1 What's the running time of $HeapSort$, as a function of the length n of the input array?

1. $O(n)$
2. $O(n \cdot logn)$
3. $O(n^2)$
4. $O(n^2 logn)$

(2) is correct. A simple but not optimal way to obtain this bound is to insert the n items into the heap and then extract the min until the heap is empty into a array. The extract min operations is $O(logn)$ for n items hence we multiply and get the bound $O(n \cdot logn)$.

2 How many times does $Dijkstra$ execute lines 13 and 15? Select the smallest bound that applies

```
12    $for\, every\, edge\, (w^*, y)\, do$
13    $DELETE\, y\, from\, H$
14    $key(y) := min\{key(y), len(w^*) + l_{w^*y}\}$
15    $Insert\, y\, into\, H$
```

1. $O(n)$
2. $O(m)$
3. $O(n^2)$
4. $O(m \cdot n)$

(b) is correct. We check every edge from (w^*, y), there are m edges in total.

3 Which of the following pattern in a computer programs suggests that a heap data structure could provide a significant speed-up?

1. Repeated lookups
2. Repeated minimum computations
3. Repeated maximum computations
4. None of the others options

(2) and (3) are correct as we can quickly access the minimum item and maximum items for the respective heap. (1) is incorrect because a balanced-search tree or a hash-table are more suitable data structures to support quick look up of any key.

4 Suppose you implement the functionality of a priority queue using an *unsorted* array. What is the worst-case running time of $Insert$ and $ExtractMin$ respectively? Assume you have a large enough array to accommodate all your insertions?

1. $\Theta(1)$ and $\Theta(n)$
2. $\Theta(n)$ and $\Theta(1)$
3. $\Theta(n)$ and $\Theta(logn)$
4. $\Theta(n)$ and $\Theta(n)$

(1) is correct. We can insert by appending the item to the end of list in constant time. To find the minimum we iterate through the array swap the value with the end of the array and delete from the back. This operations in the worst case takes $O(n)$.

5 You are given a heap with n objects. Which of the following taks can you solve using $O(1)$ $Insert$ and $ExtractMin$ operations and $O(1)$ additional work.

1. Find the object sorted in the heap with the fifth-smallest key.
2. Find the object stored in the heap with the maximum key.
3. Find the object sotred in the heap with the median key
4. None of the above.

(1) is correct. We can remove five items from the heap inorder to get the fifth-smallest key. (2) is incorrect because in order to know the maximum key of the min-heap we have to extract all n elements from the heap. (3) is incorrect because we would have to extract all n items in the heap to find the median.

6 Continuing Problem 9.7, show how to modify the heap-based implementation of Dijkstra's algorithm for each vertex $v \in V$, the smallest bottlenect of an $s - v$ path. Your algorithm should run in $O((m + n) \cdot logn)$ time, where m and n denote the number of edges and vertices, respectively.

Similarly to Problem 9.7 we can modify the Dijkstra's relaxation algorithm to the form $max\{len(v^*, l_{v^*w^*}\}$. A similar analysis to regular Dijkstra's will show that we have a bound of $O((m + n) \cdot logn)$.