Bryce Jarboe
RedID 825033151
10/17/2022
Prof. Shen | CS210

Assignment 2 Part B

Table of Contents

Legend

Constant Time Operation Input-Size N-Variable Operation <-- N/A, all are in constant time

Analysis Notes **# (added operation count)**

Push

Time Complexity Analysis

```
void PromotedCarModelStack::push(string model, int price) {
  PromotedModel pushModel(model, price);                    +1
  promoStack.push_back(pushModel);                          +1
  pair<PromotedModel, PromotedModel> pushPair;              +1
   if (trackStack.empty                    +1      Worst case looks towards the larger else statement
   pushPair = make_pair(pushModel, pushModel);                                  after comparison
  }
  else{
   if (pushModel.getPromotedPrice() < trackStack.back().first.getPromotedPrice()){      +1
     pushPair = make_pair(pushModel, trackStack.back().second);
   }
   else if (pushModel.getPromotedPrice() > trackStack.back().second.getPromotedPrice()){  +1
     pushPair = make_pair(trackStack.back().first, pushModel);
   }
   else{
     pushPair = make_pair(trackStack.back().first, trackStack.back().second);          +1
   }
  }
  trackStack.push_back(pushPair);     +1
}
```

Due to all operations remaining constant, counting is unneeded since, for any arbitrary constant **k**, **k * O(1) = O(1)**

Therefore, the Time Complexity f(N) *yielded can be expressed as*

f(N) = O(1)

Push

Auxiliary Space Complexity Analysis

Memory (non-input):

- PromotedModel pushModel
    - Object with 2 member variables of fixed size
- pair<PromotedModel, PromotedModel> pushPair
    - 2 objects with members of fixed size

For the execution of the push() function, a worst-case of 6 memory units are utilized in auxiliary space. For any arbitrary constant **k**,

k * O(1) = O(1)

Therefore, we have an auxiliary space complexity expressed as

S(N) = O(1)

Pop

Time Complexity Analysis

```
PromotedModel PromotedCarModelStack::pop() {
  if (promoStack.empty()){                    +1
    throw logic_error("Promoted car model stack is empty");
  }                          Worst-case occurs upon comparison being untrue
  PromotedModel poppedModel = promoStack.back();        +1
  promoStack.pop_back();                      +1
  trackStack.pop_back();                      +1
  return poppedModel;                         +1
}
```

Due to all operations remaining constant, counting is unneeded since, for any arbitrary constant **k**, **k * O(1) = O(1)**

Therefore, the Time Complexity f(N) *yielded can be expressed as*

f(N) = O(1)

Auxiliary Space Complexity Analysis

Memory (Non-Input):

- PromotedModel poppedModel
  - Object with 2 member variables of fixed size

For the execution of the pop() function, a worst-case of 1 memory unit are utilized in auxiliary space. For any arbitrary constant **k**,

k * O(1) = O(1)

Therefore, we have an auxiliary space complexity expressed as

S(N) = O(1)

Peek

Time Complexity Analysis

PromotedModel PromotedCarModelStack::peek() {

if (promoStack.empty()){ **+1**

throw logic_error("Promoted car model stack is empty");

} *For the* entire *program, worst-case looks towards an untrue comparison, however, in context to only this member function, either suffices*

return promoStack.back();

} **+1**

Due to all operations remaining constant, counting is unneeded since, for any arbitrary constant **k**, **k * O(1) = O(1)**

Therefore, the Time Complexity f(N) *yielded can be expressed as*

f(N) = O(1)

Auxiliary Space Complexity Analysis

Memory (Non-Input):

- N/A

For the execution of the peek() function, no new memory units are utilized as it is directly referencing an object from the input-data in its return. Should an exception be thrown, any consequential space utilized is self-contained and irrelevant to any variables in peek() itself.

Therefore, we have an auxiliary space complexity expressed as

S(N) = O(1)

Get Highest Priced Promoted Model

Time Complexity Analysis

```
PromotedModel PromotedCarModelStack::getHighestPricedPromotedModel() {
  if (promoStack.empty()){                    +1
    throw logic_error("Promoted car model stack is empty");
  }                                           +1
  return trackStack.back().second;
}
```

Due to all operations remaining constant, counting is unneeded since, for any arbitrary constant **k**, **k * O(1) = O(1)**

Therefore, the Time Complexity f(N) *yielded can be expressed as*

f(N) = O(1)

Auxiliary Space Complexity Analysis

Memory (Non-Input):

- N/A

For the execution of the getHighestPricedPromotedModel() function, no new memory units are utilized as it is directly referencing an object from the input-data in its return.

Therefore, we have an auxiliary space complexity expressed as

S(N) = O(1)

Get Lowest Priced Promoted Model

Time Complexity Analysis

```
PromotedModel PromotedCarModelStack::getLowestPricedPromotedModel() {
  if (promoStack.empty()){
    throw logic_error("Promoted car model stack is empty");
  }
  return trackStack.back().first;
}
```

Due to all operations remaining constant, counting is unneeded since, for any arbitrary constant **k**, **k * O(1) = O(1)**

Therefore, the Time Complexity f(N) *yielded can be expressed as*

f(N) = O(1)

Auxiliary Space Complexity Analysis

Memory (Non-Input):

- N/A

For the execution of the getLowestPricedPromotedModel() function, no new memory units are utilized as it is directly referencing an object from the input-data in its return.

Therefore, we have an auxiliary space complexity expressed as

S(N) = O(1)