AVL trees:

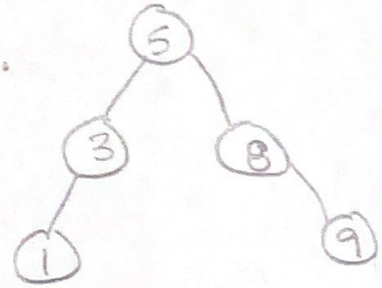
① insert the following numbers, in order, into a Binary Search Tree & AVL Tree

tree 1 : [10, 15, 18, 5, 7, 21, -3]

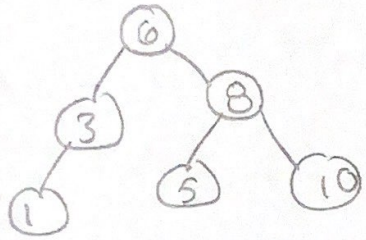
tree 2: [1, 2, 3, 4, 5, 6]

② Are the following trees valid AVL trees?

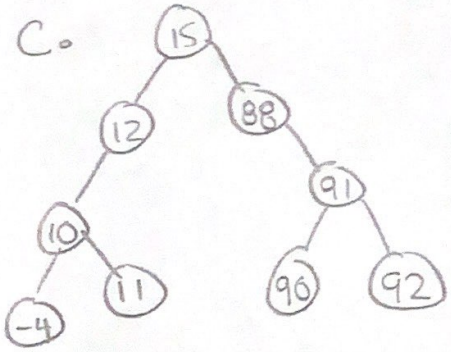
a.



b.

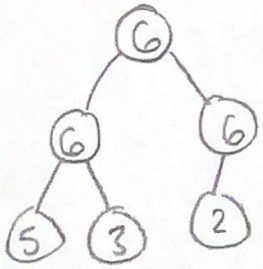


c.

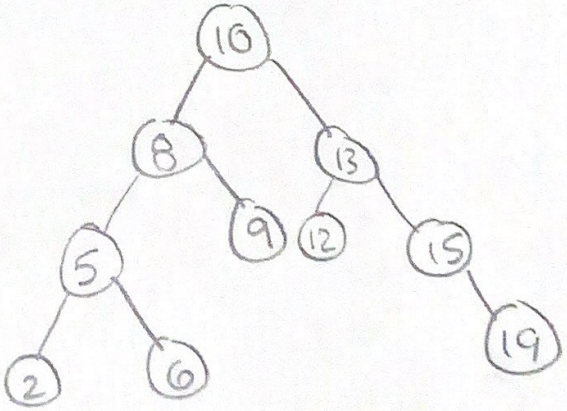


d. ∅

e.

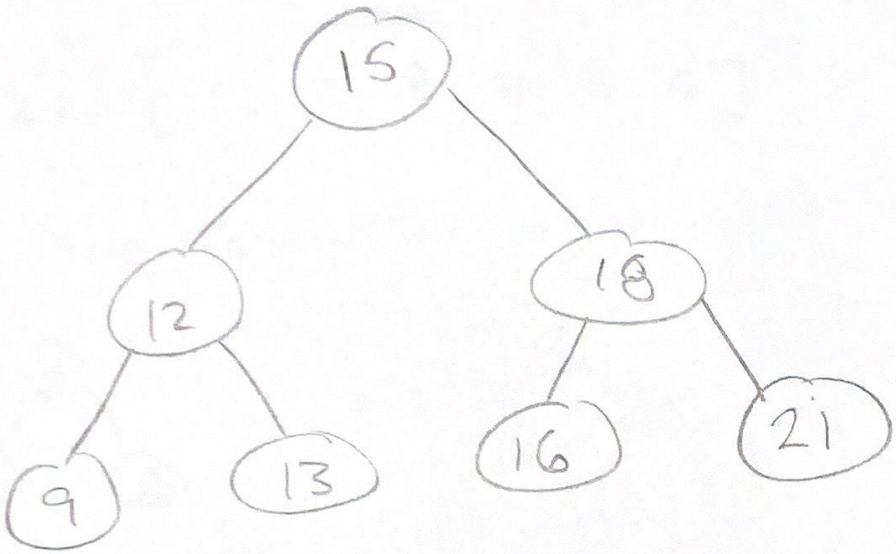


f.



AVL trees (page 2):

remove from the following tree, in order: [21, 18, 16, 9]



Runtime Questions:

* For all, describe the tree that gives you the worst case runtime

	Binary Search Tree	AVL Tree
insert		
delete		
find		
max in tree		
min in tree		

Heaps

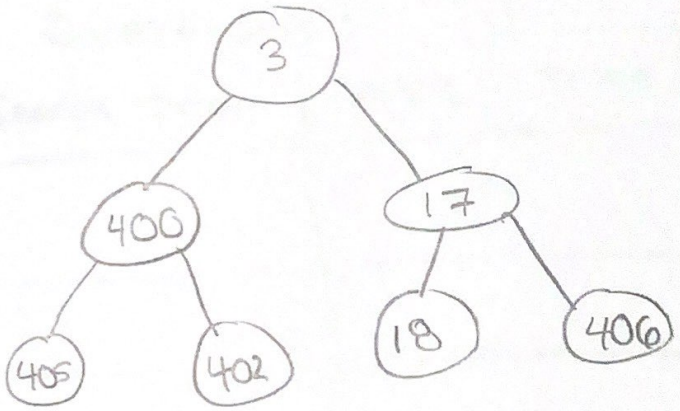
Runtime:	(min heap with n elems)
remove min	
insert	
build heap	

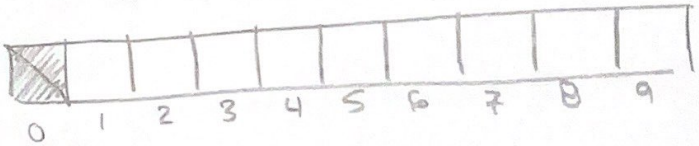
(1)

(2) what are the heap invariants?

(3) When we store heaps in an array, how do we find the parent of a node at index i? How do we find the children?

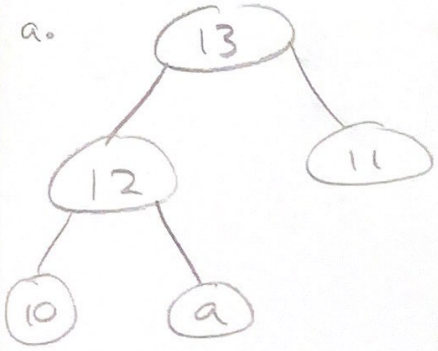
(4) Convert the following min heap tree into an array. (leave index 0 empty! (why?))



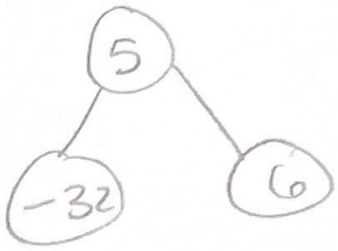


Are the following valid MAX heaps? (heaps page 2)

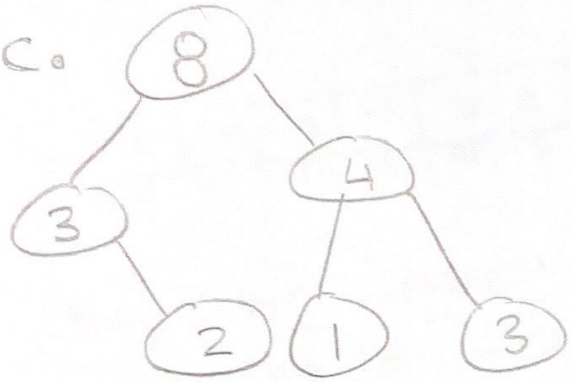
a.



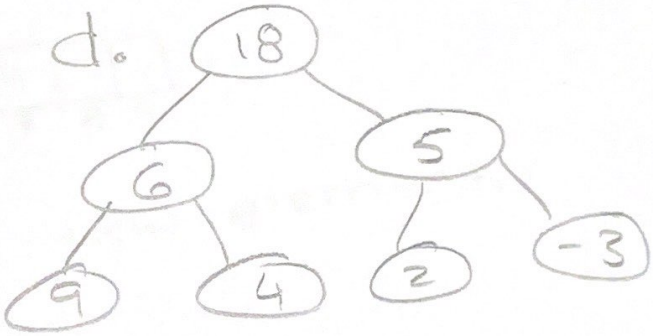
b.



c.



d.



e.

X	30	15	6	2
0	1	2	3	4

f.

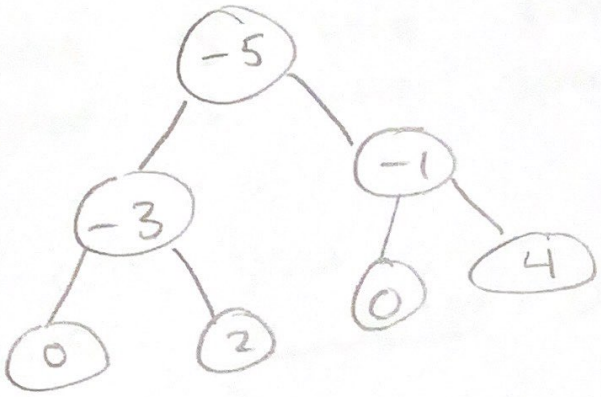
X	90	100	5
0	1	2	3

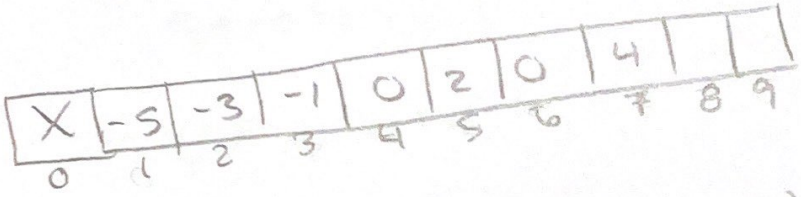
g.

X	30	15	18	12	3	7	14	13			
0	1	2	3	4	5	6	7	8	9	10	11

Heaps (more...)

Consider this min heap.





Say we insert a new element.
value = -25.

(1) where in tree / array does this value initially go? what invariant are we fixing?

(2) After step 1, what invariant do we need to fix? How do we do that?

(3) try removing the new root.
repeat steps (1-2).

(4) what is the runtime?

Heap sort

① Describe how to sort a heap, stored in an array.
- what is the time complexity?
- what is the space complexity?

② How can you sort an unordered vector in place using a heap sort strategy?
- complexity?

---

heap application

① why is a heap used in Dijkstra's algorithm?

② what is another application of a priority queue (heap)?

Hash tables

① what is a hash function?

② what makes a good hash function?

③ Describe the sequence of events from getting a key to insertion into an empty table of size 10.

④ what is the worst case runtime of insertion into a hash table using chaining (linked lists)? Describe the table that causes this runtime.

⑤ what is the worst case runtime of insertion into a hash table using linear probing (a form of open addressing)? Describe the hash table that causes this runtime.

successful)

⑥ what is the (best case) desired runtime of insertion, deletion, and search. Describe, for both types of collision resolution, what the table looks like, and what the runtime is.

⑦ when & why should we expand a hash table?

Hash tables (page 2)

① Let the hash function, $h(key) = key$ where key is a positive integer.

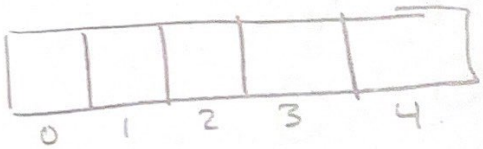
Ex: key = 35, $h(key) = 35$.

Insert the following numbers into a table, in order:

[5, 6, 13, 10, 15, 20]

---

table 1: use chaining. Expand when the load factor $\left(\frac{n}{m}\right) = \left(\frac{\text{num of elements}}{\text{num of buckets}}\right) > 0.5$

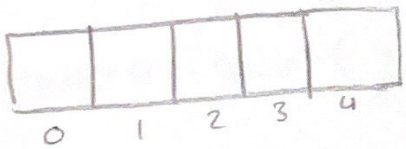


m=5

Draw the resulting table.

---

table 2: use linear probing. expand when the load factor $\left(\frac{n}{m}\right) > 0.75$



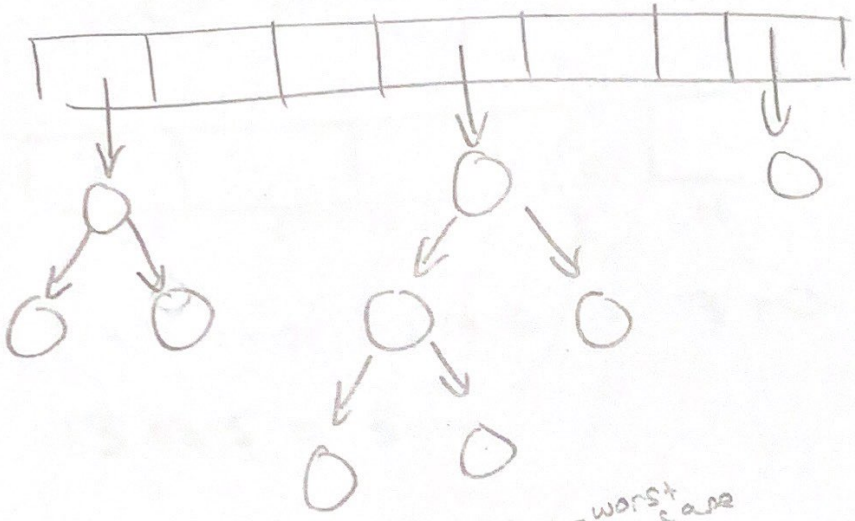
m=5

---

② Why do we need to re-hash when we expand?

Hash tables (bonus questions) (A)

Consider a hash table that uses a more advanced form of chaining. Instead of linked lists, it stores an AV based on the key.



What is the new worst case runtime of the following queries:

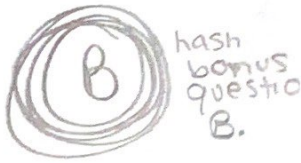
```
insert (key)
search (key)
delete (key)

deleteTable()
```

why?

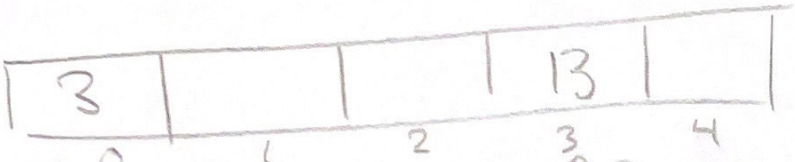
why don't we always use this approach?

Consider the hash table, which uses a modified linear probing collision resolution approach.



$$h(key, i) = key + 2i$$

i = attempt number − 1



m = 5

insert (13) → h(13, 0) = 13 + 0 = 13

13 % 5 = 3.

insert (3) → h(3, 0) = 3 + 0 = 3

3 % 5 = 3

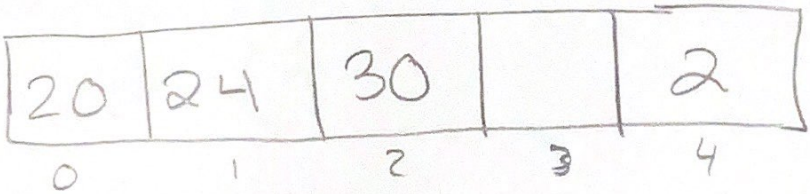
Collision!!!

→ h(3, 1) = 3 + 2 = 5

5 % 5 = 0

Consider the following table. What order could the numbers have been inserted in? Given the hash function above.

$h(key, i) = key + 2i$



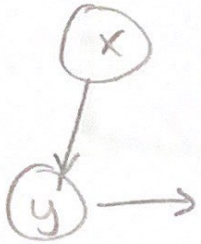
m = 5

Graphs

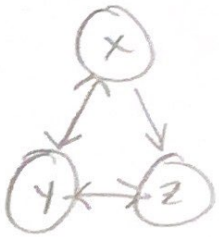
① Are the following valid graphs?

a.

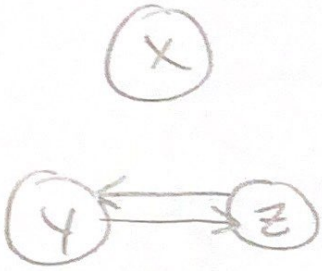
b.



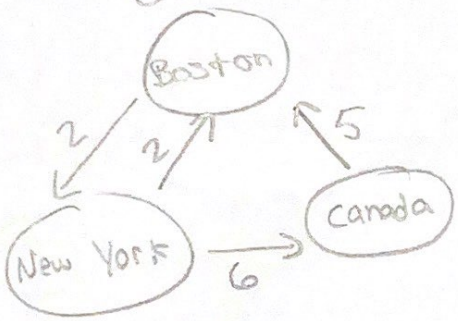
c.



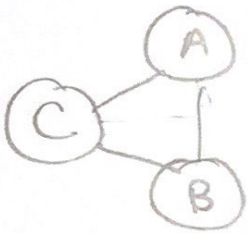
d.



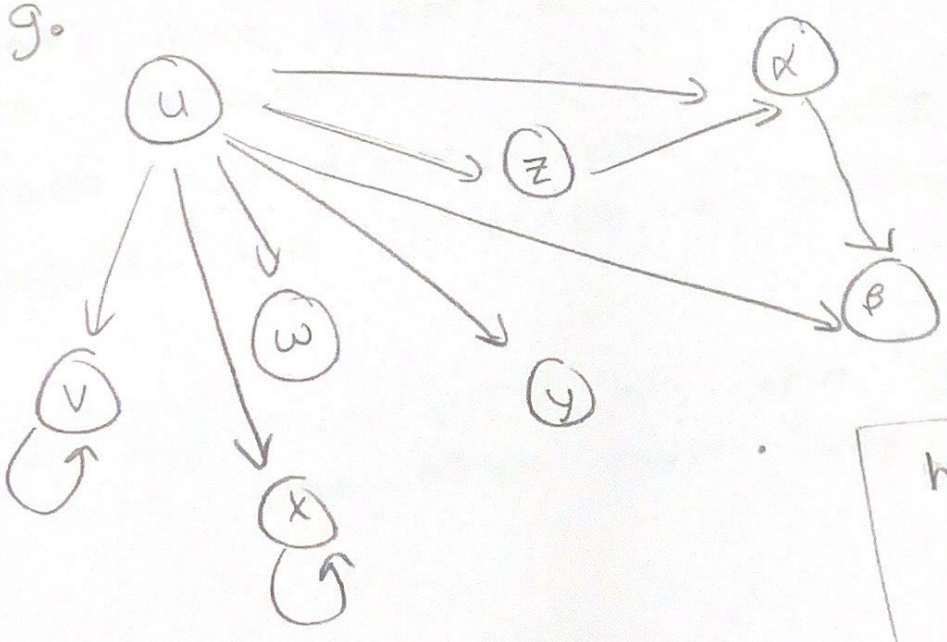
e.



f.



g.

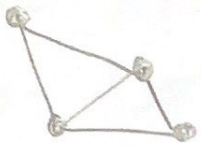


h. (no nodes)

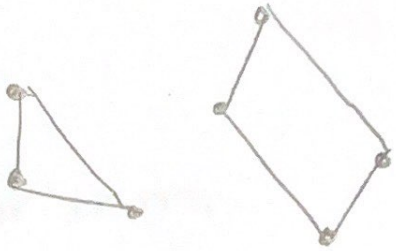
Graphs (page 2)

① Is this graph connected?

a.

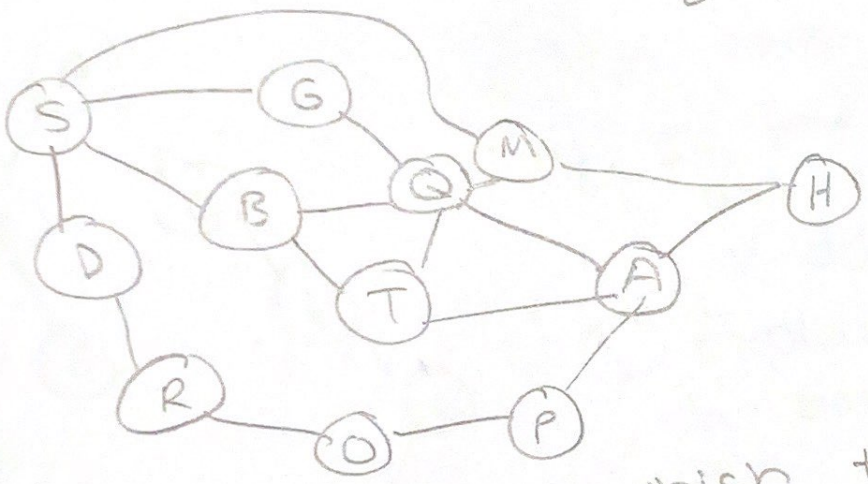


b.



② BFS

starting at (S), find the shortest path from (S) to (A). report the distance & path.



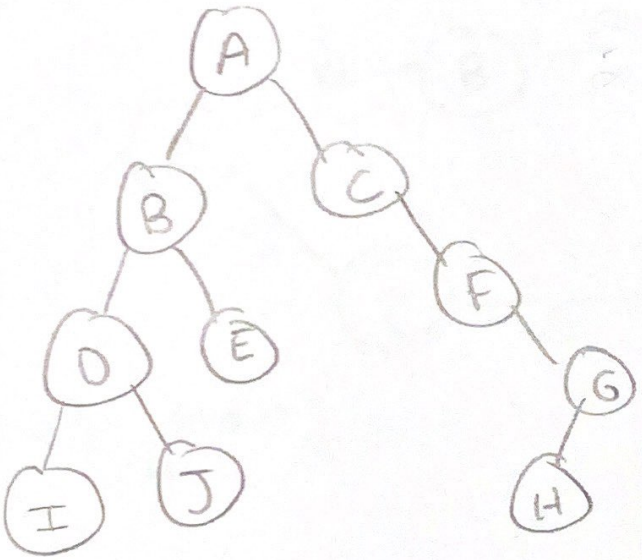
- state the order in which the verticies are explored
- track the distance, and parents
- report the distance & path.

③ DFS: run DFS on the graph. report the first path you see.

Graphs (page 2)

1. What is the runtime of...
   - a- BFS
   - b- DFS
   - c- Dijkstra's algorithm.

   why?
2. Run a in-order traversal on the following tree.



3. is this a graph?
4. Is the search we did a DFS, BFS, or dijkstra's algorithm
5. Explain how to do both DFS & BFS on a tree.

---

6. Can a BFS report the best (shortest) path from the source to a vertex when.

   1. Graph is not connected. ——— ↓ the rest are connected.
   2. Graph has weighted edges
   3. Graph has unweighted edges
   4. Graph has cycles (A→B→C→A→B→C→A...)

hard 5. Graph has n nodes and $\frac{n}{2}$ edges.

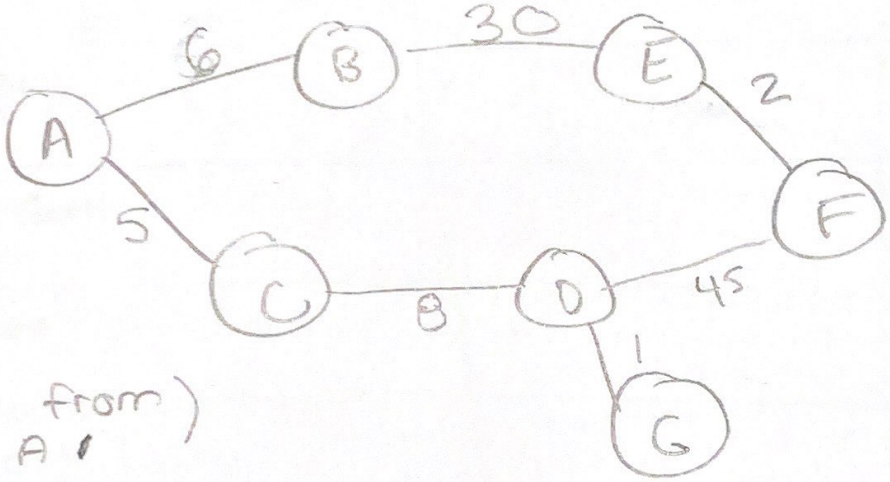
Graphs (page 4)

① Describe the runtimes

sorry about the line...

	unweighted, adjacency list	unweighted, adjacency matrix	weighted adj. list	weighted matrix
DFS				
Dijkstra's				
BFS				

② Complete Dijkstra's algorithm on...



(start from A)

2a- Fill in the complete table from lecture

2b- what is the fastest path from A → F

☆ Probably on the exam *

Sorting

① Fill in the table

also, can the algorithm be done in place?

	Best runtime	worst runtime.	example input for each (best & worst)
QuickSort			
Merge Sort			
Insertion Sort			
Selection Sort			
Counting Sort			
Radix Sort			
heap sort			

② Which sorts are best for the following situations?

ⓐ - a completly unsorted array, nothing known about the numbers.

ⓑ - numbers in range $[-n, n]$ (1 or 2 wrong)

ⓒ - numbers in range $[-n^n, n^n]$ ↙

ⓓ - an almost completly sorted array

ⓔ - a reverse sorted array.