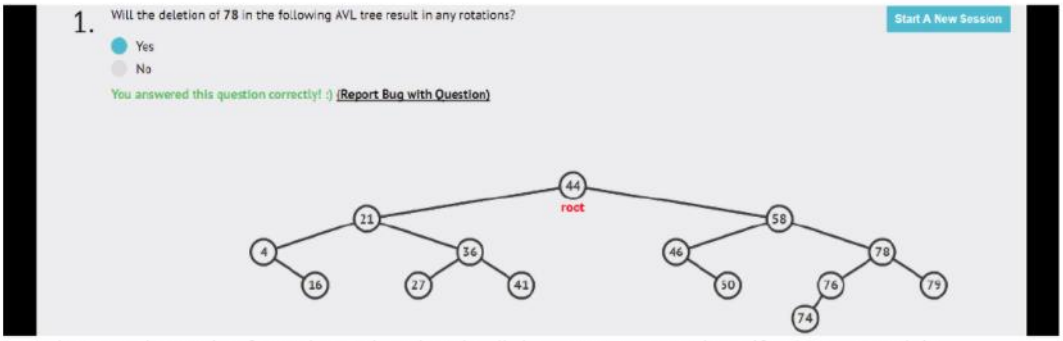


$$\frac{(2n)!}{(n+1)!\,n!}$$

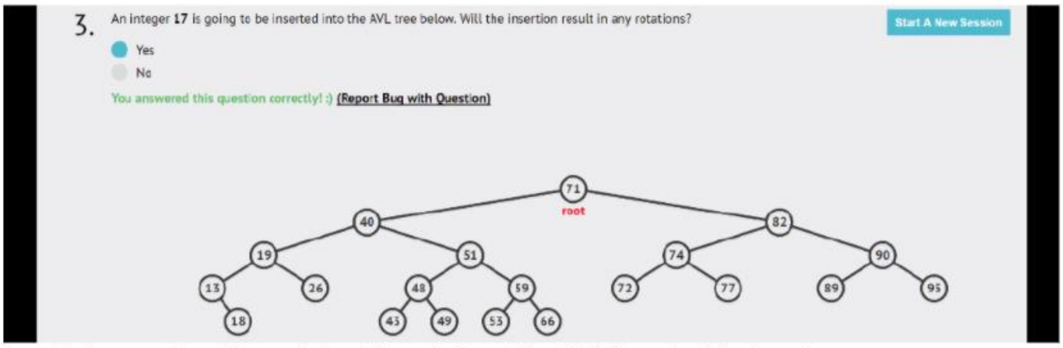


indexed-1. Leftmost is 1. Flatten the tree and count until rank.

AVL Tree

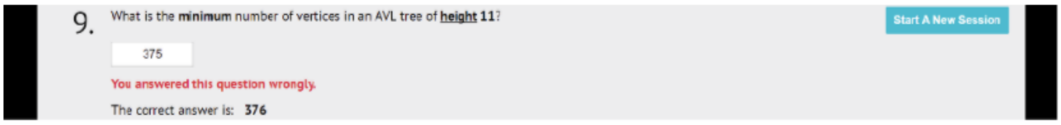


Search upwards starting from the replaced node all the way to root and see if anyone complains



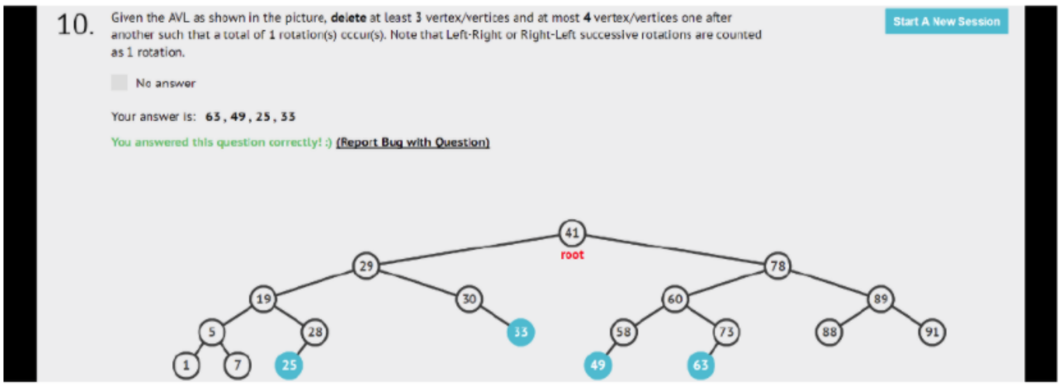
Insert into correct position and check for rotations. Max 1 "fix" required for insertion.

Matthew's VA Cheat Sheet



Height(n) = height(n-1) + height(n-2) + 1

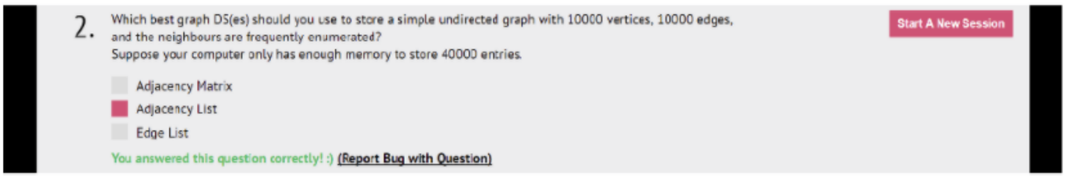
Height	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Vertices	1	2	4	7	12	20	33	54	88	143	232	376	609	986	1596	2583



Delete and check for rotation.

Graph DS

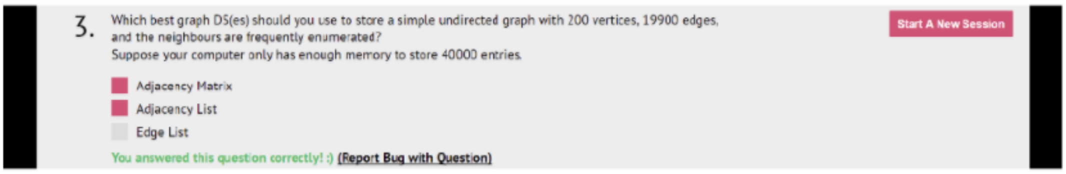
Directed acyclic, just make sure small number -> bigger number.
Tree: V vertices, V-1 edges, acyclic.
Complete graph: An edge between any pair of vertices.
Connected graph: From any vertex, can get to all other vertex


2. Which best graph DS(es) should you use to store a simple undirected graph with 10000 vertices, 10000 edges, and the neighbours are frequently enumerated?
Suppose your computer only has enough memory to store 40000 entries.

- Adjacency Matrix
- Adjacency List
- Edge List

You answered this question correctly! :) (Report Bug with Question)

Need to enumerate neighbours, **BUT NOT ENOUGH MEMORY (we need V^2 entries for AM)**

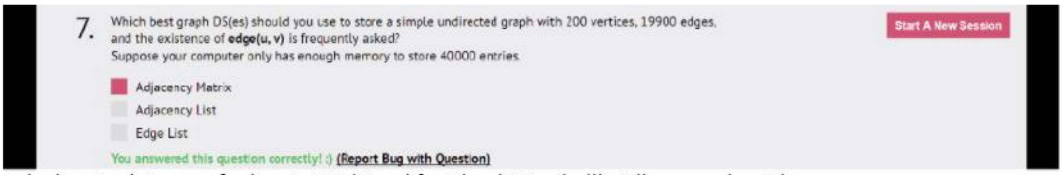

3. Which best graph DS(es) should you use to store a simple undirected graph with 200 vertices, 19900 edges, and the neighbours are frequently enumerated?
Suppose your computer only has enough memory to store 40000 entries.

- Adjacency Matrix
- Adjacency List
- Edge List

You answered this question correctly! :) (Report Bug with Question)

Adjacency Matrix enumerate not as fast but still accepted, **so it depends on memory.**

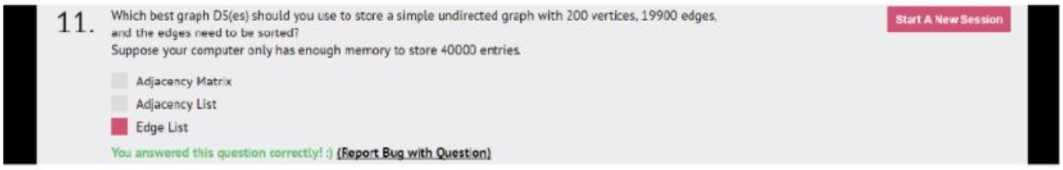
Matthew's VA Cheat Sheet


7. Which best graph DS(es) should you use to store a simple undirected graph with 200 vertices, 19900 edges, and the existence of **edge(u, v)** is frequently asked?
Suppose your computer only has enough memory to store 40000 entries.

- Adjacency Matrix
- Adjacency List
- Edge List

You answered this question correctly! :) (Report Bug with Question)

Ask about existence of edge -> AM (Good for Floyd-Warshall's 4 line wonder :D)

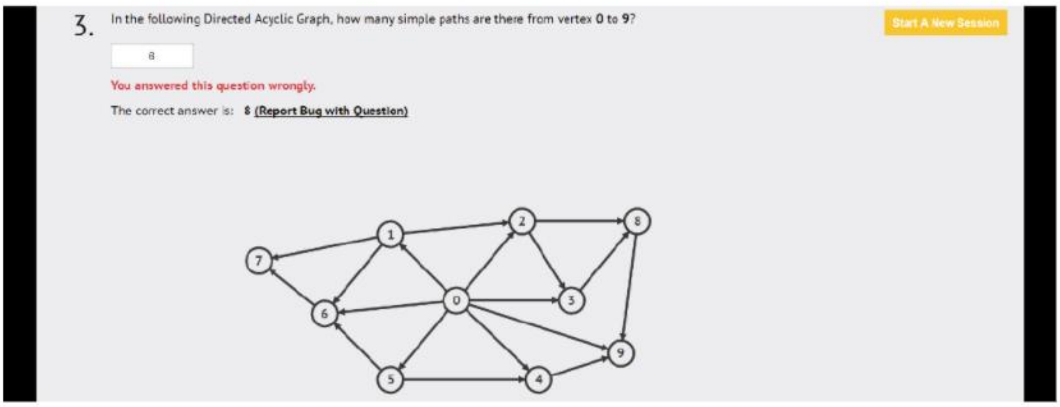

11. Which best graph DS(es) should you use to store a simple undirected graph with 200 vertices, 19900 edges, and the edges need to be sorted?
Suppose your computer only has enough memory to store 40000 entries.

- Adjacency Matrix
- Adjacency List
- Edge List

You answered this question correctly! :) (Report Bug with Question)

Edges in an edge list can be easily sorted by weight

Graph traversal

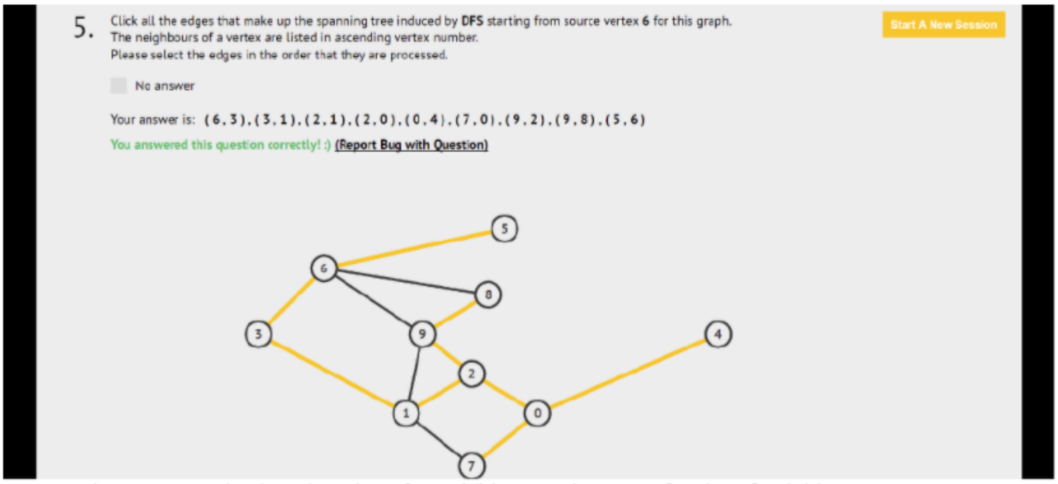

3. In the following Directed Acyclic Graph, how many simple paths are there from vertex 0 to 9?

8

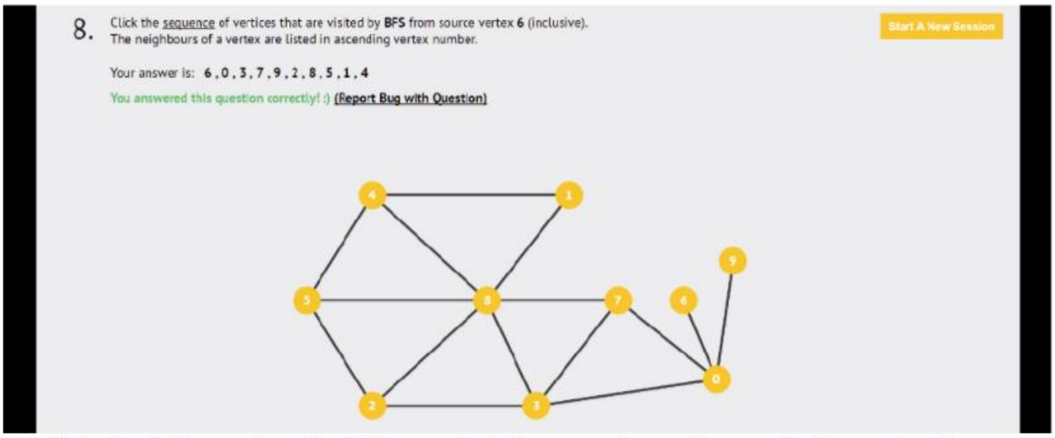
You answered this question wrongly.

The correct answer is: 8 (Report Bug with Question)

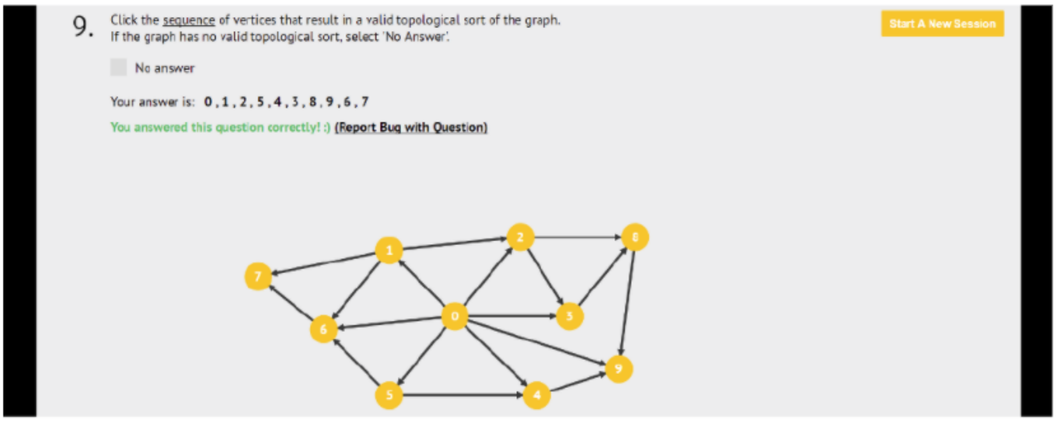
Write down all the paths



DFS: Go deep, recurse back and go deep for neighbour. Take note of order of neighbours.



BFS: Visit all neighbours, then all neighbours of neighbours, and so on. Keep track of the order of exploration of neighbours



Topological sort

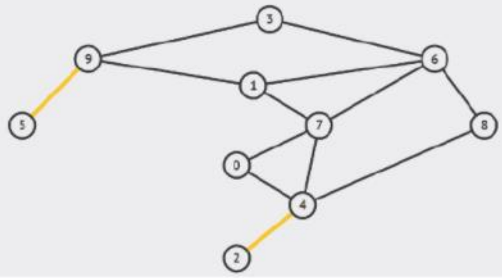
10. Click all the edges that must belong to every spanning tree of the graph shown below.

Start A New Session

No answer

Your answer is: **(9,5),(2,4)**

You answered this question correctly! :) **(Report Bug with Question)**



Vertex with only 1 edge

11. How many different spanning trees are there in a complete graph with **8** vertices?

Start A New Session

262144

You answered this question correctly! :) **(Report Bug with Question)**

$$n^{n-2}$$

13. Consider the following partial pseudocode for DFS:

```
DFS(u)
  visited[u] = true;
  for all v adjacent to u
    if (!visited[v]) { DFS(v); }
  print(u);
```

Suppose this code is run on the following BST starting from the root, click the sequence of vertices in the order that they are printed. Assume that the left child is always visited before the right child.

Start A New Session

Your answer is: **1,21,27,68,74,67,18**

You answered this question correctly! :) **(Report Bug with Question)**

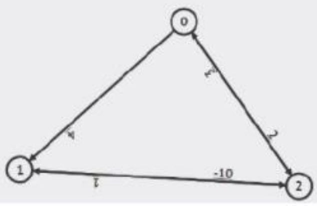
Print comes after exploring all neighbours -> Post-order traversal

SSSP

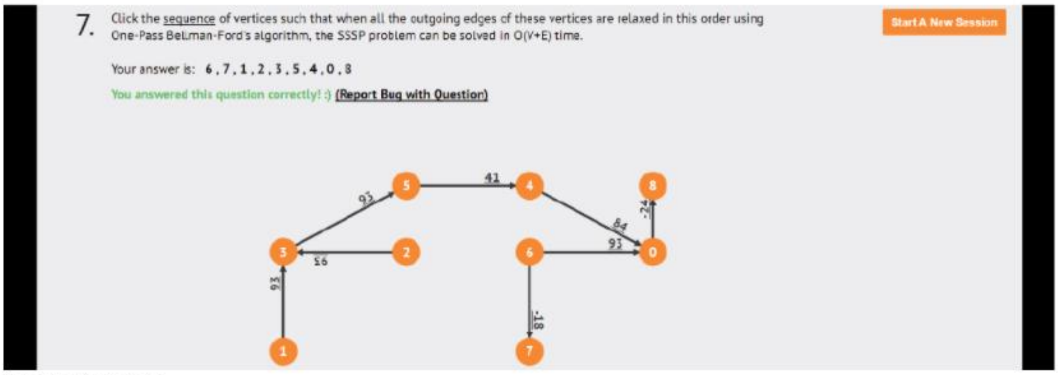
1. Draw a simple connected weighted directed graph with **3** vertices and **5** directed edges such that running **Modified Dijkstra's algorithm** from source vertex **0** makes the algorithm run indefinitely.
The weights of the edges have to be distinct.

Start A New Session

You answered this question correctly! :) **(Report Bug with Question)**

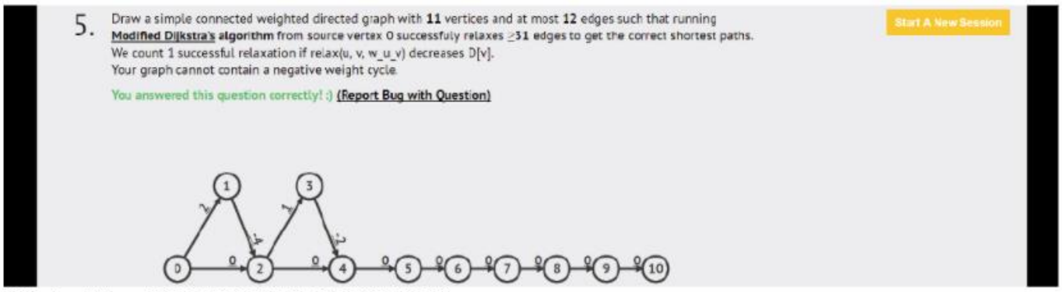


Create a negative cycle. **MAKE SURE THE WEIGHTS ARE DISTINCT**

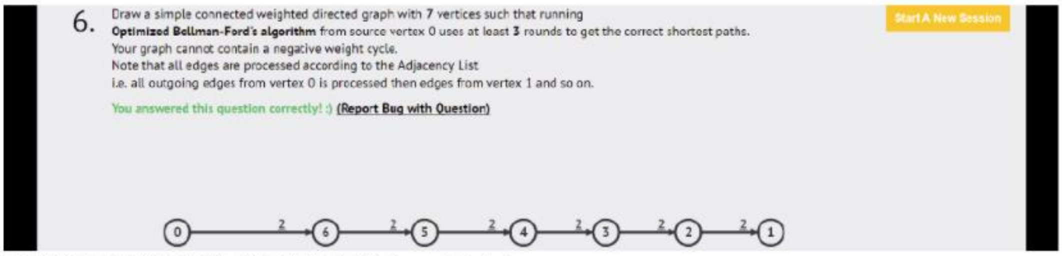


Topological sort

Matthew's VA Cheat Sheet



Dijkstra Killer. **THERE ARE (E-V+1) TRIANGLES.**



Bellman-Ford Killer. 0 -> (N-1) -> (N-2) -> … -> 2 -> 1

SSSP- Path Weight Criteria

	Bellman-Ford	Original Dijkstra	Modified Dijkstra
Terminate	Always	Always	Does not terminate when there's negative weight cycle
Wrong when there's	Negative eight cycle	Negative edges and Negative weight cycle	Negative weight cycle- it doesn't even terminate in the first place ☹

Note:

If the graph contains negative edges but does not contain any negative cycles, Bellman Ford and modified Dijkstra will give the correct answer all the time.

Original Dijkstra *could* still give the correct answer (run Bellman Ford and original Dijkstra and compare the answers)

Tables

Max Heap max swaps

Elements	6	7	8	9	10	11	12	13	14
Swaps	4	4	7	7	8	8	10	10	11

Max Heap max comparisons

Elements	6	7	8	9	10	11	12	13	14
Compare	7	8	11	14	15	16	18	20	21

Max Heap Min Comparisons: N-1

Elements	6	7	8	9	10	11	12	13	14
Compare	5	6	7	8	9	10	11	12	13

Binary Search Tree: how many permutations

Elements	1	2	3	4	5	6	7	8	9
Catalan #	1	2	5	14	42	132	429	1430	4862

AVL Tree: Minimum # of vertices (Height(n) = height(n-1) + height(n-2) + 1)

Height	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Vertices	1	2	4	7	12	20	33	54	88	143	232	376	609	986	1596	2583

Graph Structures

Neighbours Frequently Enumerated AND **HAS ENOUGH MEMORY: AL + AM** (We need V^2 memory for AM)

Neighbours Frequently Enumerated AND **NOT ENOUGH MEMORY: AL ONLY** (memory < V^2)

Existence of edge(u,v): **AM ONLY**

Edges need to be sorted: **EDGE LIST ONLY**

Number of spanning trees in complete graph n^(n-2)

Vertices	3	4	5	6	7	8	9	10	11
Ans	3	16	625	1296	16807	262144	4782969	100000000	2357947691

Path Weight Criteria

	Bellman-Ford	Original Dijkstra	Modified Dijkstra
Terminate	Always	Always	Does not terminate when there's negative weight cycle
Wrong when there's	Negative weight cycle	Negative weight (might still be correct if the negative edge doesn't mess up any of the nodes)	Negative weight cycle- it doesn't even terminate in the first place ☹

Modulo Tables for Hashing

%11

0	1	2	3	4	5	6	7	8	9	10
0	1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20	21
22	23	24	25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40	41	42	43
44	45	46	47	48	49	50	51	52	53	54
55	56	57	58	59	60	61	62	63	64	65
66	67	68	69	70	71	72	73	74	75	76
77	78	79	80	81	82	83	84	85	86	87
88	89	90	91	92	93	94	95	96	97	98
99	100	101	102	103	104	105	106	107	108	109

%13

0	1	2	3	4	5	6	7	8	9	10	11	12
0	1	2	3	4	5	6	7	8	9	10	11	12
13	14	15	16	17	18	19	20	21	22	23	24	25
26	27	28	29	30	31	32	33	34	35	36	37	38
39	40	41	42	43	44	45	46	47	48	49	50	51
52	53	54	55	56	57	58	59	60	61	62	63	64
65	66	67	68	69	70	71	72	73	74	75	76	77
78	79	80	81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100	101	102	103
104	105	106	107	108	109	110	111	112	113	114	115	116

%12

0	1	2	3	4	5	6	7	8	9	10	11
0	1	2	3	4	5	6	7	8	9	10	11
12	13	14	15	16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31	32	33	34	35
36	37	38	39	40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55	56	57	58	59
60	61	62	63	64	65	66	67	68	69	70	71
72	73	74	75	76	77	78	79	80	81	82	83
84	85	86	87	88	89	90	91	92	93	94	95
96	97	98	99	100	101	102	103	104	105	106	107
108	109	110	111	112	113	114	115	116	117	118	119

%14

0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	1	2	3	4	5	6	7	8	9	10	11	12	13
14	15	16	17	18	19	20	21	22	23	24	25	26	27
28	29	30	31	32	33	34	35	36	37	38	39	40	41
42	43	44	45	46	47	48	49	50	51	52	53	54	55
56	57	58	59	60	61	62	63	64	65	66	67	68	69
70	71	72	73	74	75	76	77	78	79	80	81	82	83
84	85	86	87	88	89	90	91	92	93	94	95	96	97
98	99	100	101	102	103	104	105	106	107	108	109	110	111

Kattis the Cat for good luck