Learning Objectives

By the end of this worksheet, you will:

- Analyse the average running time of an algorithm.
- Analyse the worst-case and best-case running time of functions.

1. **Average-case analysis.** Consider the following algorithm that we studied a few weeks ago. The input is an array A of length n, containing a list of n integers.

```
def hasEven(A):
    """A is a list of integers."""
    n = len(A)
    even = False
    for i in range(n)
        if A[i] % 2 == 0:
            print('Even number found')
            return i
    print('No even number encountered')
    return -1
```

In class we proved that the worst-case complexity of this algorithm is $\Theta(n)$. In this problem we will examine the *average case* complexity of this algorithm.[1]

For simplicity, we will assume that the input is a *binary* array A of length n. That is, A is an array containing a list of n integers, where each integer is either 0 or 1.

(a) For each $n \in \mathbb{Z}^+$, let T_n be the set of all binary arrays of length n. Write an expression (in terms of n) for $|T_n|$, the size of T_n.

[1] This was done in lecture, however the limits of summation were slightly different, and this makes a good review.

(b) For each $n \in \mathbb{Z}^+$ and each $i \in \{0, 1, \dots, n-1\}$, let $S_n(i)$ denote the set of all binary arrays A such that the first 0 occurs in position i. More precisely, let $S_n(i)$ denote the binary arrays that satisfy the following two properties:

(i) $A[i] = 0$.

(ii) for all $j \in \mathbb{N}$, if $j < i$ then $A[j] = 1$.

Also let $S_n(n)$ be the set of binary arrays that contain no 0's. For each i, $0 \leq i \leq n$, write an expression for $|S_n(i)|$.

(c) Argue that for each $n \in \mathbb{Z}^+$, each binary array of length n is in exactly one set S_i (for some $i \in \{0, \dots, n\}$). Use this to show that $\sum_{i=0}^{n} |S_n(i)| = |T_n|$.

(d) Let the runtime of the algorithm on a binary list A be the number of executions of the loop. Give an exact expression for the average runtime of the above algorithm using the quantities that you calculated.

You should get a summation; do not simplfy the summation in this part.

(e) Show that the runtime that you calculated is in $O(1)$. You may use without proof that for all $x \in \mathbb{R}$ such that $|x| < 1$, $\sum_{i=1}^{\infty} ix^i = \frac{x}{(1-x)^2}$.

2. **Bipartite graphs.** A **bipartite graph** is a graph $G = (V, E)$ that satisfies the following properties:

   (a) There exist subsets $V_1, V_2 \subset V$ such that $V_1 \neq \emptyset$, $V_2 \neq \emptyset$, and V_1 and V_2 form a *partition* of V.[2]

   (b) Every edge in E has exactly one endpoint in V_1, and exactly one endpoint in V_2. (That is, no two vertices in V_1 are adjacent, and no two vertices in V_2 are adjacent.)

   When G is bipartite, we call the partitions V_1 and V_2 a **bipartition of** G.

   (a) Prove that the following graph $G = (V, E)$ is bipartite.

   $$V = \{1, 2, 3, 4, 5, 6\} \quad \text{and} \quad E = \{(1, 2), (1, 6), (2, 3), (3, 4), (4, 5), (5, 6)\}$$

   (b) Let m and n be positive integers. A **complete bipartite graph on** (m, n) **vertices** is a graph $G = (V, E)$ that satisfies the following properties:

      i. There exist subsets $V_1, V_2 \subset V$ such that $V_1 \neq \emptyset$, $V_2 \neq \emptyset$, and V_1 and V_2 form a *partition* of V.

      ii. Every edge in E has exactly one endpoint in V_1, and exactly one endpoint in V_2. (That is, no two vertices in V_1 are adjacent, and no two vertices in V_2 are adjacent.)

      iii. (new) $|V_1| = m$ and $|V_2| = n$.

      iv. (new) For all vertices $u \in V_1$ and $w \in V_2$, u and w are adjacent.

   How many edges are in a complete bipartite graph on (m, n) vertices? Your answer will depend on m and n. Explain your answer.

[2]That is, $V_1 \cup V_2 = V$ and $V_1 \cap V_2 = \emptyset$.