CSC165H1, Problem Set 4

[redacted] Youngsin Ryoo
[redacted]

Question 1

a) Loop 3 runs for $\lceil \frac{n}{2} \rceil$ iterations, each iteration costs 1 step, so then we have the total step being $\lceil \frac{n}{2} \rceil$

Loop 2 runs for $\lceil \log_2 i \rceil$ iterations, each iteration takes $\lceil \frac{n}{2} \rceil$ steps (since loop 3 takes $\lceil \frac{n}{2} \rceil$ steps for each iteration of loop 2; and line 6 and line 9 together take 1 step, but because 1 step is constant time, we just count $\lceil \frac{n}{2} \rceil$ steps for each iteration of loop 2 for simplicity). Thus, total steps is $\lceil \log_2 i \rceil \cdot \lceil \frac{n}{2} \rceil$

Loop 1 runs for $\lceil \log_2 n \rceil$ iterations, as i takes on values $2^0, 2^1, ...2^{\lceil \log_2 n \rceil - 1}$. In each iteration of loop 1, line 4 and line 10 take constant time so they are counted as 1 step, and loop 2 has a total of $\lceil \log_2 i \rceil \cdot \lceil \frac{n}{2} \rceil$ steps; since 1 step is constant time, we just count $\lceil \log_2 i \rceil \cdot \lceil \frac{n}{2} \rceil$ steps for each iteration of loop 1 for simplicity.
So total steps in loop 1 is:

$$
\begin{aligned}
&= \sum_{i=0}^{\lceil \log_2 n \rceil - 1} \lceil \log_2 2^i \rceil \cdot \lceil \frac{n}{2} \rceil \\
&= \lceil \frac{n}{2} \rceil \cdot \sum_{i=0}^{\lceil \log_2 n \rceil - 1} \lceil \log_2 2^i \rceil \\
&= \lceil \frac{n}{2} \rceil \cdot \sum_{i=0}^{\lceil \log_2 n \rceil - 1} \lceil i \rceil \\
&= \lceil \frac{n}{2} \rceil \cdot \sum_{i=0}^{\lceil \log_2 n \rceil - 1} i && \text{(since } i \text{ is an integer already)} \\
&= \lceil \frac{n}{2} \rceil \cdot \sum_{i=1}^{\lceil \log_2 n \rceil - 1} i && \text{(since adding } i = 0 \text{ doesn't change the sum)} \\
&= \lceil \frac{n}{2} \rceil \cdot \frac{(\lceil \log_2 n \rceil - 1) \cdot (\lceil \log_2 n \rceil)}{2} \\
&= \lceil \frac{n}{2} \rceil \cdot \frac{\lceil \log_2 n \rceil^2 - \lceil \log_2 n \rceil}{2}
\end{aligned}
$$

Line 2 takes 1 step. So the total number of steps of this algorithm is $1 + \lceil \frac{n}{2} \rceil \cdot \frac{\lceil \log_2 n \rceil^2 - \lceil \log_2 n \rceil}{2}$

Since the fastest growing term in the second term of the multiplication is $\lceil \log_2 n \rceil^2 = \lceil \log_2 n \rceil \cdot \lceil \log_2 n \rceil$, and the fastest growing term in the first term of the multiplication is n, we would have $\Theta(n \log_2 n \log_2 n)$

b) To begin, Loop 2 runs for i iterations. In each iteration, line 7 and line 8 take constant time that we count them as 1 step. Thus, the total number of steps for loop 2 is i.

In loop 3, it runs for $n-i$ iterations. In each iteration, line 11 and line 12 take constant time that we count them as 1 step. Thus, the total number of steps for loop 3 is $n - i$

In the outer loop, loop 1 takes n iterations ($i = 0, \cdots, n-1$). When i is odd, loop 2 runs; when i is even, loop 3 runs.

Case 1: n **is even:** In this case, there will be $\frac{n}{2}$ number of odd numbers from $0, 1, 2, ...n-1$, and $\frac{n}{2}$ number of even numbers from $0, 1, 2, ...n-1$ as well. So we take 0 to $\frac{n}{2} - 1$ for both the summations for odd and even i's.
We also count line 4, line 5 or 9 (depending on if i is even or odd), and line 13 as 1 step because they all take constant time - since 1 step is of lower order than the number of steps loop 2 or 3 takes, we will just count the number of steps from loop 2 and 3 for each iteration of loop 1 for simplicity.
Therefore, the number of total steps of loop 1 in this case is:

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

(first sum for the even numbers from $0, 1, ...n-1$, second sum for the odd numbers from $0, 1, ...n-1$)

$$= \sum_{i=0}^{(n/2)-1} (n - 2i + 2i + 1) \qquad \text{(combined summations for first and second)}$$

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

$$= \frac{n}{2} \cdot (n + 1)$$

$$= \frac{1}{2} \cdot (n^2 + n)$$

Now, outside of loop 1, line 2 takes 1 step and we can add that to $\frac{1}{2} \cdot (n^2 + n)$. However, in the expression for the total number of steps of this algorithm $\frac{1}{2} \cdot (n^2 + n) + 1$, n^2 is the fastest growing term. This will give us $\Theta(n^2)$ for the case where n is even.

Case 2: n **is odd:** In this case, there will be $\frac{n-1}{2}$ number of odd numbers from $0, 1, 2, ...n-1$, and $\frac{n-1}{2} + 1$ number of even numbers from $0, 1, 2, ...n-1$. So we take 0 to $\frac{n-1}{2} - 1$ for the odd summations and $\frac{n-1}{2} + 1 - 1 = \frac{n-1}{2}$ for the even summations.
We also count line 4, line 5 or 9 (depending on if i is even or odd), and line 13 as 1 step because they all take constant time - since 1 step is of lower order than the number of steps loop 2 or 3 takes, we will just count the number of steps from loop 2 and 3 for each iteration of loop 1 for simplicity.

Therefore, the total steps for this case is:

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

(first sum for the even numbers from $0, 1, ...n-1$, second sum for the odd numbers from $0, 1, ...n-1$)

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

(took $(n-2(\frac{n-1}{2}))$ from the first sum out)

$$= (\sum_{i=0}^{\frac{n-1}{2}-1}(n-2i+2i+1)) + (n-n+1) \qquad \text{(combined summations)}$$

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

$$= \frac{n-1}{2}(n+1) + 1$$

$$= \frac{n^2-1}{2} + 1$$

$$= \frac{1}{2}(n^2+1)$$

Now, outside loop 1, line 2 takes 1 step so the total number of steps of this algorithm is $\frac{1}{2}(n^2+1)+1$. However, n^2 is the fastest growing term in the whole expression. This will give us $\Theta(n^2)$ for the even case.

Both cases conclude in the same theta bound, and thus, the theta bound on the running time for this function is $\Theta(n^2)$.

c) In the previous question, we examined the running time of cases where n is odd or even. When we subtract 1 from each expression in the 2 cases (subtracting 1 because 1 is the number of step that the line (line 2) outside loop 1 takes) resembles the amount of print statements that get outputted from the function.
Since in loop 1, we counted line 4, line 5 or 9 (depending on if i is even or odd), and line 13 as 1 step because they all take constant time; and in loop 2, we counted line 7 and line 8 as 1 step since they take constant time; similarly, in loop 3, we counted line 11 and 12 as 1 step. We know that for each iteration i of loop 1, the print statements are generated i times in loop 2 (if line 5 evaluates to True), and $n-i$ times in loop 3 (if line 5 evaluates to False), and loop 1 has n iterations, so the number of print statements executed is the same as the number of steps we got in 1b). Thus we have Case 1 of n is even generating $\frac{1}{2} \cdot (n^2+n)$ print statements. Case 2 of n is odd generating $\frac{1}{2}(n^2+1)$ print statements.

Question 2

a) Let $n = \text{len}(\text{lst})$, $n \in \mathbb{Z}^+$

For a fixed iteration in loop 1, loop 2 runs for at most i iterations, $j = 0, 1, 2, ..., i-1$. i is the loop variable for loop 1. We know that each iteration takes 1 step (since line 5 and line 6 both take constant time), so the total steps is at most i.

In loop 1, it runs for at most n iterations, $i = 0, 1, 2, ..., n-1$. Every iteration takes at most i steps, from previously.

Total number of steps at most:

$$\begin{aligned} &= \sum_{i=0}^{n-1} i \\ &= \frac{n(n-1)}{2} \\ &= \frac{n^2 - n}{2} \end{aligned}$$

The line of "return False" takes 1 step, then the total steps is at most $1 + \frac{n^2-n}{2}$ which is $\mathcal{O}(n^2)$ as n^2 is the fastest growing term that we can disregard the lower order terms.

b) Let $n \in \mathbb{Z}^+$, $\text{len}(\text{lst}) = n$, and let $\text{lst} = [0, 1, ..., n-1]$ where the elements are natural numbers from 0 to $n-1$ in ascending order. Let $s = -100$.

Then, we would have $i = 0, 1, ..., n-1$, if we fix i, then $j = 0, 1, ..., i-1$. Thus, $\text{lst}[i] = i$, $\text{lst}[j] = j$, and $\text{lst}[i] + \text{lst}[j] = i + j$.

From the definition of i, i is $n-1$ at most. From the definition of j, j is at most $i - 1 = n - 1 - 1 = n - 2$. So then, $i + j = n - 1 + n - 2 = 2n - 3$ at most.

Because the elements in the lst are in ascending order from 0 to $n-1$, $\text{lst}[i] + \text{lst}[j]$ will never be negative, which means it will never be equal to -100. So, the loops will go through all the elements and the function will terminate when it hits line 7, which is "return False".

In this case, total number of steps:

$$\begin{aligned} &= \sum_{i=0}^{n-1} i + 1 && \text{(adds 1 step from "return False")} \\ &= \frac{n(n-1)}{2} + 1 \\ &= \frac{n^2 - n}{2} + 1 \end{aligned}$$

Then, the runtime function on this lst and s takes $\frac{n^2-n}{2} + 1$, which is $\Omega(n^2)$ as n^2 is the fastest growing term that we can disregard lower order terms. This matches the upper bound in a).

c) Let $n \in \mathbb{Z}^+$.
Let $s = \lceil \sqrt{n} \rceil$.
Let lst be a list of length n where every element is 0 except for $(\lceil \sqrt{n} \rceil + 1)^{th}$ element. The $(\lceil \sqrt{n} \rceil + 1)^{th}$ will be equal to $(\lceil \sqrt{n} \rceil)$.
Therefore, our input family will be [0, 0, ..., $\lceil \sqrt{n} \rceil$, ..., 0]
Want to show that $RT_{some} \in \Theta(\text{len}(\text{lst}))$
i.e. $\Theta(n)$

For loop 1:
When $i = 0, \ldots, \lceil \sqrt{n} \rceil$: for each iteration i of loop 1, loop 2 will need to iterate i times since lst[i] + lst[j] is always equal to 0 and not equal to s.
When $i = \lceil \sqrt{n} \rceil + 1$, lst[i] $= \lceil \sqrt{n} \rceil$. So the function will terminate at $j = 0$ in loop 2, because lst[0] = 0, and so lst[i] + lst[j] $= \lceil \sqrt{n} \rceil$.

Then, we have the total running time:

$$= \sum_{i=0}^{\lceil \sqrt{n} \rceil} i + 1$$

(adds 1 since when $i = \lceil \sqrt{n} \rceil + 1$, loop 2 iterates once ($j = 0$) and returns True)

$$= \frac{\lceil \sqrt{n} \rceil (\lceil \sqrt{n} \rceil + 1)}{2} + 1$$

$$= \frac{n + \lceil \sqrt{n} \rceil}{2} + 1$$

Since we can disregard the lower order terms that take constant time, we can get rid of $\frac{1}{2}$ in the denominator from the first term and the second term, 1.
By properties of ceiling, we know that $\lceil \sqrt{n} \rceil \in \Omega(\sqrt{n})$. Also, we know that $\lceil \sqrt{n} \rceil < \sqrt{n} + 1$. This means that $\lceil \sqrt{n} \rceil \in \mathcal{O}(\sqrt{n})$.
For the first term, n, we know that it is $\in \Theta(n)$. Because $1 > \frac{1}{2}$, we know that the second term, $\lceil n^{\frac{1}{2}} \rceil$, is $\in \mathcal{O}(n)$.
Then, $n + \lceil \sqrt{n} \rceil \in \Theta(n)$.
Finally, we can conclude that the $RT_{some} \in \Theta(n)$, which is $\Theta(\text{len}(\text{lst}))$.

Question 3

a) **Part 1: Find upper bound**

Let $n = \text{len}(\text{lst})$, $n \in \mathbb{Z}^+$ since the precondition is that lst is a non-empty list of integers.

For a fixed iteration i in loop 1, and fixed iteration j in loop 2, loop 3:
- has $j - i$ iterations $(k = i, \cdots, j - 1)$

- each iteration takes 2 steps at most (when "if lst[j] − lst[k] < d" evaluates to True)
- total number of steps at most is $2(j-i)$

For a fixed iteration i in loop 1, loop 2:
- has $n-i-1$ iterations ($j = i+1, \cdots, n-1$)
- each iteration takes at most $2(j-i)$ steps - total number of steps at most is:

$$
\begin{aligned}
\sum_{j=i+1}^{n-1} 2(j-i) &= 2\sum_{j=i+1}^{n-1}(j-i) \\
&= 2(\sum_{j=i+1}^{n-1} j - \sum_{j=i+1}^{n-1} i) \\
&= 2(\sum_{j=1}^{n-1-i} j + i - \sum_{j=1}^{n-1-i} i) && \text{(change of index)} \\
&= 2(\sum_{j=1}^{n-1-i} j + \sum_{j=1}^{n-1-i} i - \sum_{j=1}^{n-1-i} i) \\
&= 2\sum_{j=1}^{n-1-i} j \\
&= 2 \cdot \frac{(n-i-1)(n-i-1+1)}{2} && \text{(since } j \in \mathbb{N}) \\
&= (n-i-1)(n-i)
\end{aligned}
$$

For a fixed iteration i in loop 1, loop 4:
- has $n-i-1$ iterations ($j = i+1, \cdots, n-1$)
- each iteration takes 1 step
- total number of steps is $n-i-1$

For a fixed iteration i in loop 1, in the case where the if condition in line 5 of the code evaluates to True, loop 2 and loop 4 will run, therefore the total number of steps at most is:

$$
\begin{aligned}
(n-i-1)(n-i) + (n-i-1) &= (n-i-1)(n-i+1) \\
&= (n-i)^2 - 1
\end{aligned}
$$

For a fixed iteration i in loop 1, loop 5:
- has at most $n-i-1$ iterations, when lst[j] > 0 for all j ($j = i+1, \cdots, n-1$)
- each iteration takes 1 step
- total number of steps at most is $n-i-1$

For a fixed iteration i in loop 1, in the case where the if condition in line 5 of the code evaluates to False, the else block starting from line 13 of the code will execute, j = i + 1 takes 1 step, then loop 5 will run, which has a total number of steps at most $n-i-1$. Therefore, the total number of steps when the else block executes is $n-i$

Since $(n-i)$ has a lower order than $(n-i)^2-1$, for a fixed iteration i in loop 1, the total number of steps at most is $(n-i)^2-1$

Loop 1:
- has $(n-1)$ iterations $(i=0,1,\cdots,n-2)$
- each iteration takes at most $(n-i)^2-1$ steps
- total number of steps at most is:

$$\begin{aligned}
\sum_{i=0}^{n-2}((n-i)^2-1) &= \sum_{i=0}^{n-2}((n^2-2ni+i^2)-1) \\
&= \sum_{i=0}^{n-2}(n^2-1)+\sum_{i=0}^{n-2}(i^2-2ni) \\
&= (n-1)(n^2-1)-2n\sum_{i=0}^{n-2}i+\sum_{i=0}^{n-2}i^2 \\
&= (n-1)(n^2-1)-2n\cdot\frac{(n-2)(n-1)}{2}+\sum_{i=1}^{n-2}i^2 \\
&\qquad\qquad\text{(since } i\in\mathbb{N}\text{, and } i^2=0 \text{ when } i=0) \\
&= (n-1)(n^2-1)-n(n-2)(n-1)+\frac{(n-2)(n-1)(2n-3)}{6} \\
&\qquad\qquad\text{(since } \textstyle\sum_{i=1}^{n} i^2=\frac{n(n+1)(2n+1)}{6}, i\in\mathbb{N}) \\
&= \frac{1}{3}n^3+\frac{1}{2}n^2-\frac{5}{6}n
\end{aligned}$$

Therefore, the total number of steps of this algorithm is at most $\frac{1}{3}n^3+\frac{1}{2}n^2-\frac{5}{6}n$, which is $\mathcal{O}(n^3)$

Part 2: Find lower bound

Let $n \in \mathbb{Z}^+$

Let len(lst) = n, and let lst = [2, ⋯, 2n], i.e., ∀i in range(n), lst[i] = 2(i+1), so the if condition in line 5 of the code evaluates to True for every element in the lst. Therefore, loop 2, loop 3 and loop 4 will always execute.

For a fixed iteration j in loop 2, loop 3:
- has $j-i$ iterations $(k=i,\cdots,j-1)$
- each iteration takes 1 step at least (when "if lst[j] − lst[k] < d" always evaluates

to False)
- total number of steps at least is $(j - i)$

For a fixed iteration i in loop 1, loop 2:
- has $n - i - 1$ iterations $(j = i + 1, \cdots, n - 1)$
- each iteration takes at least $(j - i)$ steps
- total number of steps at least is:

$$
\begin{aligned}
\sum_{j=i+1}^{n-1}(j-i) &= \sum_{j=i+1}^{n-1} j - \sum_{j=i+1}^{n-1} i \\
&= \frac{(n-i-1)(n-i)}{2} \qquad \text{(similar to the calculation in Part 1)}
\end{aligned}
$$

Loop 1:
- has $(n - 1)$ iterations $(i = 0, 1, \cdots, n - 2)$
- each iteration takes at least $\frac{(n-i-1)(n-i)}{2}$ steps
- total number of steps at least is:

$$
\begin{aligned}
\sum_{i=0}^{n-2}\frac{(n-i-1)(n-i)}{2} &= \frac{1}{2}\sum_{i=0}^{n-2}(n^2 - 2in - n + i + i^2) \\
&= \frac{1}{2}\sum_{i=0}^{n-2}(n^2 - n) + \frac{1}{2}\sum_{i=0}^{n-2}(i^2) + \frac{1}{2}\sum_{i=0}^{n-2}(i) + \frac{1}{2}\sum_{i=0}^{n-2}(-2in) \\
&= \frac{1}{2}(n-1)(n^2 - n) + \frac{1}{2}\sum_{i=0}^{n-2}(i^2) + \frac{1}{2}\sum_{i=0}^{n-2}(i) - n\sum_{i=0}^{n-2}(i) \\
&= \frac{1}{2}(n-1)(n^2 - n) + \frac{1}{2}\frac{(n-2)(n-1)(2n-3)}{6} + (\frac{1}{2} - n)\frac{(n-2)(n-1)}{2} \\
&\qquad \text{(since } \textstyle\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}\text{)} \\
&= \frac{1}{2}(n-1)[(n^2 - n) + \frac{1}{6}(n-2)(2n-3) + (\frac{1}{2} - n)(n-2)] \\
&= \frac{1}{2}(n-1)(\frac{1}{3}n + \frac{1}{3}n^2) \\
&= \frac{1}{6}(n^3 - n)
\end{aligned}
$$

Therefore, the total number of steps of this algorithm is at least $\frac{1}{6}(n^3 - n)$, which is $\Omega(n^3)$
So the total number of steps is $\Theta(n^3)$

b) Let $RT_{func}(x)$ be the running time of executing func(x)
g is an upper bound on $BC_{func}(n)$ iff $\forall n \in \mathbb{N}, \exists x \in \mathcal{I}_n, RT_{func}(x) \leq g(n)$

h is a lower bound on $BC_{func}(n)$ iff $\forall n \in \mathbb{N}, \forall x \in \mathcal{I}_n, RT_{func}(x) \geq h(n)$

Part 1: Find lower bound

Let $n \in \mathbb{Z}^+$ since the precondition is that lst is a non-empty list of integers and $n =$ len(lst).

For a fixed iteration i in loop 1, and fixed iteration j in loop 2, loop 3:
- has $j - i$ iterations ($k = i, \cdots, j-1$)
- each iteration takes 1 step at least (when "if lst [j] − lst [k] < d" evaluates to False)
- total number of steps at least is $(j - i)$

For a fixed iteration i in loop 1, loop 2:
- has $n - i - 1$ iterations ($j = i+1, \cdots, n-1$)
- each iteration takes at least $(j - i)$ steps
- total number of steps at least is:

$$\begin{aligned} \sum_{j=i+1}^{n-1}(j-i) &= \sum_{j=i+1}^{n-1} j - \sum_{j=i+1}^{n-1} i \\ &= \frac{(n-i-1)(n-i)}{2} \end{aligned}$$

For a fixed iteration i in loop 1, loop 4:
- has $n - i - 1$ iterations ($j = i+1, \cdots, n-1$)
- each iteration takes 1 step
- total number of steps is $n - i - 1$

For a fixed iteration i in loop 1, in the case where the if condition in line 5 of the code evaluates to True, loop 2 and loop 4 will run, therefore the total number of steps at least is:

$$\frac{(n-i-1)(n-i)}{2} + (n-i-1) = \frac{1}{2}(n-i-1)(n-i+2)$$

For a fixed iteration i in loop 1, loop 5:
- has at least 0 iterations, when lst [j] > 0 evaluates to False for all j ($j = i+1, \cdots, n-1$)
- total number of steps at least is 0

For a fixed iteration i in loop 1, in the case where the if condition in line 5 of the code evaluates to False, the else block starting from line 13 of the code will execute, j = i + 1 takes 1 step, then loop 5 will run, which has a total number of steps at least 0. Therefore, the total number of steps when the else block executes is at least 1.

Since 1 has a lower order than $\frac{1}{2}(n-i-1)(n-i+2)$, for a fixed iteration i in loop 1, the total number of steps at least is 1.

Loop 1:
- has $(n-1)$ iterations $(i = 0, 1, \cdots, n-2)$
- each iteration takes at least 1 step
- total number of steps at least is:

$$\sum_{i=0}^{n-2} 1 = n - 1$$

Therefore, the total number of steps of this algorithm is at least $n-1$, which is $\Omega(n)$

Part 2: Find upper bound

Let $n \in \mathbb{Z}^+$
Let len(lst) = n, and let lst = [-2n+1, -2n+3, -2n+5, ⋯, -1],
i.e., ∀i in range(n), lst[i] = −2n + (2i+1), so the if condition in line 5 of the code evaluates to False for every element in the lst.
Therefore, loop 2, loop 3 and loop 4 will never execute.

Therefore, the else block starting from line 13 will execute, and j=i+1 takes 1 step.
Since every element in the lst is also negative, loop 5 will never execute.
Therefore, for a fixed iteration i in loop 1, there is 1 step.

So Loop 1:
- has $n-1$ iterations $(i = 0, 1, \cdots, n-2)$
- each iteration takes 1 step
- total number of steps is $n-1$

Therefore, the total number of steps of this algorithm is $\mathcal{O}(n)$.
So the total number of steps is $\Theta(n)$.