1 Solutions to HW1

Exercise 1.1
Show that the worst case running time of HEAPSORT is $\Omega(n \log n)$.

Solution 1.1
Problem 2 $\implies$ Problem 1

Exercise 1.2
Show that when all elements are distinct, the best case running time of HEAPSORT is $\Omega(n \log n)$.

Solution 1.2
Let's assume that $n = 2^k - 1$, in other words, it forms a complete binary tree.

We are now going to explore the biggest $\frac{n}{2}$ elements, let's call them big. We are now interested how many big elements are leaves and how many are inner nodes.

Observation 1. The big elements induce a subtree in the heap.

and one more observation.

Observation 2. Any tree with $n$ vertices has at most $\frac{n+1}{2}$ leaves.

This observation implies that number of inner vertices, let's call the set of them $A$, in the heap is at least $\frac{n}{2}$. In the first $\frac{n}{2}$ steps of heapsort we only move to the root the leaves of the original heap. If vertices of $A$ was sorted in proces of first $\frac{n}{2}$ steps of heapsort they have to go all the way up from they original position, because they cannot become leaves of heap. Let's make a bound on how much it takes.

Lemma 1. There is at least $\frac{n}{4}$ of vertices of $A$ which are deeper than $\frac{\log n}{2}$

Proof. Assume it is not true. Let's count how many vertices can be in the heap of depth $\frac{\log n}{2}$. It can have at most $2^{\frac{\log n}{2}} = n^{\frac{1}{2}}$.

for big enough $n$ the expression $n^{\frac{1}{2}}$ is smaller than $\frac{n}{4}$. Which means there are at least $\frac{n}{4}$ vertices from $A$ which are deeper.

But this already conclude that the best case running time of heapsort is $\Omega(n \log n)$ becase there is $\frac{n}{4}$ vertices where each of them has to make $\frac{\log n}{2}$ steps which is $\frac{n \log n}{8} = \Omega(n \log n)$. 

1
Exercise 1.3
QUICKSORT: Argue that for any constant $0 < \alpha \leq \frac{1}{2}$, the probability is approximately $1 - 2\alpha$ that on a random input array, PARTITION produces a split more balanced than $1 - \alpha$ to $\alpha$.

Solution 1.3
We start with simple observation which helps with our intuition:

Observation 3. Probability of choosing element $a$ in our current settings is same as choosing element $a$ by RANDOM-PARTITION in sorted array.

We can now assume a sorted list and RANDOM-PARTITION which shots randomly uniformly into elements. Because we want just an approximate solution instead of sorted discrete elements we can think of an interval, see Fig. 1.

$$\alpha \quad \frac{1}{2} \quad 1 - \alpha$$

Figure 1: Input data as interval.

Which partition of interval is more balanced that $\alpha$ and $1 - \alpha$, see Fig. 1. The answer to that is on Fig. 2, on each half of interval more balanced partition is interval of size $\frac{1}{2} - \alpha$ which implies that on whole interval it is $1 - 2\alpha$ as promised.

$$\alpha \quad \frac{1}{2} \quad 1 - \alpha$$

$$\frac{1}{2} - \alpha \quad \frac{1}{2} - \alpha$$

Figure 2: That is why more balanced partition has probability $1 - 2\alpha$.

Exercise 1.4
Show that the best case running time of QUICKSORT is $\Omega(n \log n)$.
Solution 1.4
Analysis is very much same as showing upper bound for the worst case time!
Assume that best case running time is defined by a function $T$.

\[ T(n) \geq \min\{T(n-k) + T(k-1)\} + \Omega(n) \]

where min is over possible values of $k$ (which are $[1, \ldots, n]$). We will now guess that $T(n) \geq cn\log n$ for some constant $c$. Our assumption implies

\[ T(n) \geq \min\{c(n-k)\log(n-k) + c(k-1)\log(k-1)\} + \Omega(n) \]

tools from calculus help us find out that minimum is for $k$ approximately $\frac{n}{2}$ which together with right choice of constant $c$ implies $\Omega(n\log n)$.

Exercise 1.5
Show how to sort $n$ integers in the range $0$ to $n^3 - 1$ in $O(n)$ time.

Solution 1.5
We can convert number from binary (or decimal) base to base $n$, in constant time (if we can assume that division and modulo is in constant time). And produce numbers in base $n$ with constant number of digits in this case at most $3$. Therefore, it is a perfect case to use radix sort which will take only $O(n)$ time.

Exercise 1.6
Prove the lower bound of $\lceil \frac{3n^2}{2} \rceil - 2$ comparisons in the worst case to find both the maximum and minimum of $n$ numbers. (Hint: Consider how many numbers are potentially either the maximum or minimum, and investigate how a comparison acts these counts.)

Solution 1.6
To solve this problem we will use the adversary technique. We are trying to construct as bad case as we can and hopefully it is bad enough to get bound $\lceil \frac{3n^2}{2} \rceil - 2$.

In the beginning we do not know nothing about our $n$ elements. All elements are candidates for being maximum and minimum. We will simulate process of comparisons and keep elements in 4 different buckets, according to what we know so far about them. The first bucket we call "max/min" and refers to elements which still can be maximum and minimum. The second bucket is "max" and refers to elements which can be maximum but cannot be minimum. Analogically the third bucket is "min". And the last bucket is called "neither" and contains elements which cannot be minimum or maximum.

We now define consistent adversary answers.
### Comparison | Answer
--- | ---
1. max/min vs max/min | answer either way
2. max/min vs neither | answer max/min is larger
3. min vs min | answer either way
4. max vs max | answer either way
5. max vs "other" | answer max is larger
6. min vs "other" | answer "other" is larger
7. neither vs neither | answer arbitrarily but consistently

In the beginning we have $2n$ candidates $n$ for maximum and $n$ for minimum, during the process of algorithm we need to remove $2(n-1)$ of them so we have just one candidate for maximum and one candidate for minimum. If we follow the table, in the first case the larger element is moved to max and smaller element to min, which means we eliminated two candidates. In the all other cases we eliminate at most one candidate. Because the first case can occur at most $\lfloor n/2 \rfloor$ times we can get the following lower bound.

$$(2(n-1) - 2\lfloor \frac{n}{2} \rfloor) + \lfloor \frac{n}{2} \rfloor = 2n - 2 - \lfloor \frac{n}{2} \rfloor = \lfloor \frac{3n}{2} \rfloor - 2.$$