Problem 1

(a) (i) Procedure for HEAVY(i, j)[1]:

Algorithm 1 Heavy, returns the set of heavy elements within a range i, j, inclusive.

```
1:  procedure HEAVY(i, j)
2:      if i = j then
3:          return {A[i]}                 ▷ Base case of a single element, which is heavy on its own.
4:      end if
5:      m ← ⌊(i+j)/2⌋                      ▷ Take the midpoint of i and j.
6:      L ← HEAVY(i, m)                    ▷ Recurse on left half
7:      R ← HEAVY(m+1, j)                  ▷ Recurse on right half
8:      Create Set S                       ▷ Initialise an empty set, to be returned.
9:      for all e ∈ L ∪ R do               ▷ For each (distinct) element in the both L and R
10:         c ← count of e in A[i...j]     ▷ O(j − i + 1), linear w.r.t. size of A[i...j]
11:         if c > (3/20)(j − i + 1) then
12:             S ← S ∪ {e}                ▷ e is (still) heavy, O(2|S|)
13:         end if
14:     end for
15:     return S                           ▷ Return the heavy elements for A[i...j]
16: end procedure
```

(ii) The algorithm HEAVY first checks the base case, input where $i = j$. For any base case, the set of heavy items is a singleton of the element $A[i]$ itself (since $1 \geq 1 = \lceil \frac{3}{20} \rceil$).

Next, we divide the problem into two subproblems HEAVY(i,m) and HEAVY$(m+1, j)$, each of which return a set of heavy numbers from the respective subarrays $A[i \dots m]$ and $A[m+1 \dots j]$.

We take the union of these sets ($L \cup R$) to ensure no duplicated elements, then walk through the set. For each distinct element, we count the occurrences in $A[i \dots j]$ (1.10) and check if the count c satisfies the heaviness condition (1.11). Note that the counting needs to make one pass through the array and is hence $O(j-i+1)$. If the condition is satisfied, the element is added (1.12) to a separate set (1.8).

(b) Let's do some induction!

[1]Algorithm construction assisted by *redacted*.

Let the inductive proposition be $I(n)$: for some i, j, such that $i < j$, $n = j - i + 1$, $\textsc{Heavy}(i, j)$ returns a set S such that e is heavy for all $e \in S$.

Base case ($n = 1$). By the proposition, for $I(1)$, we have $i = j$ and we correctly return the set of heavy elements, which is the singleton set $\{A[i]\}$.

General case ($n > 1$). Assume $I(k)$ is true for all $k < n$. Now we have $m = \left\lfloor \frac{i+j}{2} \right\rfloor$ and we split into subcases with sizes $m - i + 1 = \left\lfloor \frac{j-i}{2} \right\rfloor + 1$ and $j - (m+1) + 1 = \left\lceil \frac{j-i}{2} \right\rceil$.

Since $\left\lfloor \frac{j-i}{2} \right\rfloor + 1 < n$ and $\left\lceil \frac{j-i}{2} \right\rceil < n$, we know by assumption that I is true and that $\textsc{Heavy}$ correctly returns the set of heavy elements of $A[i \dots m]$ and $A[m+1 \dots j]$ respectively.

We then check each element e of the union of the subresults and ensure that the element is still heavy based on the condition that e occurs at least 15% of the time.

Note that non-heavy elements are correctly discarded. If some array element e isn't heavy (occurs less than 15%) on the left and isn't heavy on the right, then e isn't heavy on the combined array.

(c) **Base Case** ($n = 1$). $T(1) = 1$. This is the cost of constructing $\{A[i]\}$.

General Case ($n > 1$). $T(n) = T\left(\left\lfloor \frac{n-1}{2} \right\rfloor + 1\right) + T\left(\left\lceil \frac{n-1}{2} \right\rceil\right) + c_1 n$. For the subcases, we are reusing the derivations $\left\lfloor \frac{n}{2} \right\rfloor + 1$ and $\left\lceil \frac{n}{2} \right\rceil$ from part (b) by substituting $n - 1 = j - i$.

Note that $|L|$, $|R|$, and thus $|L \cup R|$ are bounded by a constant. The maximum amount of heavy numbers for any $\textsc{Heavy}(i, j)$ for some $n = j - i + 1$ is 6. Why? We prove by contradiction. Suppose the set returned contains 7 or more distinct heavy elements. Each element would have to appear more than 15% of the set, accounting for 105% (of n). This is a contradiction since the set should only contain up to 100% of items. This result allows us simplify the complexity of lines 9 to 14 to $c_1 n$ where the n comes from line 10 and the c_1 comes from summing the constant operations (set union $L \cup R$, $S \cup \{e\}$, etc.).

We then have $T(n) \leq T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + T\left(\left\lceil \frac{n}{2} \right\rceil\right) + c_1 n \leq 2T \leq 2T\left(\left\lceil \frac{n}{2} \right\rceil\right) + c_1 n$. By the master theorem for inequalities, we have $c = \log_2 2 = 1$ and thus $T(n) = O(n \log n)$.

Problem 2

(a) (i) Let $X_i = \begin{cases} 1 & \text{if } A[i] \text{ is a local minimum} \\ 0 & \text{otherwise} \end{cases}$.

For end cases, we have $P(X_1 = 1) = P(X_n = 1) = \frac{1}{2}$. (Why?[2]) For middle cases X_i, we consider $A[i-1], A[i], A[i+1]$. There are 3! permutations and exactly 2 of these permutations situate $A[i]$ as a local minimum. Thus we have $P(X_i = 1) = \frac{2}{3!} = \frac{1}{3}$.

There are 2 end cases and $n-2$ middle cases. The expected number of local minima in A is hence $E(\sum_{i=1}^{n} X_i) = 2 \times \frac{1}{2} + (n-2) \times \frac{1}{3} = \frac{n+1}{3}$. (Source.[3])

(ii) Generalising from the array end case and middle case, let S be the data structure under inspection, n be the number of nodes of S, i be some index $1 \leq i \leq n$, and m be the number of neighbours of $S[i]$ (the i^{th} node of S). Let X_i be the indicator random variable, indicating if $S[i]$ is a local minimum. Then $P(X_i = 1) = \frac{m!}{(m+1)!} = \frac{1}{m+1}$ (we fix the i^{th} node and take the permutations of the m neighbours).

The nodes and expected values are broken down as follows: Firstly, the root. There is only one; with $P(X_1 = 1) = \frac{1}{3}$. Secondly, the leaves, each with one neighbour. There are $\frac{n+1}{2}$. For $n \geq 2$, we have $P(X_i = 1) = \frac{1}{2}$, where $S[i]$ are leaves.

This leaves[4] the nodes in the middle levels, each of which have three neighbours. Theare are $n - 1 - \frac{n+1}{2} = \frac{n-3}{2}$. For $n \geq 3$, we have $P(X_i = 1) = \frac{1}{4}$, where $S(i)$ are middle nodes.

[2] We can prove this by induction. Let P_n denote the probability that the first element (left end) is a local minimum. Given an array A with random permutation of $\langle 1, \ldots, n \rangle$ $(n \geq 2)$, we have a base case of $P_2 = \frac{1}{2}$. Assume $P_{n-1} = \frac{1}{2}$. We formulate $P_n = \frac{1}{n} + \frac{n-2}{n} P_{n-1}$, which by solving gives $\frac{1}{2}$.

P_n can be pictured as inserting a new larger element n into the random permutation of $\langle 1, \ldots, n-1 \rangle$. We retain the local minimum by inserting n into the second index (corresponding to $\frac{1}{n}$) or inserting in the third index or beyond (corresponding to the recursive $\frac{n-2}{n} P_{n-1}$).

[3] Solution inspired from: `https://math.stackexchange.com/q/680660/615376`

[4] haha

The total expected value of local minima in a tree is then

$$E(\sum_{i=1}^{n} X_i) = \begin{cases} 0 & \text{if } n = 0 \\ \frac{1}{3} & \text{if } n = 1 \\ \frac{1}{3} + \frac{n+1}{2} \cdot \frac{1}{2} + \frac{n-3}{2} \cdot \frac{1}{4} = \frac{9n+5}{24} & \text{if } n \geq 3 \end{cases}$$

(Note that the $n = 2$ case doesn't exist by assumption.)

(iii) We treat (i, j) elements as nodes and use similar methodologies. Let $n = m^2$ be the number of elements of the matrix.

For each node, if $M(i, j)$ has

- 2 neighbours (corners):

$$X_{(i,j)} = \begin{cases} 0 & \text{if } m = 1 \\ 4 & \text{if } m \geq 2 \end{cases}$$

- 3 neighbours (edges, non-corners):

$$X_{(i,j)} = \begin{cases} 0 & \text{if } m = 1 \\ 4m - 8 & \text{if } m \geq 2 \end{cases}$$

- 4 neighbours:

$$X_{(i,j)} = \begin{cases} 0 & \text{if } m = 1 \\ (m-2)^2 & \text{if } m \geq 2 \end{cases}$$

The expected value of local minima is then

$$\begin{cases} 0 & \text{if } m = 1 \\ 4 \cdot \frac{1}{3} + (4m-8) \cdot \frac{1}{4} + (m-2)^2 \cdot \frac{1}{5} = m - \frac{2}{3} + \frac{1}{5}(m-2)^2 & \text{if } m \geq 2 \end{cases}$$

(b) (i) Similar to the array case, but without the end points. For each middle element, fix the middle element and permute the other two: two cases per X_i. Let X_i be the indicator random variable indicating if $A[i]$ is a saddle point.

$$E\left(\sum_{i=1}^{n} X_i\right) = \frac{n-2}{3}.$$

(ii) We only consider the elements in the middle level. So $P(X_i) = 0$ for i where $T[i]$ is a root or leaf.

From a.ii, we know there are $\frac{n-3}{2}$. Each of these nodes have 4 neighbours, strictly ordered. For some $T[i]$, we consider its parent, $T[i]$, and its two children. Fixing the parent and $T[i]$, we deduce that there are two permutations (with $T[i]$ as a saddle point) out of 4!. Thus, $P(X_i = 1) = \frac{1}{12}$, and $E(\sum_{i=0}^{n} X_i) = \frac{n-2}{12}$.

(iii) We split this problem into multiple cases. For corners, there is 1 out of 3! permutations satisfying the constraint. For non-corner edges, 2 out of 4! permutations. For inner cells, 4 out of 5! permutations. The expected value is then

$$\begin{aligned} E\left(\sum_{i=0}^{n}\right) &= \begin{cases} 0 & \text{if } m = 1 \\ 4 \cdot \frac{1}{3!} + (4m-8) \cdot \frac{2}{4!} + (m-2)^2 \cdot \frac{4}{5!} & \text{if } m \geq 2 \end{cases} \\ &= \begin{cases} 0 & \text{if } m = 1 \\ \frac{2}{3} + \frac{m-2}{3} + \frac{(m-2)^2}{30} & \text{if } m \geq 2 \end{cases} \end{aligned}$$

Problem 3

(a) (i) There is at least one minimum, and it is not at infinity, since x_i are defined to be real numbers. Thus, there exists z_1 such that $f(x)$ is monotonically decreasing.

(ii)

Problem 4

(a) $T(n) = O(n^{5/2})$

(b) $T(n) = O(n^{\log_2 9})$

(c) $T(n) = O(n^2 \log n)$

(d) $T(n) = O(n^{\log_4 3})$

(e) $T(n) = O(n^{\log_7 2} \log n)$

(f) $T(n) = O(n \log n)$