Lecture 23 Exercise Solutions

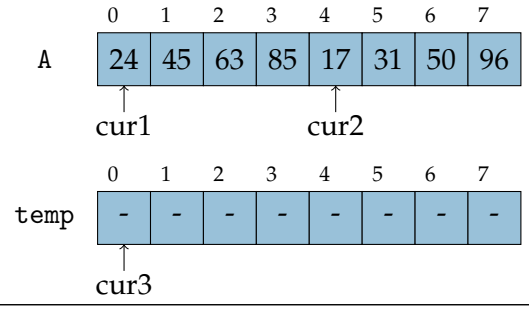
Mark Eramian

Exercise 1

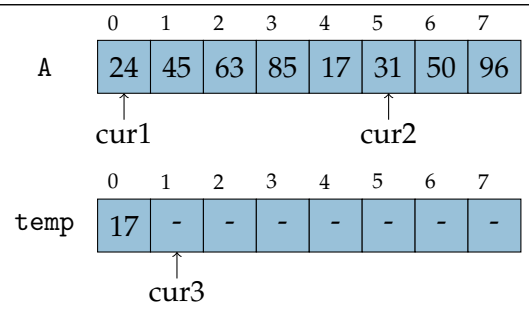
a)
```
// Assuming A is the array to be sorted...
int[] temp = new int[A.length];
merge( A, temp, 0, 4, 7);
```

b) Step-by-step merge operation:

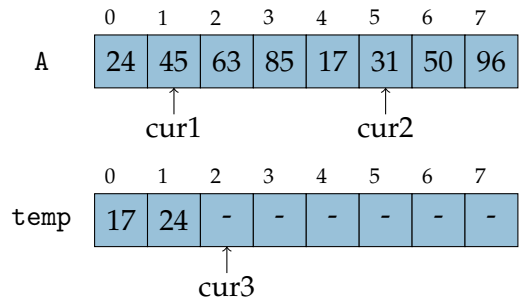
Initialization.



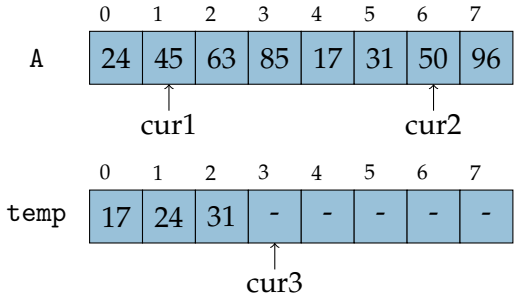
First loop, after 1 iteration.



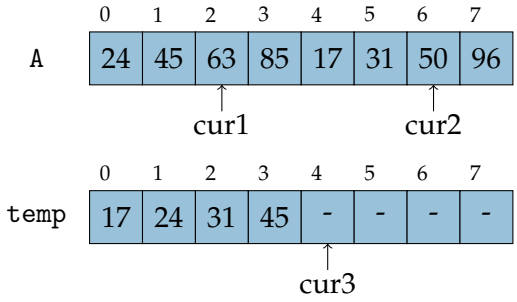
First loop, after 2 iterations.



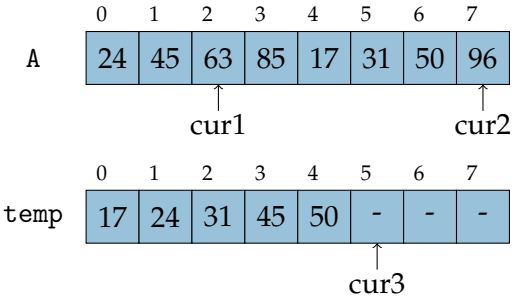
First loop, after 3 iterations.



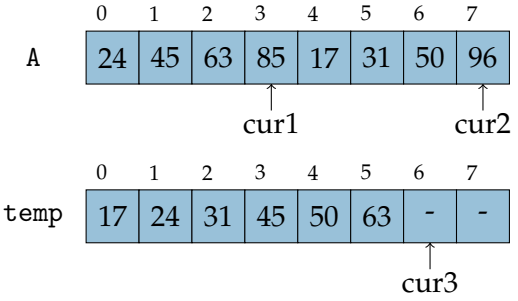
First loop, after 4 iterations.



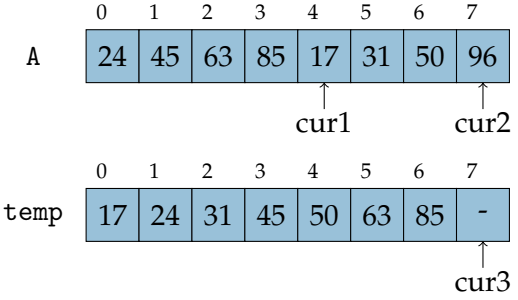
First loop, after 5 iterations.



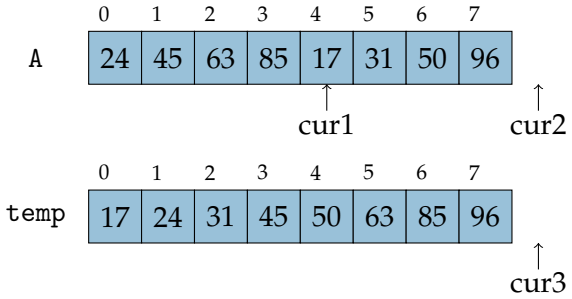
First loop, after 6 iterations.



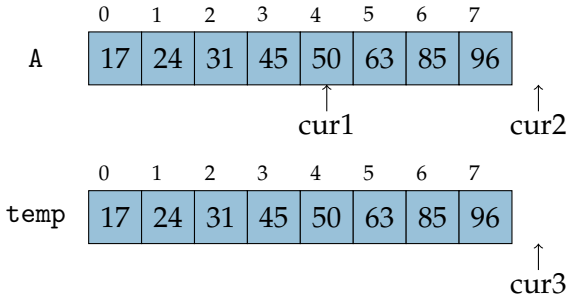
First loop, after 7 iterations. At this point, first loop condition is false (`cur1 < start2nd` is false). So we proceed to...



... second loop! It's condition is false so we end up in the third loop. Arrays shown after 1 iteration of third loop.

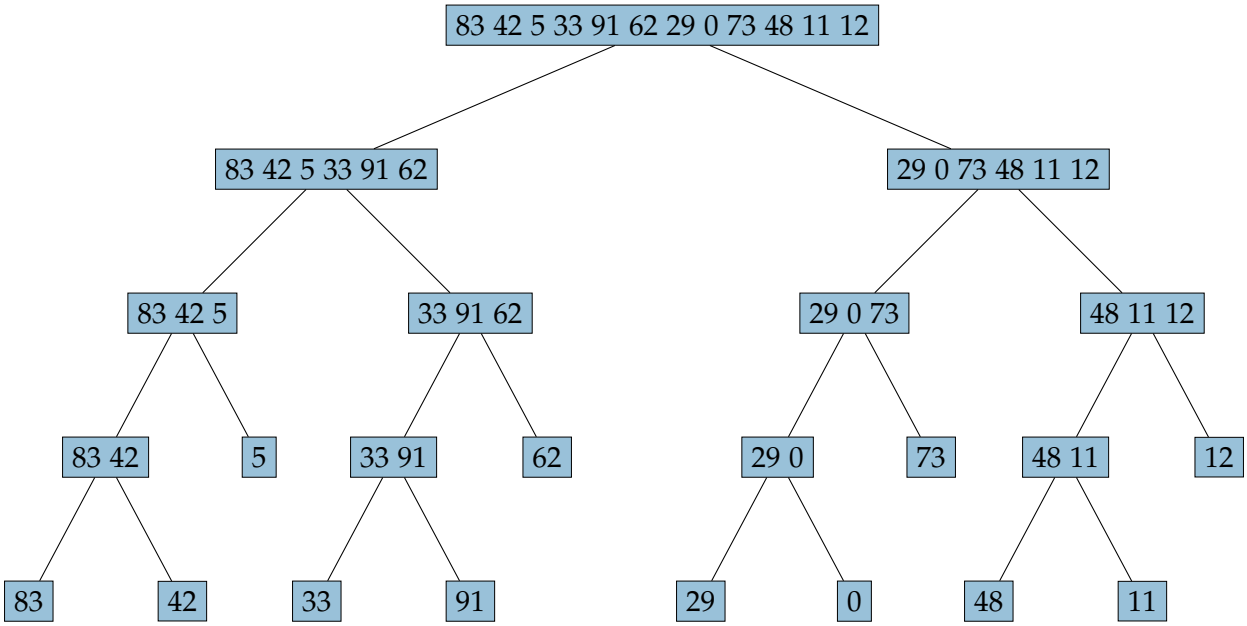


The third loop condition is now false, so we copy `temp` back into `A` and we're done. `A` is sorted!

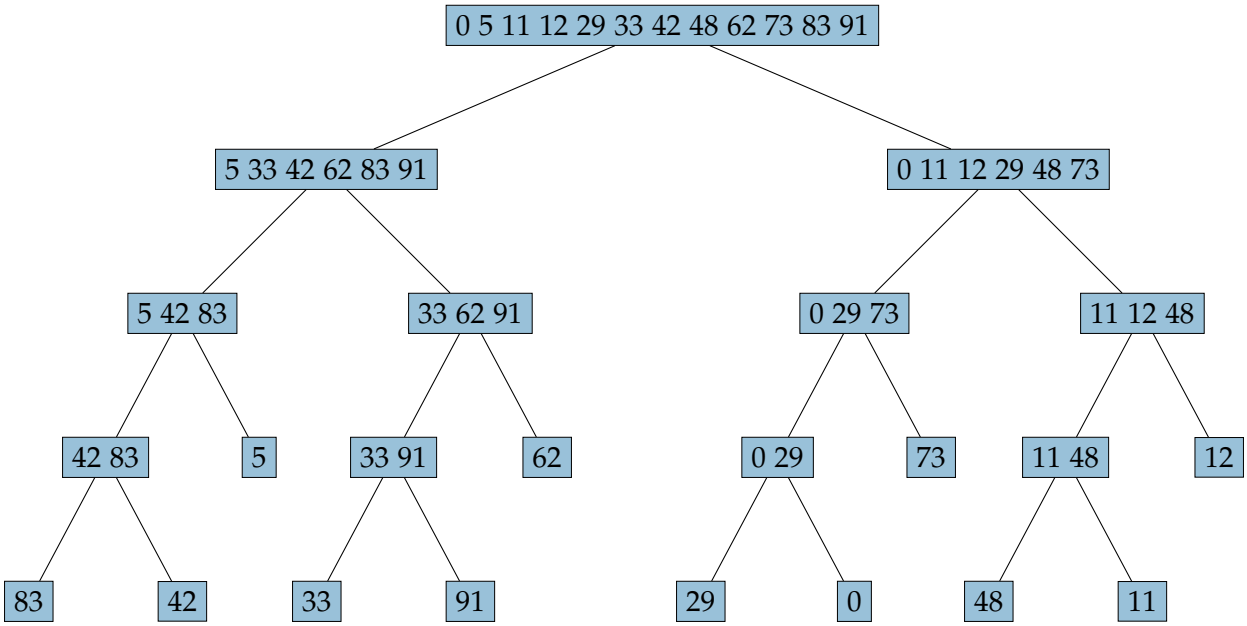


Exercise 2

The merge sort tree – input sequences:



The merge sort tree – output sequences:

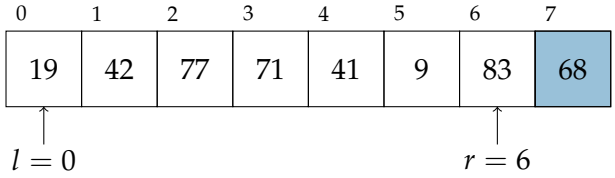


Exercise 3

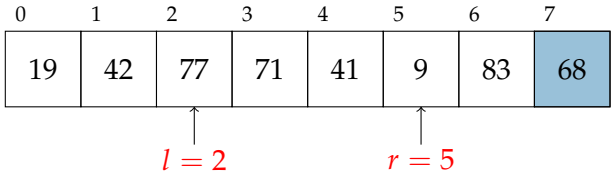
a) `// A is the array to be sorted`
`partition(A, 0, 7);`

b) Step-by-step trace of partitioning:

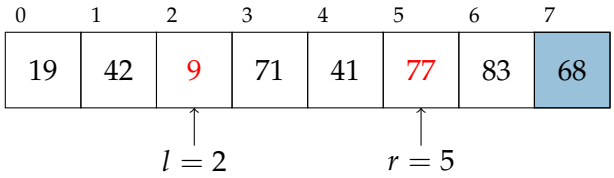
1. Initial Setup



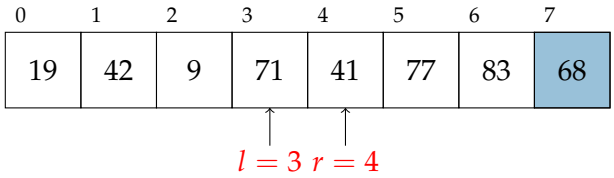
2. Advance l and r



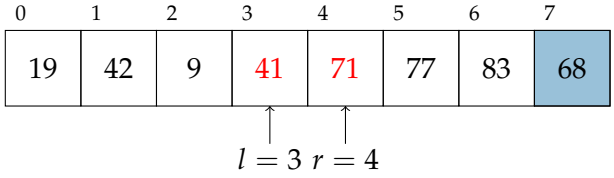
3. Swap



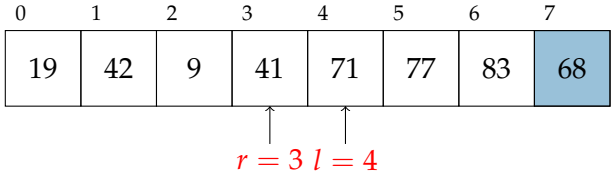
4. Advance l and r



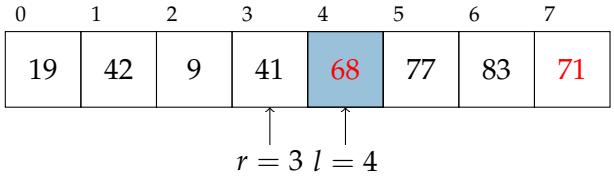
5. Swap



6. Advance l and r, now $l > r$

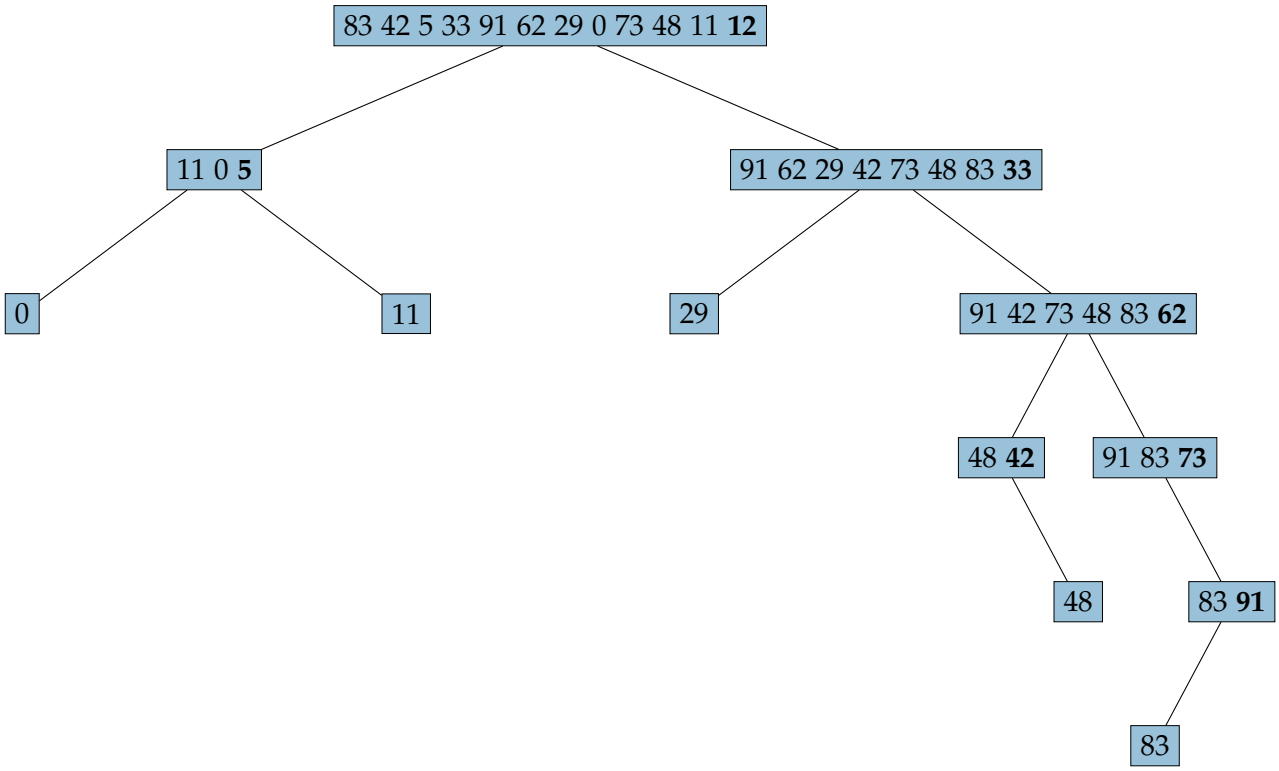


7. Swap l with pivot.

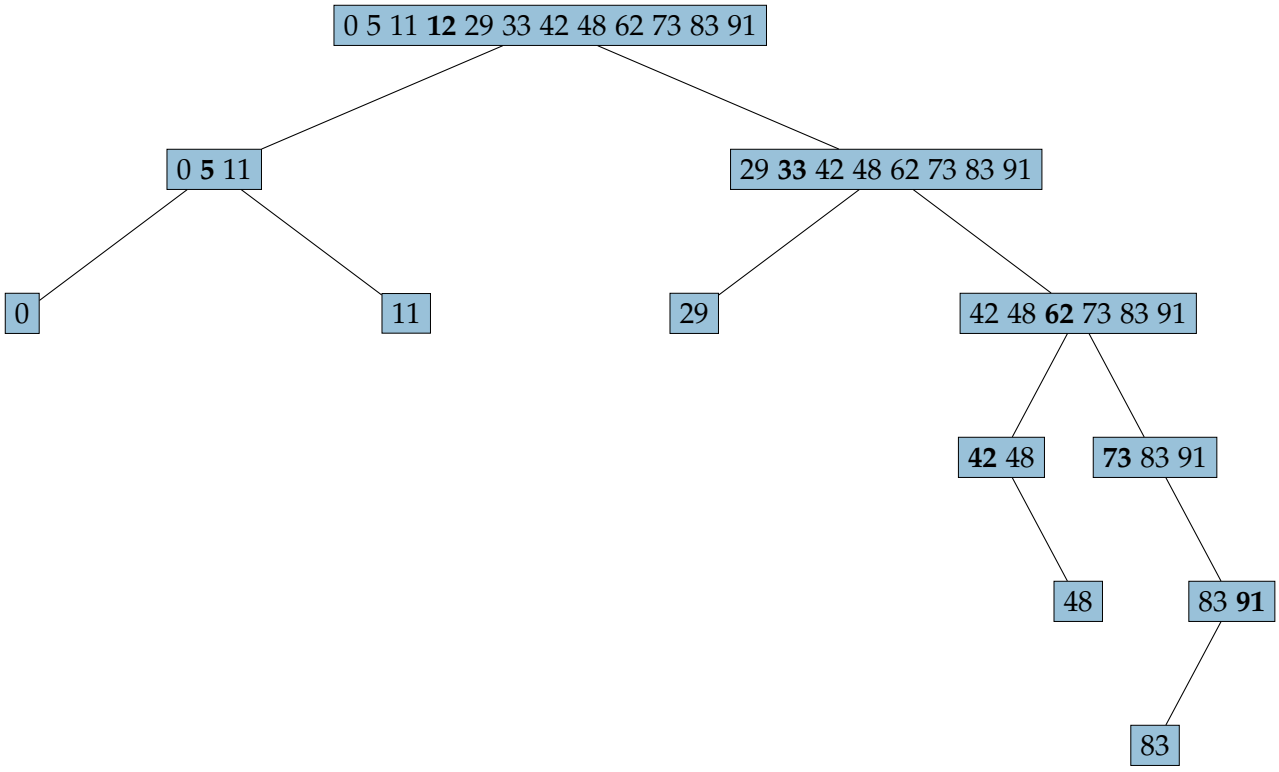


Exercise 4

Quick sort tree – input sequences, pivots shown in bold:

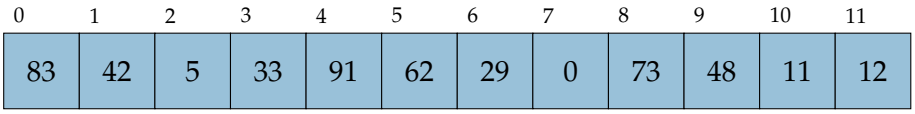


Quick sort tree – output sequences, pivots shown in bold:

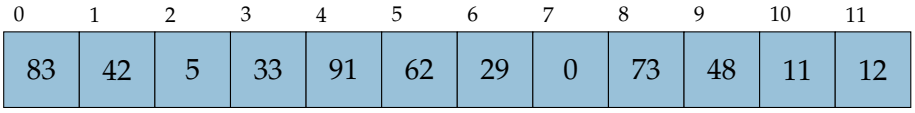

0 5 11 **12** 29 33 42 48 62 73 83 91
0 **5** 11
29 **33** 42 48 62 73 83 91
0
11
29
42 48 **62** 73 83 91
42 48
73 83 91
48
83 **91**
83

Exercise 5

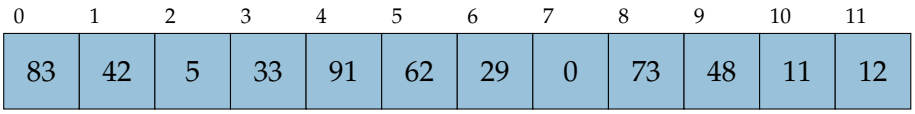
1. Initialization, $i = 11/2 = 5$



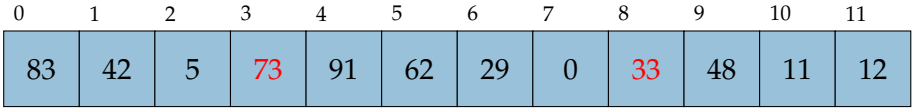
2. `moveDown()` for $i = 5$, 0 swaps; 62 is bigger than 12.



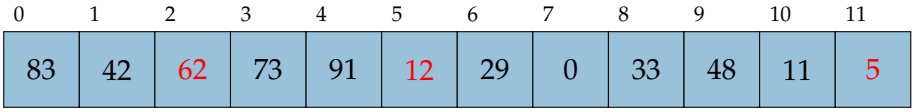
3. `moveDown()` for $i = 4$, 0 swaps; 91 is bigger than 48 and 11.



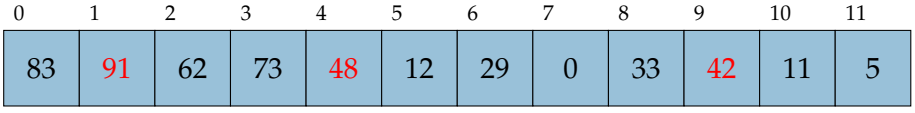
4. `moveDown()` for $i = 3$, 1 swap



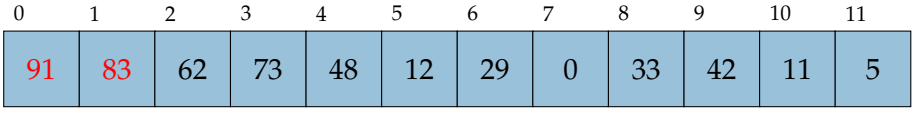
5. `moveDown()` for $i = 2$, 2 swaps.



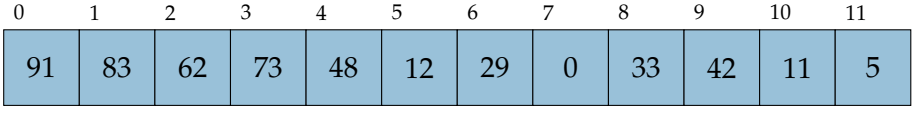
6. `moveDown()` for $i = 1$, 2 swaps.



7. `moveDown()` for $i = 0$, 1 swap.

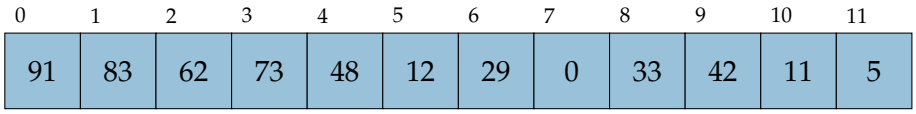


8. The array is now a heap.

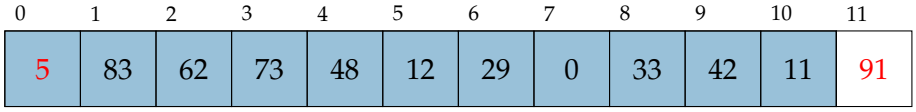


Exercise 6

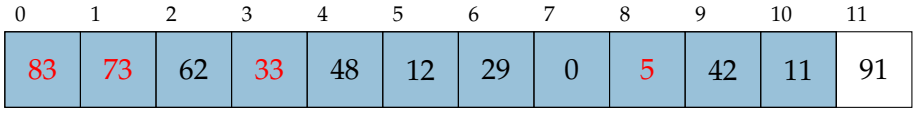
1. Starting with the heap from step 8 of Ex. 5., initialize $i = 11$



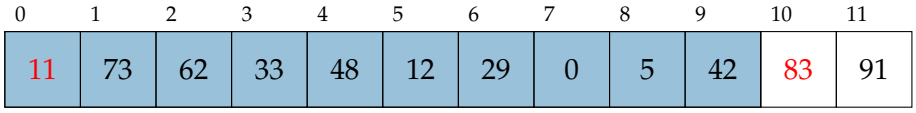
2a. Delete root of heap by swapping with index i; decrement i.



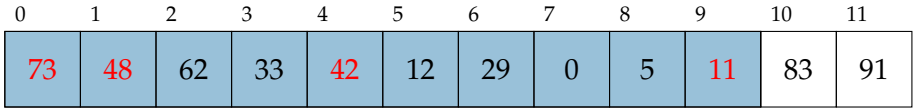
2b. `moveDown(data, 0, i=10)`.



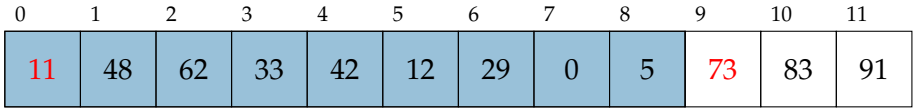
3a. Delete root of heap by swapping with index $i = 10$. Decrement i.



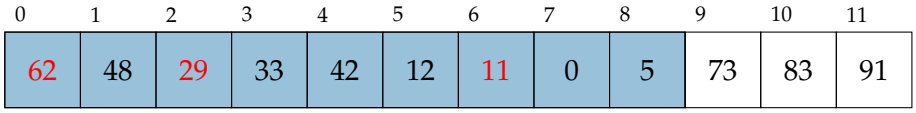
3b. `moveDown(data, 0, i=9)`.



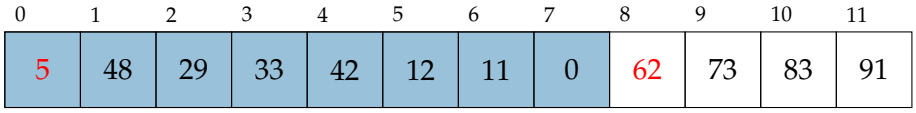
4a. Delete root of heap by swapping with index $i = 9$. Decrement i.



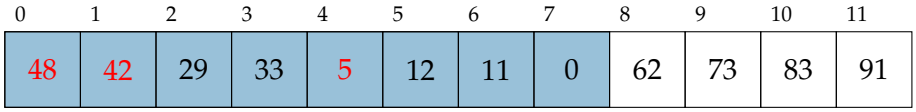
4b. `moveDown(data, 0, i=8)`.



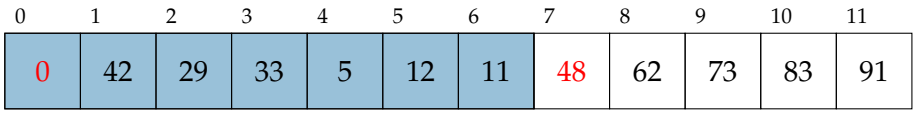
5a. Delete root of heap by swapping with index $i = 8$. Decrement i.



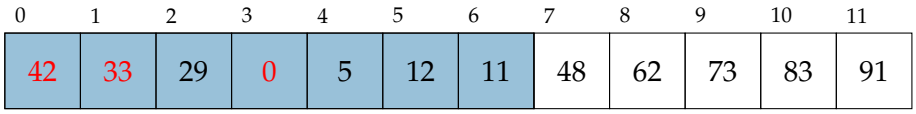
5b. `moveDown(data, 0, i=7)`.



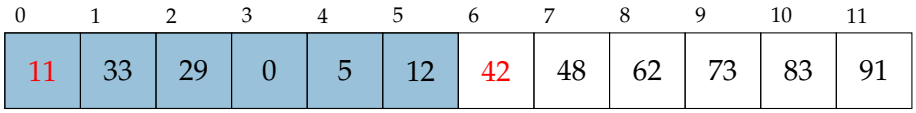
6a. Delete root of heap by swapping with index $i = 7$. Decrement i.



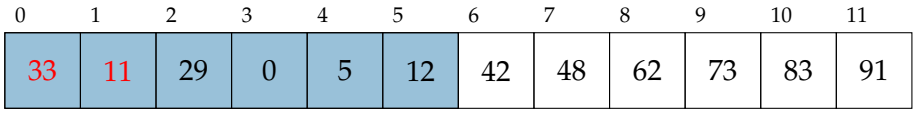
6b. `moveDown(data, 0, i=6)`.



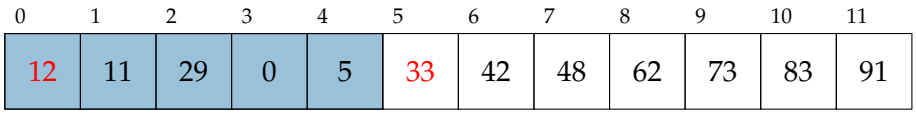
7a. Delete root of heap by swapping with index $i = 6$. Decrement i.



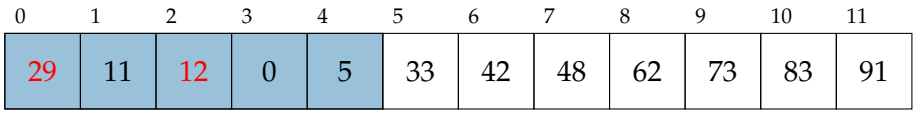
7b. `moveDown(data, 0, i=5)`.



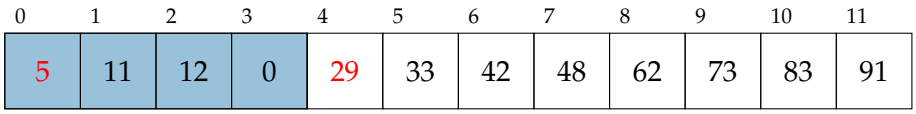
8a. Delete root of heap by swapping with index $i = 5$. Decrement i.



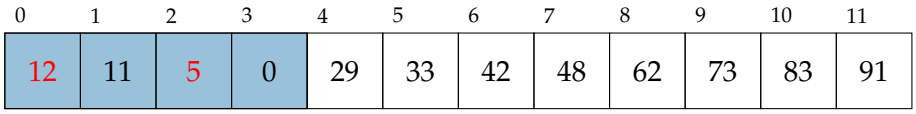
8b. `moveDown(data, 0, i=4)`.



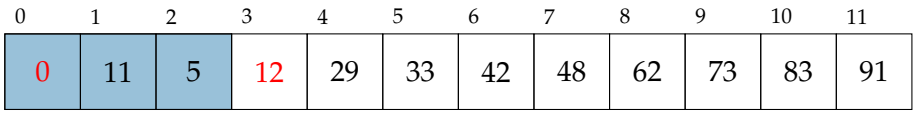
9a. Delete root of heap by swapping with index $i = 4$. Decrement i.



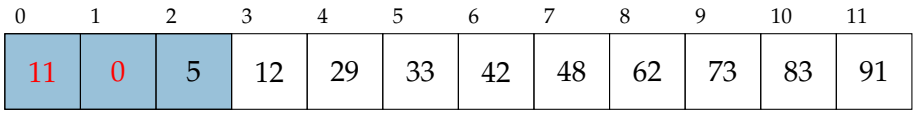
9b. `moveDown(data, 0, i=3)`.



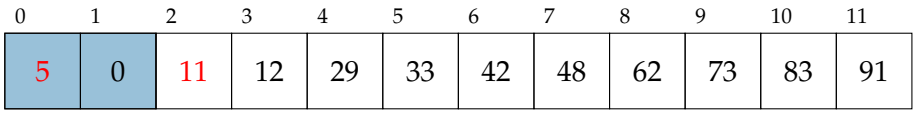
10a. Delete root of heap by swapping with index $i = 3$. Decrement i.



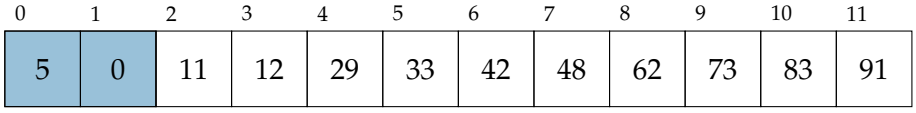
10b. `moveDown(data, 0, i=2)`.



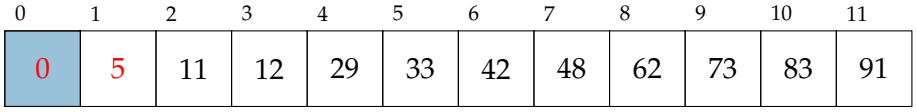
11a. Delete root of heap by swapping with index $i = 2$. Decrement i.



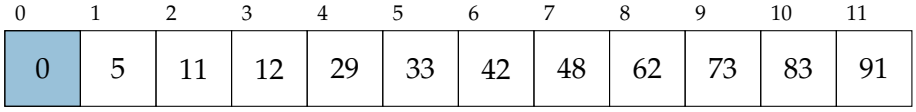
11b. `moveDown(data, 0, i=1)`.



12a. Delete root of heap by swapping with index $i = 1$. Decrement i.



12b. `moveDown(data, 0, i=0)`.



3. $i = 0$ so the loop stops and the array is sorted.