Question 2

You are given an array A of n integers. You are required to find indices i, j, k (not necessarily distinct) such that $A[i] + A[j] = A[k]$, or return that no such indices exist.

Design algorithms which solve this problem and run in:

2.1 **[6 marks]** worst case $\Theta(n^2 \log n)$ time.

Answer:

- First, sort the array A
- Second, set the output loop i from 1 to n, $A[i]$ represents the value of position i
- in array A.
- Set the input loop j from $i+1$ to n, $A[j]$ represents the value of position j in
- array A.
- $p = A[i] + A[j]$
-
- Use dichotomy, set the range of the search array is $[x, y]$, now, $x = 1, y = n$.
- $m = \lfloor \frac{x+y}{2} \rfloor$
- If $p = A[m]$ the result found, A[i] + A[j] = A[m], end.
- If $p > A[m]$, the range become $[m+1, y]$ and then go back and start
- the next comparison.
- If $p < A[m]$, the range become $[x, m-1]$ and then go back and start
- the next comparison.
-
- If have result, both loop finish, If else, continue.

Reason:
First, the sort part time complexity is $\theta(nlogn)$.
Then, run a nested loop to count the sum of the two data, and use dichotomy to find whether the sum is in this array or not. The worst case of nested loop is $\theta(n^2)$ and the dichotomy method $\theta(logn)$. Therefore, the worst case is $\theta(n^2 logn)$.

2.2 **[6 marks]** *expected* $\Theta(n^2)$ time.

Answer:

Please refer to my solution to 2.3.

2.3 [**8 marks**] worst case $\Theta(n^2)$ time.

Answer:

- Set the output loop i from 1 to n, $A[i]$ represents the value of position i
- in array A.
- During every loop, create a new hash map M which record two integers.
- Set the input loop j from $i+1$ to n, $A[j]$ represents the value of position j in
- array A.
- If $A[i]-A[j]$ can find the result in M, the result found, $A[j]+A$[the value of key $A[i]-A[j]]=a[i]$, end.
- Else, set A[j] as key, position j as value, put the pair$\{A[j], j\}$ in M.
-
- If have result, both loop finish, If else, continue.

Reason:
This method uses nested loops, the first loop $A[i]$ become the sum target and create a new Hash map within each loop. The second loop searches for the results. In each loop, the difference between $A[i]$ and $A[j]$ is first searched in the Hash map as the key, the time complexity is $\theta(1)$. If there are records before, it means that these three values exist in the array. If not, set $A[j]$ as key, j as value, and put it in the hash map. The nested loop time complexity is $\theta(n^2)$