Please enter your name and uID below.

Name: Sabath Rodriguez

uID: u1254500

Submission notes

- **(PS 1 specific) This is a warm-up assignment. Full points will be given for answers that have some correct elements and that clearly been given a legitimate effort.**

- *Solutions must be typeset* using one of the template files. For each problem, your answer must fit in the space provided (e.g. not spill onto the next page) *without* space-saving tricks like font/margin/line spacing changes.
- Upload a PDF version of your completed problem set to Gradescope.
- Teaching staff reserve the right to request original source/tex files during the semester, so please retain the original files.
- Please remember that for problem sets, collaboration with other students must be limited to a high-level discussion of solution strategies. If you do collaborate with other students in this way, you must identify the students and describe the nature of the collaboration. You are not allowed to create a group solution, and all work that you

hand in must be written in your own words. Do not base your solution on any other written solution, regardless of the source.

1. (Total Induction)
To prove by induction our 1st step is to solve a base, which in our case let's say it's n = 1.

That means that $\sum_{4}^{6} k = 4 + 5 + 6 = 15$. Now let's try n = 2.

$\sum_{8}^{12} k = 8 + 9 + 10 + 11 + 12 = 30$. Okay, now for our 2nd step, assume $\sum_{i}^{n} i = \frac{n(n+1)}{2}$.

Okay and by rules of summation we know that $1 + (\sum_{i=1}^{6n} k - \sum_{i=1}^{4n-1} k) = 1 + \sum_{i=4n}^{6n} k$. So now let's leave n as n.

1) $\sum_{i=1}^{6n} k = \frac{6n(6n+1)}{2}$ & $\sum_{i=1}^{4n-1} k = \frac{4n-1((4n-1)+1)}{2}$

2) $\frac{36n^2+6n}{2} - \frac{16n^2-4n}{2} = \frac{20n^2+10n}{2}$

3) $10n^2 + 5n + 1$

Okay and for our final step, let's plug in n = n + 1.

1) $\sum_{i=1}^{6n+1} k = \frac{6n+1(6(n+1)+1)}{2}$ & $\sum_{i=1}^{4(n+1)-1} k = \frac{4(n+1)-1((4(n+1)-1)+1)}{2}$

2) $\frac{6n+6(6n+7)}{2} - \frac{4n+3(4n+4)}{2} = \frac{20n^2+50n+30}{2}$

3) $10n^2 + 25n + 15 + 1$

All we need to do now is plug in n+1 into our closed form of $1 + \sum_{i=4n}^{6n} k$, which is $10n^2 + 5n + 1$.
This gives us,

1) $10(n + 1)^2 + 5(n + 1)$

2) $10n^2 + 25n + 15 + 1$, which is equal to $1 + \sum_{i=4n+1}^{6n+1} k$.

For our final step, we want to show that $f(n) = 10n^2 + 25n + 15$ is upper bounded by $g(n) = n^2$.

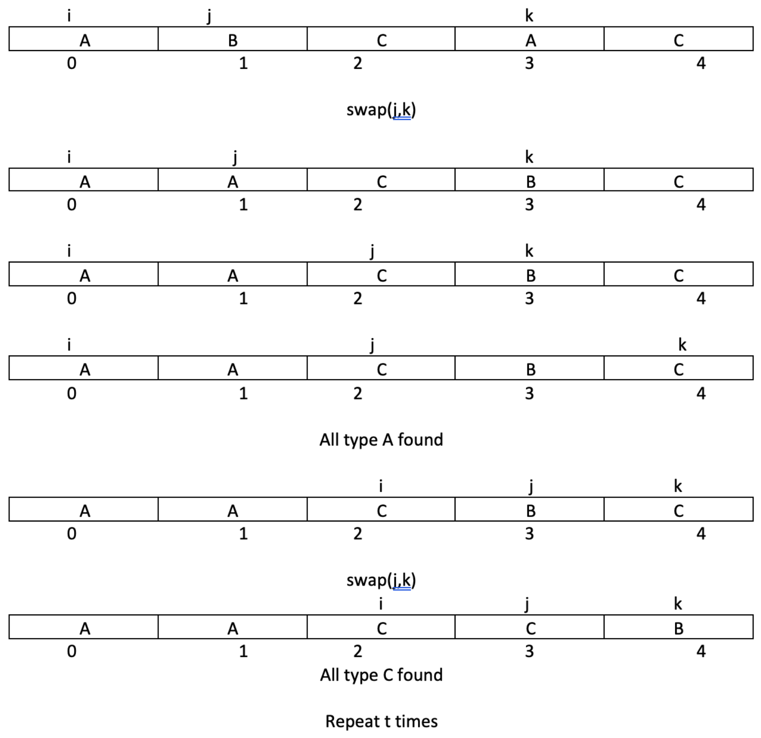
1) $f(n) \le c * g(n)$

2) $10n^2 + 25n + 16 \le c * n^2$

3) $10n^2 + 25n + 16 \le c * n * n$

4) $if\ c = 50,\ 10n^2 + 25n + 16 \le 50 * n * n,\ this\ is\ always\ true.$

2. (CattleSorting)
We start by picking the item/cattle at the first position, let's call this position i, and then find all elements/cattle of the same type. We first need to keep track of the index/position of the first element/cattle that is not the same as the first element/cattle, let's call this position *j*. Once we find the first element that's different, we continue iterating until we either find an element/cattle of the same type or reach the end of the array/cattles? In the case that we find a cattle of the same type as the one we initially chose, let's call this position *k*, we would then swap cattle at position *j* with the one at position *k,* and then increment position *j* by 1*.* Once we swap those two cattle, we would then continue iterating from position *k* to the end of our array/cattle and perform the same operation, that is if we find an element/cattle of the same type, then we swap cattle as position k with the one at position j. Once we are finished with all cattle of that type we then would set position i equal to position j, and repeat this process t times (t := # of types of cattle). Therefore, if there are n number of cattle, we know we would at the worse case, which is two of the same type are at opposite ends, we would have to go from 0 to n (number of cattle) for our first swap, and we would repeat this process by the number of types of cattle, which in our case is t. So our running time would be O(n*t).



3. (Vacations)
To prove that no remote employee takes a vacation before any in-person employees, I first set P = “no remote employee is given a vacation before an in-person” and therefore $\neg P$ = “a remote employee is given a vacation before an in-person”. We then try to prove for $\neg P$, giving us our contradiction ideally. First we should set r = “remote” and i = “in-person”. On line 4 we check if k = r and if k + 1 = i, meaning “is there a remote employee ahead of an in-person employee in our array?”, if there is, then we swap r and n, so *swap(E[k], [k+1]),* and we do that from [0,n-2], meaning we check from 0 to the last index (we have to go to n-2 because we check i+1). We also know that according to line 3, our second for loop, j, goes from [0,n-i-2], meaning that if we swap k = r with k + 1 = i, leaving r at + 1, we then don’t swap that element again. Our contradiction lies here, regardless of wherever there is an r before an i, we will always swap them, leaving r after every i. So, wherever there is a remote employee ahead of an in-person employee, we swap them and then shorten the amount of the elements we check on our next iteration by 1.

...	r	i	...
0	k	k+1	n