Name: Guangfeng Lin
Student #: 301312131

Assignment 1

1. $\sum_{i=0}^{n} (3i^3 - 6i + 2)$

$$P(n) = \sum_{i=0}^{n} (3i^3 - 6i + 2) = 2 + \sum_{i=1}^{n} (3i^3 - 6i + 2)$$

$$= 2 + 3\sum_{i=1}^{n} i^3 - 6\sum_{i=1}^{n} i + 2\sum_{i=1} 1$$

$$= 2 + 3\frac{n^2(n+1)^2}{4} - 6\frac{n(n+1)}{2} + 2n$$

$$= \frac{3}{4}n^4 + \frac{3}{2}n^3 + \frac{9}{4}n^2 - n + 2$$

prove RHS = LHS:

$$\text{RHS: } P(n) - P(n-1) = \frac{3}{4}n^4 + \frac{3}{2}n^3 + \frac{9}{4}n^2 - n + 2 - \left(\frac{3}{4}(n-1)^4 + \frac{3}{2}(n-1)^3 + \frac{9}{4}(n-1)^2 - (n-1) + 2\right)$$

$$= \frac{3}{4}n^4 + \frac{3}{2}n^3 + \frac{9}{4}n^2 - n + 2 - \frac{3}{4}(n-1)^4 - \frac{3}{2}(n-1)^3 - \frac{9}{4}(n-1)^2 + (n-1) - 2$$

$$= 3n^3 - 6n + 2$$

$$\text{LHS: } P(n) - P(n-1) = \sum_{i=0}^{n} (3i^3 - 6i + 2) - \sum_{i=0}^{n-1} (3i^3 - 6i + 2)$$

$$= \sum_{i=0}^{1} (3i^3 - 6i + 2)$$

$$= 3n^3 - 6n + 2$$

RHS = LHS

By induction sum = p(n) //

2. a) What is the asymptotic worst-case running time of Aerosoft ? show your work.

The worst-case running time of Aerosoft is $O(n^{\log_{\frac{4}{3}} 3})$.

$T(n)$: The time taken for input of size n. in $n = j - i + 1$

$m_1 = i + 3 * n/4$ $T(3n/4)$ call Aerosort 3 times. therefore, $T(n)$ of Aerosort = $T(n) = 3T(3n/4)$

$T(n) = 3T(3n/4) + O(1)$ for $n < 10$.

If $(n < 10)$, for $n < 10$ we do constant work. therefore $T(n) = O(1)$.

By master theorem, $a = 3$, $b = 4/3$ $c = 0$

Therefore, the worst-case running time is $O(n^{\log_{\frac{4}{3}} 3})$.

b) Prove that Aerosort $(A, 1, n)$ correctly sorts an array A of n elements.

prove by induction.

Base case : $n < 10$

Since sorting $A[i...j]$ by insertion-sort therefore the array is sorted.

Inductive Hypothesis : for all n, $n = j - i + 1$ Aerosort(A, i, j) sorts $A[i...j]$ is valid.

Inductive case: let the elements of the array be labeled as $[a1, a2, ..., an]$.

By calling the first recursion sort, $a1 <= a2, <= a3 ... <= a^{3n/4}$.

After the second recursion sort, $a^{n/4} <= ... <= an$.

If $a^{3n/4} <= a^{3n/4+1}$ then we can conclude that Aerosort $(A, 1, n)$ is correctly sorted.

After the first recursive call let the first half of the array be the smaller numbers than the second half of the array, therefore, after the second recursive call the sorted number of the second half of the array are the numbers less than the first half.

Therefore, proved //.

Scanned with CamScanner

```
3. Class arr {
       int max
       int min
   }

   Get-Max_Min (arr, int low, int high)
       int max , min ; mid.
       if (low == high)
           min = arr[low]
           max = arr[max]
       return min , max

       if (high == low+1)
           if (arr[low] > arr[high])
               max = arr[low]
               min = arr[high]
           else
               max = arr[high]
               min = arr[low]

       return min, max.

       mid=(low + high)/2

       arr. lefthalf = new arr
       arr. righ half = new arr
       arr array = new arr
       left_half. =      Get-Max_Min (arr, low, mid)
       righ_half. =      Get- Max_ Min (arr, mid+1, high).
       if(left_half.min < right_half. min)
           array. min = left_half.min.
       else
           array. min = right_half.min.

       if (left_half.max > right_half.max).
           array. max = left_half. max
       else
           array. max = right_half. max.

       return array.
```

4. ① Split the binary integer into two halfs, the less and the most significant half.

② The less significant half can be represented as l.

The most significant half can be represented as m.

③ Then the input $= m2^{\frac{\beta}{2}} + l$.

④ Using recursive function to convert l and m to be decimal.

⑤ Compute the decimal values 2^{2^i} up to 2^{β}

⑥ Since the count of number is $\lg(\beta)$, the runnig time is $O(\lg^2(\beta))$.

⑦ The relation of $T(n) = T(\beta) = 2T(\frac{\beta}{2}) + M(\frac{\beta}{2})$

The runing time is

$T(\beta) \in O(M(\beta)\lg(\beta))$.

Some binary number to decimal number will taby only $O(\beta)$ time then the algorithm will take $O(M(\beta)\lg(\beta)) \in \Omega(\beta\lg(\beta))$ to process.