1 Part 1 Theory

1.1 Stable Matching Problem

a) Explain stable matching in your own words.

Stable matching involves finding a stable match for two sets of elements with equal size, given a preference order for each element. This problem is usually presented as the stable marriage problem, where given N men and N women where each person has ranked all members of opposite sex in order of preference. The men and women have to get married so that there are no two people of opposite sex who would both rather have each other than their current partners. If there exists a matching where a pair prefer each other to their current partners, then the matching is not stable.

b) Consider a version of the SMP in which we are attempting to match schools with students.

Since the student s prefers university u the most, the student will try to match with university u first. If the university is unmatched, it will match with the student s. If university u is matched already, it will "trade up" because it prefers student s to all other students.
If it is the university u that tries to match up with students, the same situation will occur. Because university u prefers student s the most, it will try to match up with student s first. If student s is unmatched, the student will accept the matching. If student s is matched to another university, they will "trade up" to university u because they prefer that university the most. In other words, in every S of this occurence, the pair *(s,u)* belongs to S.

1.2 Asymptotic Growth Rate

a) Sorted list of functions in ascending order of growth rate.

$$\begin{aligned} g_3(n) &= 400 \\ g_5(n) &= 3 * n \\ g_4(n) &= log(n) \\ g_1(n) &= n^2 * log(n) \\ g_6(n) &= 5 * n^3 \\ g_2(n) &= n! \end{aligned} \tag{1}$$

We have the relation $n * log(n) \leq n^2$ for all $n \geq 1$, so we know we kan classify $g_5(n)$, $g_2(n)$ and $g_6(n)$ with a larger growth rate than $g_4(n)$ and $g_1(n)$. $g_3(n)$ is classified as the smallest growth rate, seeing as it is only run once and can be simplified to $O(1)$. Other than that, the rest of the functions were sorted following mathematical logic.

b) Match expressions so that there is an $f_i(n) = \Theta(g_i(n))$.

The $\Theta(g_i(n))$ notation means that $f_i(n) = \Theta(g_i(n))$ are asymptotically tight bound. This occurs when $f_i(n)$ is both $f_i(n) = O(g_i(n))$ and $f_i(n) = \Omega(g_i(n))$.

$$\begin{aligned} f_1(n) = &\quad \Theta(g_4(n)) \\ f_2(n) = &\quad \Theta(g_1(n)) \\ f_3(n) = &\quad \Theta(g_3(n)) \\ f_4(n) = &\quad \Theta(g_2(n)) \end{aligned} \tag{2}$$

c) Consider the relationship between the functions $f(n)$ and $g(n)$ for Ω, O and Θ.

$$\begin{aligned} f(n) = &\quad lg(n^{lg7}) \\ g(n) = &\quad lg(7^{lgn}) \end{aligned} \tag{3}$$

We start by considering the relationship $f(n) = \Omega(g(n))$. For this relationship to be true, $g(n)$ has to be an asymptotic lower bound for $f(n)$. We try to solve for this by

$$\begin{aligned} f(n) > &\quad g(n) \\ lg(n^{lg7}) > &\quad lg(7^{lgn}) \end{aligned} \tag{4}$$

This inequality returns no results, meaning that $f(n) \neq \Omega(g(n)) \rightarrow g(n)$ is not an asymptotic lower bound for $f(n)$.
For the relationship $f(n) = O(g(n))$ to be true, $g(n)$ has to be an asymptotic upper bound for $f(n)$. We try to solve for this by

$$\begin{aligned} f(n) < &\quad g(n) \\ lg(n^{lg7}) < &\quad lg(7^{lgn}) \end{aligned} \tag{5}$$

Which returns a valid result, meaning the inequality is valid. $g(n)$ therefore has to be an asymptotic upper bound for $f(n)$. $f(n) = O(g(n)) \rightarrow g(n)$ is an asymptotic upper bound for $f(n)$
For the relationship $f(n) = \Theta(g(n))$ to be true, $g(n)$ has to be an asymptotic tight bound for $f(n)$, meaning that both $f(n) = \Omega(g(n))$ and $f(n) = O(g(n))$ has to be true. We have considered both relationships and found that only $f(n) = O(g(n))$ is true. Therefore $f(n) \neq \Theta(g(n)) \rightarrow g(n)$ is not an asymptotic tight bound for $f(n)$.

d) Simplify the asymptotic expression $\Theta(n^2) + O(n)$ without loss of precision.

We simplify the expression to $\Theta(n^2)$ by using the additivity property of asymptotic growth rates that allows us to compute the following:

$$\text{If } f = \Theta(h) \text{ and } g = O(h) \text{ then } f + g = \Theta(h) \tag{6}$$

In this case, $f = \Theta(n^2)$ and $g = O(n)$, meaning $f + g$ can be simplified to $\Theta(n^2)$.

1.3 Runtime Analysis

a) Analysis of ArraySum

The ArraySum algorithm inputs a 2D $n \times n$ list into Sum[][] from List[]. We analyse this algorithm by finding the sum of the arithmetic sequence. We recognise an insertion pattern in the algorithm and have the arithmetic sum:

$$n + (n-1) + (n-2) + ... + 1 = \sum_{k=1}^{n} k = \frac{n(n+1)}{2} \in O(n^2) \tag{7}$$

The worst-case runtime of ArraySum is therefore $O(n^2)$.

b) Analysis of Made-Up Algorithm

Given the information of the runtime of algorithm Bentley being $\Theta(n^2)$, and the fact that algorithm Murray is a nested for-loop inside of Bentley, this means the total runtime time of Murray is

$$n \times n \times n = \Theta(n^3) \tag{8}$$