Tutorial 01 - Basic C++, Basic OOP, Analysis

CS2040C Semester 1 2020/2021

By Wira, AY2021 S1, adapted from slides by Jin Zhe, AY1920 S1, Ranald, AY1718 S1/S2

Question 1

(Basic) List ADT

List Array ADT

Imagine *list* as a *chain* of beads

You can add beads to the front and back

(and in between)

We will elaborate more about "**ADT**" in Tut 02.

For now, understand the syntax of C++ classes.

List Array ADT - Methods

`get(i)`

Gets the *i*-th element from the front (0-indexed)

`search(v)`

Return the **first** index which contains v

List Array ADT - Methods

`insert(i, v)`

Insert element *v* at index *i*.

`remove(i)`

Remove the element at index *i*.

List Array ADT - Methods

`printList()`

Prints the list from front to back.

`sortList()`

Sort the list, in default ascending order.

Question 1a, 1b, 1c

```
class ListArray {
  private:
    int N;
    int A[10];              // question 1a
  public:
    ListArray() : N(0) {} // question 1b
    int get(int i) {
        return A[i];        // question 1c
    }
}
```

Question 1a

```
class ListArray {
  private:
    int N;
    int A[10];
  public:
    ListArray() : N(0) {}
    int get(int i) {
        return A[i];
    }
}
```

Anything wrong with this line?

Question 1a

```
class ListArray {
  private:
    int N;
    int A[10];
  public:
    ListArray() : N(0) {}
    int get(int i) {
        return A[i];
    }
}
```

Anything wrong with this line?

Limited to **10** items!

Question 1b

```
class ListArray {
  private:
    int N;
    int A[10];
  public:
    ListArray() : N(0) {}
    int get(int i) {
        return A[i];
    }
}
```

What does this line mean?

Question 1b

```
class ListArray {
  private:
    int N;
    int A[10];
  public:
    ListArray() : N(0) {}
    int get(int i) {
        return A[i];
    }
}
```

What does this line mean?

Pass `0` to constructor of integer `N`. Effectively:

```
ListArray() {
    N = 0;
}
```

Question 1c

```
class ListArray {
  private:
    int N;
    int A[10];
  public:
    ListArray() : N(0) {}
    int get(int i) {
        return A[i];
    }
}
```

Anything potential issues with this line?

Question 1c

```
class ListArray {
  private:
    int N;
    int A[10];
  public:
    ListArray() : N(0) {}
    int get(int i) {
        return A[i];
    }
}
```

Anything potential issues with this line?

No “safeguard”!

What if `i > N - 1`?
What if `i` is negative?

Question 1d, 1e, 1f

```
void insert(int i, int v) {
  if ((N == 10) || (i < 0) || (i > N))  // question 1d
    return;
  for (int j = i; j <= N-1; j++)        // question 1e
    A[j+1] = A[j];
  A[i] = v;
  N++;
}

void remove(int i) {
  for (int j = i; j < N-1; j++)         // question 1f
    A[j] = A[j+1];
  N--;
}
```

Question 1d

```
void insert(int i, int v) {
  if ((N == 10) || (i < 0) || (i > N))
    return;
  for (int j = i; j <= N-1; j++)
    A[j+1] = A[j];
  A[i] = v;
  N++;
}
```

What does this line mean?

Question 1d

```
void insert(int i, int v) {
  if ((N == 10) || (i < 0) || (i > N))
    return;
  for (int j = i; j <= N-1; j++)
    A[j+1] = A[j];
  A[i] = v;
  N++;
}
```

What does this line mean?

If …
`N` is `10` or
`i` is negative or
`i > N`

Stop inserting.

Question 1d

```
void insert(int i, int v) {
  if ((N == 10) || (i < 0) || (i > N))
    return;
  for (int j = i; j <= N-1; j++)
    A[j+1] = A[j];
  A[i] = v;
  N++;
}
```

What does this line mean?

If …
`N` is `10` or
`i` is negative or
`i > N`

Stop inserting.

Is there anything wrong with this?
How many possible values of `i` are accepted?

Question 1d

```
void insert(int i, int v) {
  if ((N == 10) || (i < 0) || (i > N))
    return;
  for (int j = i; j <= N-1; j++)
    A[j+1] = A[j];
  A[i] = v;
  N++;
}
```

What does this line mean?

If …
`N` is `10` or
`i` is negative or
`i > N`

Stop inserting.

Is there anything wrong with this?
How many possible values of `i` are accepted?

N+1 possible insertion points

Question 1e

```
void insert(int i, int v) {
  if ((N == 10) || (i < 0) || (i > N))
    return;
  for (int j = i; j <= N-1; j++)
    A[j+1] = A[j];
  A[i] = v;
  N++;
}
```

Any potential issues with this line?

Question 1e

```
void insert(int i, int v) {
  if ((N == 10) || (i < 0) || (i > N))
    return;
  for (int j = i; j <= N-1; j++)
    A[j+1] = A[j];
  A[i] = v;
  N++;
}
```

Any potential issues with this line?

Values are being overwritten in the wrong order!

```
A[i+1] = A[i];
A[i+2] = A[i+1];
A[i+3] = A[i+2];
```

… etc

Question 1f

```
void remove(int i) {
  for (int j = i; j < N-1; j++)
    A[j] = A[j+1];
  N--;
}
```

Any potential issues with this line?

Question 1f

```
void remove(int i) {
  if (i < 0 || i >= N)
    return;
  for (int j = i; j < N-1; j++)
    A[j] = A[j+1];
  N--;
}
```

Any potential issues with this line?

No. This is correct.

```
A[i]   = A[i+1];
A[i+1] = A[i+2];
A[i+2] = A[i+3];
```

… etc;

Can also add safeguard `i` in [0..N-1]

Question 1g

```
void sortList() { // sort array A, question 1g
    // ...
}
```

Implement this routine using any sorting algorithm that you know!

Question 1g

```
void sortList() { // sort array A, question 1g
    sort(A, A+N);
}
```

Implement this routine using any sorting algorithm that you know!

Approach: Use std library

Question 1h

```
int main() {
    ListArray* LA = new ListArray();
    LA->insert(0, 5);
    LA->insert(0, 1);
    // ...
}
```

Can we just write `ListArray LA;` in this line?

Question 1h

```
int main() {
    ListArray* LA = new ListArray();
    LA->insert(0, 5); // (*LA).insert(0,5)
    LA->insert(0, 1);
    // ...
}
```

Can we just write `ListArray LA;` in this line?

Yes, but we need to modify the way we call methods:

```
ListArray LA;
LA.insert(0, 5);
cout << LA.get(1) << endl;
```

Question 1h

What's the difference between the two methods?

```
ListArray LA;
LA.insert(0, 5);
cout << LA.get(1) << endl;
```

```
ListArray *LA = new ListArray();
LA->insert(0, 5);
cout << LA->get(1) << endl;
```

Question 1g

```
void sortList() { // sort array A, question 1g
    for (int i = 0; i < N; i++) {
        for (int j = 1; j < N; j++) {
            if (A[j-1] > A[j])
                swap(A[j-1], A[j]);
        }
    }
}
```

Implement this routine using any sorting algorithm that you know!

Approach: manual implementation

Question 3

Analysis/ Order of Growth

Complexity analysis

- Is a rough estimate of how execution time will grow with size of input. I.E. *Order of growth*

- *Time complexity* is commonly used as a metric for comparing the performance of different algorithms on the same task

- When order of growth is applied to measure **memory consumption** of algorithms, we call that *space complexity*

Two rules

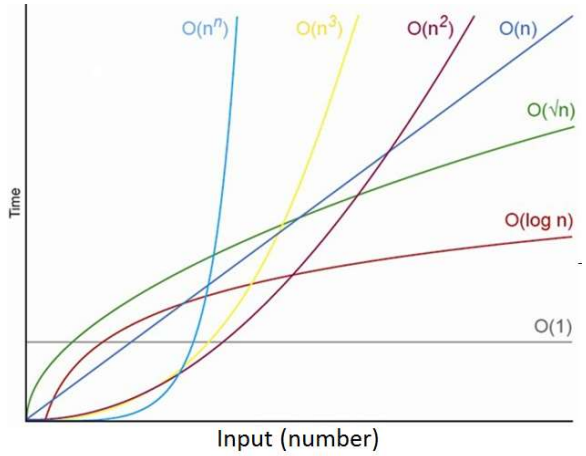
Two rules:

1. Constants can be ignored

   `O(3n) = O(n)`

2. Smaller complexity terms can be ignored

   `O(n + log(n) + n²) = O(n²)`



Real life application?

Single thread computer: ~10^8 operations/sec

Sorting **N = 10^6** numbers

$O(n^2)$ → ~ 2.5 hours

$O(n \times \log(n))$ → ~ 0.2 seconds

Kattis runs submissions at around 4×10^8 operations/second.

Question 3

What is the bound to the following functions?

a. $F(n) = \log(2^n) + \sqrt{n} + 100\,000\,000$

b. $F(n) = n + \frac{1}{2}n + \frac{1}{3}n + \frac{1}{4}n + \ldots + 1$

c. $G(n) = n + \frac{1}{2}n + \frac{1}{4}n + \frac{1}{8}n + \ldots + 1$

Question 3.a)

$F(n) = \log(2^n) + \sqrt{n} + 100\,000\,000$

$O(F(n)) = O(\log(2^n) + \sqrt{n} + 100\,000\,000)$

$= O(n * \mathbf{log\ 2} + \sqrt{n})$

$= O(n + \sqrt{n})$

$= \mathbf{O(n)}$

Question 3.b)

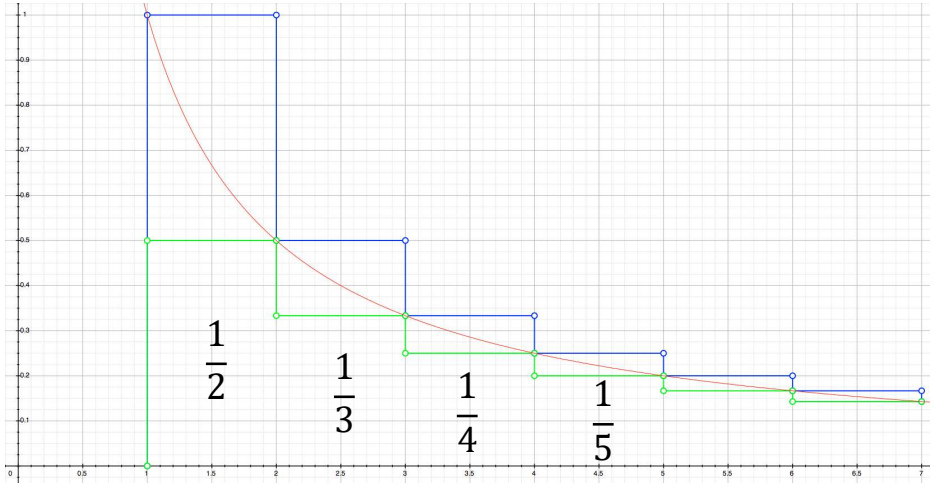
$$F(n) = n + \tfrac{1}{2} n + \tfrac{1}{3} n + \tfrac{1}{4} n + \ldots + 1$$

$$O(F(n)) = O(n + \tfrac{1}{2} n + \tfrac{1}{3} n + \tfrac{1}{4} n + \ldots + 1)$$

$$= O(n \underbrace{(1 + \tfrac{1}{2} + \tfrac{1}{3} + \tfrac{1}{4} + \ldots + 1/n)}_{\text{How to deal with this?}})$$

Harmonic Series

$$\sum_{n=1}^{\infty} \frac{1}{n} = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \cdots$$



Using integration:

$$\log(N+1) < \sum_{n=1}^{N} \frac{1}{n} = 1 + \sum_{n=1}^{N} \frac{1}{n+1} < 1 + \int_{1}^{N} \frac{dx}{x+1} = 1 + \log(N+1) = O(\log N)$$

You can try prove the lower bound as well.

Question 3.b)

From Harmonic Series

$O(1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \ldots + 1/n) = O(\log n)$

$O(F(n)) = O(n\,(1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \ldots + 1/n))$

$= \mathbf{O(n \log n)}$

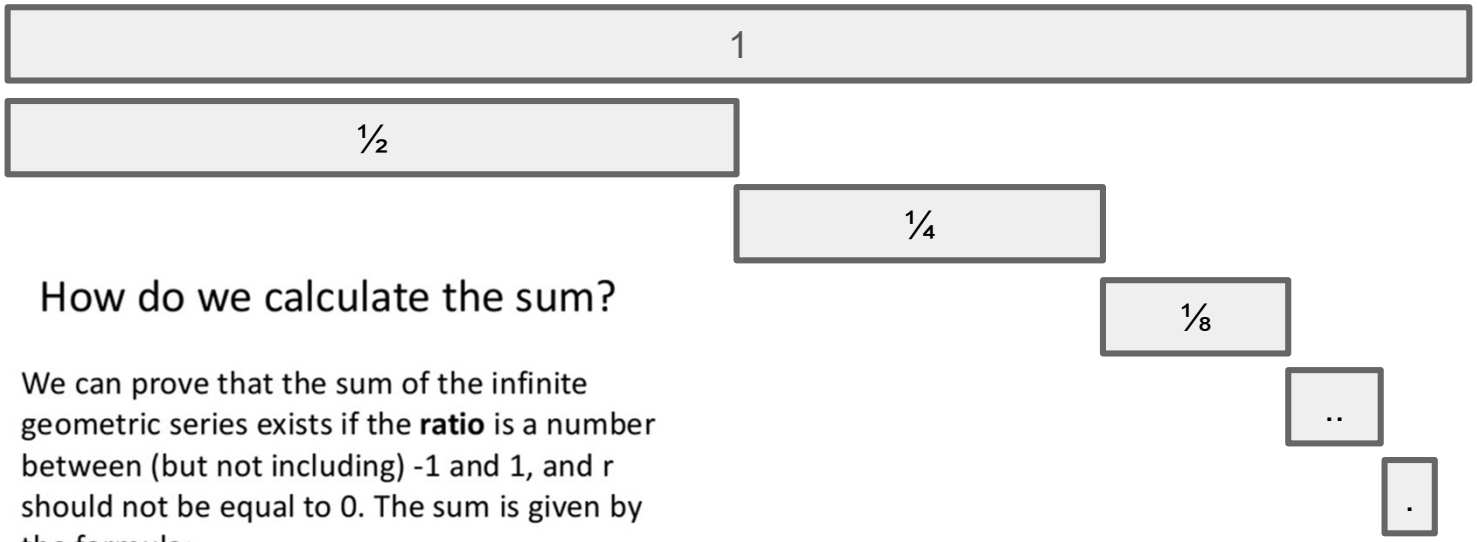
Question 3.c)

$$G(n) = n + \tfrac{1}{2} n + \tfrac{1}{4} n + \tfrac{1}{8} n + \ldots + 1$$

$$O(G(n)) = O(n + \tfrac{1}{2} n + \tfrac{1}{4} n + \tfrac{1}{8} n + \ldots + 1)$$

$$= O(n (\underbrace{1 + \tfrac{1}{2} + \tfrac{1}{4} + \tfrac{1}{8} + \ldots + 1/n}_{\text{How to deal with this?}}))$$

Convergent Geometric Series



How do we calculate the sum?

We can prove that the sum of the infinite geometric series exists if the **ratio** is a number between (but not including) -1 and 1, and r should not be equal to 0. The sum is given by the formula:

$$\sum_{k=0}^{\infty} ar^k = a + ar + ar^2 + ar^3 + \cdots = \frac{a}{1-r}$$

Question 3.c)

From Convergent Geometric Series

$O(1 + ½ + ¼ + ⅛ + \ldots + 1/n) = O(1 / (1 - ½))$

$= O(2)$

$= O(1)$

$O(G(n)) = O(n\,(1 + ½ + ¼ + ⅛ + \ldots + 1/n))$

Additional questions

Example (AY17/18 S1 Midterm Paper)

What's the time complexity?

```
int N, counter = 0;
cin >> N;
for (int i = N; i >= 1; i--) {
    for (int j = 1; j <= N/i; j++) {
        counter++;
    }
}
cout << counter << endl;
```

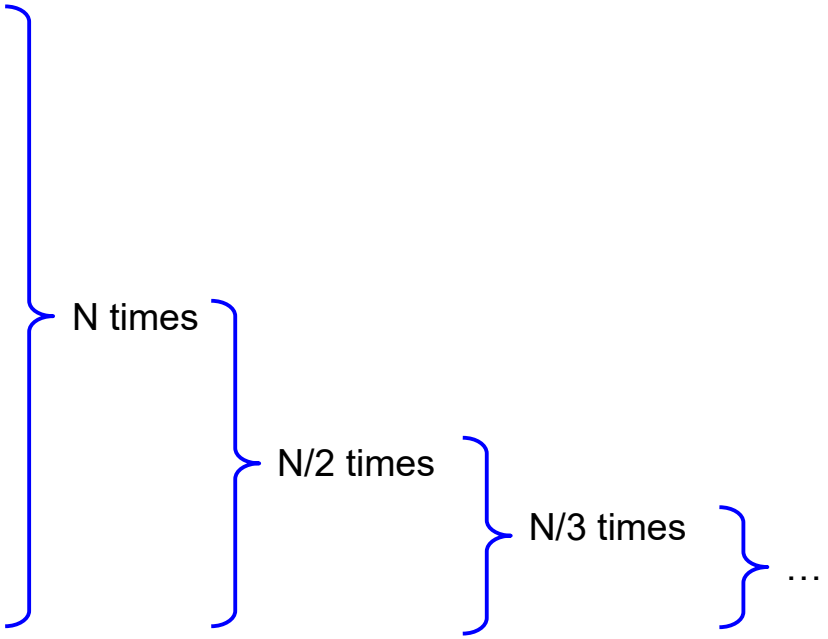
What's the time complexity?

```
int N, counter = 0;
cin >> N;
for (int i = N; i >= 1; i--) {
    for (int j = 1; j <= N/i; j++) {
        counter++;
    }
}
cout << counter << endl;
```

O(N log N)

Analysis

- i == N, j == 1
- i == N-1, j == 1
- i == N-2, j == 1
- ...
- i == N/2, j == 1, j == 2
- ...
- i == N/3. j == 1, j == 2, j == 3
- ...
- i == 1, j == 1 to j == N



$$N + N/2 + N/3 + N/4 + N/5 + N/6 + \ldots + 1$$

What's the time complexity?

```
int N, counter = 0;
cin >> N;
for (int i = 0; i < N; i++) {
    for (int j = 0; j < i; j++) {
        counter += j;
    }
}
cout << counter << endl;
```

What's the time complexity?

```
int N, counter = 0;
cin >> N;
for (int i = 0; i < N; i++) {
    for (int j = 0; j < i; j++) {
        counter += j;
    }
}
cout << counter << endl;
```

$O(N^2)$

What's the time complexity?

```
int N, counter = 0;
cin >> N;
for (int i = 0; i < N; i++) {
    for (int j = 0; j < N; j++) {
        counter++;
        i++;
    }
}
cout << counter << endl;
```

What's the time complexity?

```
int N, counter = 0;
cin >> N;
for (int i = 0; i < N; i++) {
    for (int j = 0; j < N; j++) {
        counter++;
        i++;
    }
}
cout << counter << endl;
```

O(N)

Questions

/basicprogramming1