2\. a) 1. Index of $P_{i,j} = \frac{i(i+1)}{2} + j$

2\. Index of Left child of $P_{i,j}$: $\frac{(i+1)(i+2)}{2} + j$

Index of Right " : $\frac{(i+1)(i+2)}{2} + (j+1)$

3\. Index of Left parent of $P_{i,j}$: $\frac{(i-1)i}{2} + (j-1)$

Index of Right " : $\frac{(i-1)i}{2} + j$

4\. ~~Left~~ Index of left child of A[i] :

" right " :

5\. Index of left parent of A[i] :

" right " :

b) Lower ~~[illegible]~~ bound : $\frac{(l+1)l}{2} + 1$

Upper ~~[illegible]~~ bound : $\frac{(l+1)l}{2} + (l+1)$

~~c) deleteMax(A)~~

c) delete-Max(A):

A: an array-based pyramid

```
max ← A[0]
swap(A[0], A[size(A)-1])
size(A) ← size(A) - 1
bubble-down(A, 0)                // from M2 Slide 8
return max
```

d) Swapping first with last element, and then removing the last element maintains the structural property of the pyramid.

Bubble-down ~~foresees~~ fixes the ordering of elements after the swap, maintaining the ordering property.

Therefore the resulting tree is a pyramid.

From part (b), we saw:

$$\frac{(l+1)l}{2} + 1 \le n \le \frac{(l+1)(l+2)}{2}$$

$$\Rightarrow l^2 + l + 2 \le 2n$$

$$\Rightarrow l \le \sqrt{2n-2-l} \qquad [l \ge 0]$$

$$\Rightarrow l \in O(\sqrt{n})$$

Also, bubble-down() does l swaps at a maximum. Rest of the operations inside delete Max() are constant time.

$\therefore$ the algorithm has time complexity $O(\sqrt{n})$.

[Also, $2n \le l^2 + 3l + 2$

$$\Rightarrow \sqrt{2n-3l-2} \le l$$

$$\therefore l \in \Theta(\sqrt{n})$$]

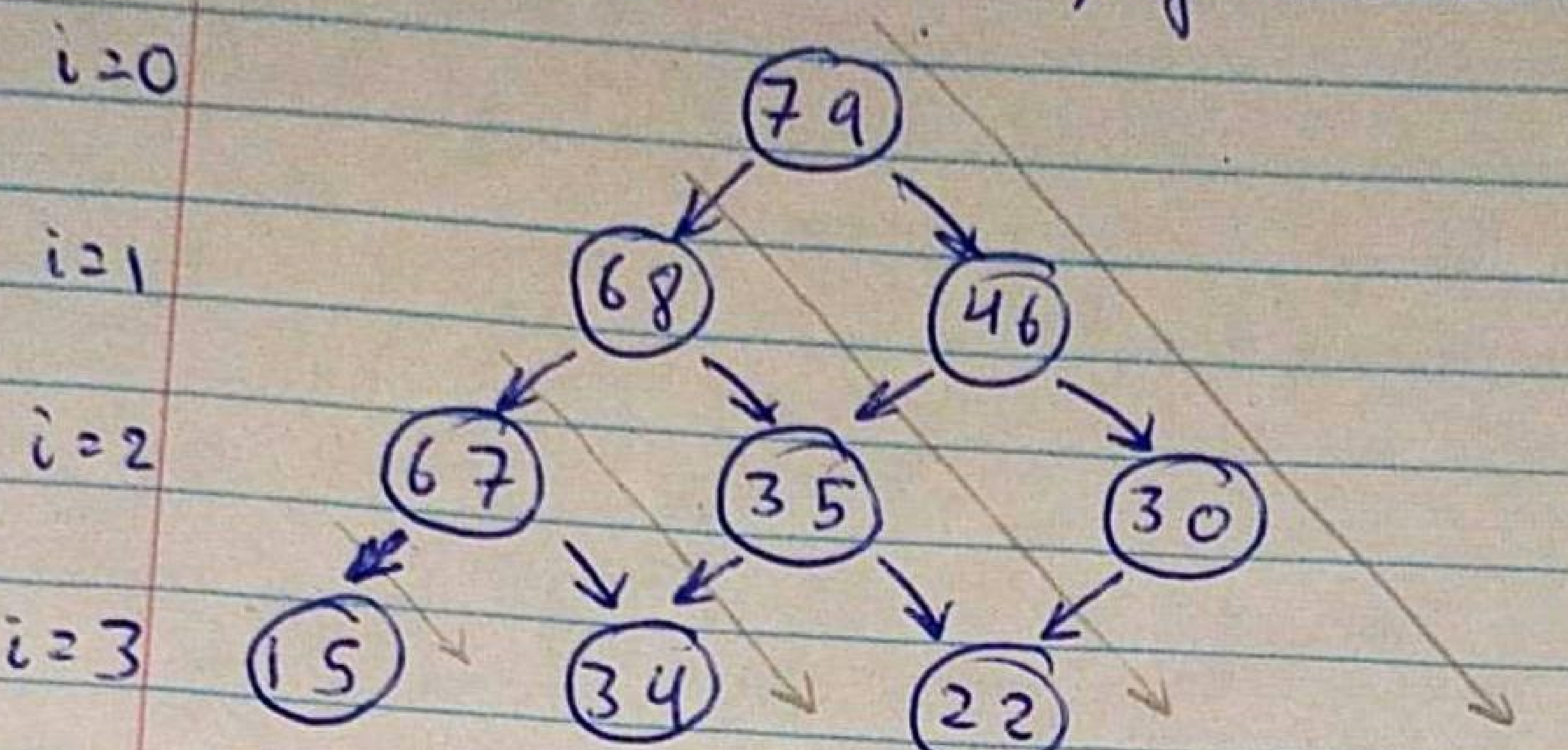
d) insert (A, x)
A: an array-based pyramid
x: a new item
1. size(A) ← size(A) + 1
2. A[size(A) − 1] ← x
3. bubble-up (A, size(A) − 1) // from M2 slide 7

Adding the element at the end of the array would mean that we are either filling up a level or creating a new level with the element on the left-most side of the pyramid. This is consistent with the structural property.

bubble-up fixes the ordering of the pyramid ∴ the resulting tree is a pyramid.

Following from part c, bubble-up also does l ~~parts~~ swaps at a maximum.
∴ algorithm has the time complexity $O(\sqrt{n})$.

e) Consider the pyramid:



Note that the elements following the arrows drawn in pencil are sorted. This means it is possible to perform binary search on these individual lists.

Let the last level be donated by L. $L = 3$ in this case. We have $L+1$ sorted lists.

```
contains (A, x)
A : an array-based pyramid
x : element to be searched
1. for i ← 0   to L do
2.      elems ← [x : elements in sorted list #i]
3.      if(binarySearch (elems))
4.           return True
5. return False
```

binarySearch() complexity is $\Theta(\log n)$.
From part (c), $L \in \Theta(\sqrt{n})$.

$\therefore$ run-time of the algorithm is $\Theta(\sqrt{n} \log n)$ since line 2 is achieved in constant time.