Decomposition
Announcements
Modular Design
Separation of Concerns
Separation of Concerns

A design principle: Isolate different parts of a program that address different concerns
Separation of Concerns

A design principle: Isolate different parts of a program that address different concerns
A modular component can be developed and tested independently
Separation of Concerns

A design principle: Isolate different parts of a program that address different concerns. A modular component can be developed and tested independently.

Hog

- Hog Game Simulator
- Game Commentary
- Player Strategies
Separation of Concerns

A design principle: Isolate different parts of a program that address different concerns

A modular component can be developed and tested independently

Hog

- Hog Game Simulator

- Game rules

- Game Commentary

- Player Strategies
Separation of Concerns

A design principle: Isolate different parts of a program that address different concerns. A modular component can be developed and tested independently.

Hog

- Hog Game Simulator
- Game rules
- Ordering of events

- Game Commentary

- Player Strategies
Separation of Concerns

A design principle: Isolate different parts of a program that address different concerns. A modular component can be developed and tested independently.

Hog

- Hog Game Simulator
- Game rules
- Ordering of events
- State tracking to determine the winner

Game Commentary

Player Strategies

Hog Game Simulator

Game Commentary

Player Strategies
Separation of Concerns

A design principle: Isolate different parts of a program that address different concerns
A modular component can be developed and tested independently

Hog

- Hog Game Simulator
  - Game rules
  - Ordering of events
  - State tracking to determine the winner

- Game Commentary
  - Event descriptions

- Player Strategies
Separation of Concerns

A design principle: Isolate different parts of a program that address different concerns. A modular component can be developed and tested independently.

Hog

- Hog Game Simulator
  - Game rules
  - Ordering of events
  - State tracking to determine the winner

- Game Commentary
  - Event descriptions
  - User input

- Player Strategies
Separation of Concerns

A design principle: Isolate different parts of a program that address different concerns. A modular component can be developed and tested independently.

**Hog**

- Hog Game Simulator
  - Game rules
  - Ordering of events
  - State tracking to determine the winner

- Game Commentary
  - Event descriptions
  - User input

- Player Strategies
  - Decision rules

Hog Game Simulator

Game Commentary

Player Strategies
Separation of Concerns

A design principle: Isolate different parts of a program that address different concerns. A modular component can be developed and tested independently.

Hog

- Hog Game Simulator
  - Game rules
  - Ordering of events
  - State tracking to determine the winner

- Game Commentary
  - Event descriptions
  - User input

- Player Strategies
  - Decision rules
  - Strategy parameters (e.g., margins & number of dice)
Separation of Concerns

A design principle: Isolate different parts of a program that address different concerns.
A modular component can be developed and tested independently.

**Hog**

- Hog Game Simulator
  - Game rules
  - Ordering of events
  - State tracking to determine the winner

- Game Commentary
  - Event descriptions
  - User input

- Player Strategies
  - Decision rules
  - Strategy parameters (e.g., margins & number of dice)
Separation of Concerns

A design principle: Isolate different parts of a program that address different concerns.
A modular component can be developed and tested independently.

### Hog

- **Hog Game Simulator**
  - Game rules
  - Ordering of events
  - State tracking to determine the winner

- **Game Commentary**
  - Event descriptions
  - User input

- **Player Strategies**
  - Decision rules
  - Strategy parameters (e.g., margins & number of dice)

### Ants

- **Ants Game Simulator**

- **Actions**

- **Tunnel Structure**
**Separation of Concerns**

A design principle: Isolate different parts of a program that address different concerns

A modular component can be developed and tested independently

---

**Hog**

- **Hog Game Simulator**
  - Game rules
  - Ordering of events
  - State tracking to determine the winner

- **Game Commentary**
  - Event descriptions
  - User input

- **Player Strategies**
  - Decision rules
  - Strategy parameters (e.g., margins & number of dice)

---

**Ants**

- **Ants Game Simulator**
  - Order of actions

- **Actions**

- **Tunnel Structure**
Separation of Concerns

A design principle: Isolate different parts of a program that address different concerns
A modular component can be developed and tested independently

Hog
- Hog Game Simulator
  - Game rules
  - Ordering of events
  - State tracking to determine the winner
- Game Commentary
  - Event descriptions
  - User input
- Player Strategies
  - Decision rules
  - Strategy parameters (e.g., margins & number of dice)

Ants
- Ants Game Simulator
  - Order of actions
  - Food tracking
- Actions
- Tunnel Structure
Separation of Concerns

A design principle: Isolate different parts of a program that address different concerns
A modular component can be developed and tested independently

Hog
- Hog Game Simulator
  - Game rules
  - Ordering of events
  - State tracking to determine the winner
- Game Commentary
  - Event descriptions
  - User input
- Player Strategies
  - Decision rules
  - Strategy parameters (e.g., margins & number of dice)

Ants
- Ants Game Simulator
  - Order of actions
  - Food tracking
  - Game ending conditions
- Actions
- Tunnel Structure
Separation of Concerns

A design principle: Isolate different parts of a program that address different concerns.
A modular component can be developed and tested independently.

### Hog

- **Hog Game Simulator**
  - Game rules
  - Ordering of events
  - State tracking to determine the winner

- **Game Commentary**
  - Event descriptions
  - User input

- **Player Strategies**
  - Decision rules
  - Strategy parameters (e.g., margins & number of dice)

### Ants

- **Ants Game Simulator**
  - Order of actions
  - Food tracking
  - Game ending conditions

- **Actions**
  - Characteristics of different ants & bees

- **Tunnel Structure**

Hog Game Simulator

Game Commentary

Player Strategies

Ants Game Simulator

Actions

Tunnel Structure
Separation of Concerns

A design principle: Isolate different parts of a program that address different concerns.
A modular component can be developed and tested independently.

### Hog

- **Hog Game Simulator**
  - Game rules
  - Ordering of events
  - State tracking to determine the winner

- **Game Commentary**
  - Event descriptions
  - User input

- **Player Strategies**
  - Decision rules
  - Strategy parameters (e.g., margins & number of dice)

### Ants

- **Ants Game Simulator**
  - Order of actions
  - Food tracking
  - Game ending conditions

- **Actions**
  - Characteristics of different ants & bees

- **Tunnel Structure**
  - Entrances & exits
### Separation of Concerns

A design principle: Isolate different parts of a program that address different concerns. A modular component can be developed and tested independently.

<table>
<thead>
<tr>
<th>Hog Game Simulator</th>
<th>Game Commentary</th>
<th>Player Strategies</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Game rules</td>
<td>• Event descriptions</td>
<td>• Decision rules</td>
</tr>
<tr>
<td>• Ordering of events</td>
<td>• User input</td>
<td>• Strategy parameters</td>
</tr>
<tr>
<td>• State tracking to determine the winner</td>
<td></td>
<td>(e.g., margins &amp; number of dice)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Ants Game Simulator</th>
<th>Actions</th>
<th>Tunnel Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Order of actions</td>
<td>• Characteristics of different ants &amp; bees</td>
<td>• Entrances &amp; exits</td>
</tr>
<tr>
<td>• Food tracking</td>
<td></td>
<td>• Locations of insects</td>
</tr>
<tr>
<td>• Game ending conditions</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example: Restaurant Search
Restaurant Search Data

Given the following data, look up a restaurant by name and show related restaurants.
Restaurant Search Data

Given the following data, look up a restaurant by name and show related restaurants.

{"business_id": "gclB3ED6uk6viWlolSb_uA", "name": "Cafe 3", "stars": 2.0, "price": 1, ...}
Restaurant Search Data

Given the following data, look up a restaurant by name and show related restaurants.

{"business_id": "gclB3ED6uk6viWlolSb_uA", "name": "Cafe 3", "stars": 2.0, "price": 1, ...}
{"business_id": "WXKx2I2SEzBpeUGtDMCS8A", "name": "La Cascada Taqueria", "stars": 3.0, "price": 2}
Restaurant Search Data

Given the following data, look up a restaurant by name and show related restaurants.

{"business_id": "gclB3ED6uk6viWlolSb_uA", "name": "Cafe 3", "stars": 2.0, "price": 1, ...}
{"business_id": "WXKx2I2SeZBpeUQGtDMCS8A", "name": "La Cascada Taqueria", "stars": 3.0, "price": 2}
...
Restaurant Search Data

Given the following data, look up a restaurant by name and show related restaurants.

{"business_id": "gclB3ED6uk6viWloI5b_uA", "name": "Cafe 3", "stars": 2.0, "price": 1, ...}
{"business_id": "WXKx2I2SEzBpeUg6tDMCS8A", "name": "La Cascada Taqueria", "stars": 3.0, "price": 2}
...
{"business_id": "gclB3ED6uk6viWloI5b_uA", "user_id": "xVocUszkZtAqCxmlgWak3xVQ", "stars": 1, "text": "Cafe 3 (or Cafe Tre, as I like to say) used to be the bomb diggity when I first lived in the dorms but sadly, quality has dramatically decreased over the years....", "date": "2012-01-19", ...}
Restaurant Search Data

Given the following data, look up a restaurant by name and show related restaurants.

{"business_id": "gclB3ED6uk6viWloI5b_uA", "name": "Cafe 3", "stars": 2.0, "price": 1, ...}  
{"business_id": "WXKx2I2SEzBpeUGtDMCS8A", "name": "La Cascada Taqueria", "stars": 3.0, "price": 2}  
...  
{"business_id": "gclB3ED6uk6viWloI5b_uA", "user_id": "xVocUszkZtAqCulgWak3xVQ", "stars": 1, "text": "Cafe 3 (or Cafe Tre, as I like to say) used to be the bomb diggity when I first lived in the dorms but sadly, quality has dramatically decreased over the years....", "date": "2012-01-19", ...}  
{"business_id": "WXKx2I2SEzBpeUGtDMCS8A", "user_id": "84dCHkhWg8IDtk30VvaY5A", "stars": 2, "text": "Excuse me for being a snob but if I wanted a room temperature burrito I would take one home, stick it in the fridge for a day, throw it in the microwave for 45 seconds, then eat it. NOT go to a restaurant and pay like seven dollars for one...", "date": "2009-04-30", ...}
Restaurant Search Data

Given the following data, look up a restaurant by name and show related restaurants.

{"business_id": "gclB3ED6uk6viWlolSb_uA", "name": "Cafe 3", "stars": 2.0, "price": 1, ...}
{"business_id": "WXKx2I2SEzBpeUGtDMCS8A", "name": "La Cascada Taqueria", "stars": 3.0, "price": 2, ...}

{"business_id": "gclB3ED6uk6viWlolSb_uA", "user_id": "xVocUszkZtAqCwgWak3xVQ", "stars": 1, "text": "Cafe 3 (or Cafe Tre, as I like to say) used to be the bomb diggity when I first lived in the dorms but sadly, quality has dramatically decreased over the years....", "date": "2012-01-19", ...}

{"business_id": "WXKx2I2SEzBpeUGtDMCS8A", "user_id": "84dCHkhW8IDtk30VvaY5A", "stars": 2, "text": "-Excuse me for being a snob but if I wanted a room temperature burrito I would take one home, stick it in the fridge for a day, throw it in the microwave for 45 seconds, then eat it. NOT go to a restaurant and pay like seven dollars for one...", "date": "2009-04-30", ...}
Restaurant Search Data

Given the following data, look up a restaurant by name and show related restaurants.

```json
{"business_id": "gclB3ED6uk6viWloI5b_uA", "name": "Cafe 3", "stars": 2.0, "price": 1, ...}
{"business_id": "WXKx2I2SEzBpeUGtDMCS8A", "name": "La Cascada Taqueria", "stars": 3.0, "price": 2}
...
{"business_id": "gclB3ED6uk6viWloI5b_uA", "user_id": "xVocUszkZtAqCxtgWak3xVQ", "stars": 1, "text": "Cafe 3 (or Cafe Tre, as I like to say) used to be the bomb diggity when I first lived in the dorms but sadly, quality has dramatically decreased over the years.....", "date": "2012-01-19", ...}
{"business_id": "WXKx2I2SEzBpeUGtDMCS8A", "user_id": "84dCHkhWGBkt30VvvaY5A", "stars": 2, "text": "Excuse me for being a snob but if I wanted a room temperature burrito I would take one home, stick it in the fridge for a day, throw it in the microwave for 45 seconds, then eat it. NOT go to a restaurant and pay like seven dollars for one...", "date": "2009-04-30", ...}
...
```

(Demo)
Example: Similar Restaurants
Discussion Question: Most Similar Restaurants
Discussion Question: Most Similar Restaurants

Implement `similar`, a `Restaurant` method that takes a positive integer `k` and a function `similarity` that takes two restaurants as arguments and returns a number. Higher `similarity` values indicate more similar restaurants. The `similar` method returns a list containing the `k` most similar restaurants according to the `similarity` function, but not containing `self`. 
Discussion Question: Most Similar Restaurants

Implement `similar`, a `Restaurant` method that takes a positive integer `k` and a function `similarity` that takes two restaurants as arguments and returns a number. Higher `similarity` values indicate more similar restaurants. The `similar` method returns a list containing the `k` most similar restaurants according to the `similarity` function, but not containing `self`.

```python
def similar(self, k, similarity):
```
Discussion Question: Most Similar Restaurants

Implement `similar`, a `Restaurant` method that takes a positive integer `k` and a function `similarity` that takes two restaurants as arguments and returns a number. Higher `similarity` values indicate more similar restaurants. The `similar` method returns a list containing the `k` most similar restaurants according to the `similarity` function, but not containing `self`.

```python
def similar(self, k, similarity):
    """Return the K most similar restaurants to SELF, using SIMILARITY for comparison."""
```
Discussion Question: Most Similar Restaurants

Implement `similar`, a `Restaurant` method that takes a positive integer `k` and a function `similarity` that takes two restaurants as arguments and returns a number. Higher `similarity` values indicate more similar restaurants. The `similar` method returns a list containing the `k` most similar restaurants according to the `similarity` function, but not containing `self`.

```python
def similar(self, k, similarity):
    "Return the K most similar restaurants to SELF, using SIMILARITY for comparison."

    others = list(Restaurant.all)
```

8
Implement `similar`, a `Restaurant` method that takes a positive integer `k` and a function `similarity` that takes two restaurants as arguments and returns a number. Higher `similarity` values indicate more similar restaurants. The `similar` method returns a list containing the `k` most similar restaurants according to the `similarity` function, but not containing `self`.

```python
def similar(self, k, similarity):
    "Return the K most similar restaurants to SELF, using SIMILARITY for comparison."

    others = list(Restaurant.all)

    others.__________(__________)  # Complete the line here
```
Discussion Question: Most Similar Restaurants

Implement `similar`, a `Restaurant` method that takes a positive integer `k` and a function `similarity` that takes two restaurants as arguments and returns a number. Higher `similarity` values indicate more similar restaurants. The `similar` method returns a list containing the `k` most similar restaurants according to the `similarity` function, but not containing `self`.

```python
def similar(self, k, similarity):
    "Return the K most similar restaurants to SELF, using SIMILARITY for comparison."

    others = list(Restaurant.all)

    others.______________(______________)

    return sorted(others, key=______________________________)__________________
```
Discussion Question: Most Similar Restaurants

Implement `similar`, a `Restaurant` method that takes a positive integer `k` and a function `similarity` that takes two restaurants as arguments and returns a number. Higher `similarity` values indicate more similar restaurants. The `similar` method returns a list containing the `k` most similar restaurants according to the `similarity` function, but not containing `self`.

```python
def similar(self, k, similarity):
    "Return the K most similar restaurants to SELF, using SIMILARITY for comparison."

    others = list(Restaurant.all)

    others.______________(______________)

    return sorted(others, key=______________________________)__________________
```

`sorted(iterable, /, *, key=None, reverse=False)`
Return a new list containing all items from the iterable in ascending order. A custom key function can be supplied to customize the sort order, and the reverse flag can be set to request the result in descending order.
Discussion Question: Most Similar Restaurants

Implement `similar`, a `Restaurant` method that takes a positive integer `k` and a function `similarity` that takes two restaurants as arguments and returns a number. Higher `similarity` values indicate more similar restaurants. The `similar` method returns a list containing the `k` most similar restaurants according to the `similarity` function, but not containing `self`.

```python
def similar(self, k, similarity):
    "Return the K most similar restaurants to SELF, using SIMILARITY for comparison."

    others = list(Restaurant.all)

    others.______________(______________)

    return sorted(others, key=________________________)__________________
```

**sorted**

*Return a new list containing all items from the iterable in ascending order. A custom key function can be supplied to customize the sort order, and the reverse flag can be set to request the result in descending order.*
Implement `similar`, a `Restaurant` method that takes a positive integer `k` and a function `similarity` that takes two restaurants as arguments and returns a number. Higher `similarity` values indicate more similar restaurants. The `similar` method returns a list containing the `k` most similar restaurants according to the `similarity` function, but not containing `self`.

```python
def similar(self, k, similarity):
    "Return the K most similar restaurants to SELF, using SIMILARITY for comparison."

    others = list(Restaurant.all)
    others.__________(__________)  # remove(self)

    return sorted(others, key=_____________________________)____________________
```

`sorted(iterable, /, *, key=None, reverse=False)`
Return a new list containing all items from the iterable in ascending order. A custom key function can be supplied to customize the sort order, and the reverse flag can be set to request the result in descending order.
Discussion Question: Most Similar Restaurants

Implement `similar`, a `Restaurant` method that takes a positive integer `k` and a function `similarity` that takes two restaurants as arguments and returns a number. Higher `similarity` values indicate more similar restaurants. The `similar` method returns a list containing the `k` most similar restaurants according to the `similarity` function, but not containing `self`.

```python
def similar(self, k, similarity):
    """Return the K most similar restaurants to SELF, using SIMILARITY for comparison.""

    others = list(Restaurant.all)
    others.____________(___________)

    return sorted(others, key=________lambda r: -________(__________)_________)___________
```

`sorted(iterable, /, *, key=None, reverse=False)`
Return a new list containing all items from the iterable in ascending order. A custom key function can be supplied to customize the sort order, and the reverse flag can be set to request the result in descending order.
Discussion Question: Most Similar Restaurants

Implement `similar`, a `Restaurant` method that takes a positive integer `k` and a function `similarity` that takes two restaurants as arguments and returns a number. Higher `similarity` values indicate more similar restaurants. The `similar` method returns a list containing the `k` most similar restaurants according to the `similarity` function, but not containing `self`.

```python
def similar(self, k, similarity):
    "Return the K most similar restaurants to SELF, using SIMILARITY for comparison."

    others = list(Restaurant.all)

    others.___________(___________)

    return sorted(others, key=lambda r: -similarity(self, r))[:k]
```

`sorted(iterable, /, *, key=None, reverse=False)`
Return a new list containing all items from the iterable in ascending order. A custom key function can be supplied to customize the sort order, and the reverse flag can be set to request the result in descending order.
Example: Reading Files

(Demo)
Set Intersection
Linear-Time Intersection of Sorted Lists

Given two sorted lists with no repeats, return the number of elements that appear in both.
Linear-Time Intersection of Sorted Lists

Given two sorted lists with no repeats, return the number of elements that appear in both.

```plaintext
3  4  6  7  9  10
```
Linear-Time Intersection of Sorted Lists

Given two sorted lists with no repeats, return the number of elements that appear in both.

```
3  4  6  7  9  10
```

```
1  3  5  7  8
```
Linear-Time Intersection of Sorted Lists

Given two sorted lists with no repeats, return the number of elements that appear in both.

3 4 6 7 9 10

1 3 5 7 8
Linear-Time Intersection of Sorted Lists

Given two sorted lists with no repeats, return the number of elements that appear in both.

```
3  4  6  7  9  10
```
```
1  3  5  7  8
```
Linear-Time Intersection of Sorted Lists

Given two sorted lists with no repeats, return the number of elements that appear in both.

```
3  4  6  7  9  10
    ▼

1  3  5  7  8
    ▼
```
Linear-Time Intersection of Sorted Lists

Given two sorted lists with no repeats, return the number of elements that appear in both.

```
3 4 6 7 9 10
1 3 5 7 8
```
Linear-Time Intersection of Sorted Lists

Given two sorted lists with no repeats, return the number of elements that appear in both.

3 4 6 7 9 10

1 3 5 7 8
Linear-Time Intersection of Sorted Lists

Given two sorted lists with no repeats, return the number of elements that appear in both.

\[
\begin{array}{cccccc}
3 & 4 & 6 & 7 & 9 & 10 \\
\hline
1 & 3 & 5 & 7 & 8 \\
\end{array}
\]
Linear-Time Intersection of Sorted Lists

Given two sorted lists with no repeats, return the number of elements that appear in both.

\[
\begin{array}{cccccc}
3 & 4 & 6 & 7 & 9 & 10 \\
\end{array}
\]

\[
\begin{array}{cccccc}
1 & 3 & 5 & 7 & 8 \\
\end{array}
\]
## Linear-Time Intersection of Sorted Lists

Given two sorted lists with no repeats, return the number of elements that appear in both.

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>4</td>
<td>6</td>
<td>7</td>
<td>9</td>
<td>10</td>
</tr>
</tbody>
</table>

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>5</td>
<td>7</td>
<td>8</td>
</tr>
</tbody>
</table>
Given two sorted lists with no repeats, return the number of elements that appear in both.

```python
def fast_overlap(s, t):
    """Return the overlap between sorted S and sorted T."

    >>> fast_overlap([3, 4, 6, 7, 9, 10], [1, 3, 5, 7, 8])
    2
    
    i, j, count = 0, 0, 0
    while ____________________________________________:
        if s[i] == t[j]:
            count, i, j = ____________________________
        elif s[i] < t[j]:
            ____________________________
        else:
            ____________________________
    return count
```
Linear-Time Intersection of Sorted Lists

Given two sorted lists with no repeats, return the number of elements that appear in both.

```python
def fast_overlap(s, t):
    """Return the overlap between sorted S and sorted T."
    >>> fast_overlap([3, 4, 6, 7, 9, 10], [1, 3, 5, 7, 8])
    2
    """
    i, j, count = 0, 0, 0
    while i < len(s) and j < len(t):
        if s[i] == t[j]:
            count, i, j = _______________________
        elif s[i] < t[j]:
            ______________________________________
        else:____________________________________
    return count
```
Linear-Time Intersection of Sorted Lists

Given two sorted lists with no repeats, return the number of elements that appear in both.

```python
def fast_overlap(s, t):
    """Return the overlap between sorted S and sorted T."
    >>> fast_overlap([3, 4, 6, 7, 9, 10], [1, 3, 5, 7, 8])
    2
    i, j, count = 0, 0, 0
    while i < len(s) and j < len(t):
        if s[i] == t[j]:
            count, i, j = count + 1, i + 1, j + 1
        elif s[i] < t[j]:
            ____________________________
        else:
            ____________________________
    return count
```
Given two sorted lists with no repeats, return the number of elements that appear in both.

```python
def fast_overlap(s, t):
    """Return the overlap between sorted S and sorted T."

    >>> fast_overlap([3, 4, 6, 7, 9, 10], [1, 3, 5, 7, 8])
    2
    >>>
    i, j, count = 0, 0, 0
    while i < len(s) and j < len(t):
        if s[i] == t[j]:
            count, i, j = count + 1, i + 1, j + 1
        elif s[i] < t[j]:
            i = i + 1
        else:
            j = j + 1
    return count
```

Linear-Time Intersection of Sorted Lists
Linear-Time Intersection of Sorted Lists

Given two sorted lists with no repeats, return the number of elements that appear in both.

```python
def fast_overlap(s, t):
    """Return the overlap between sorted S and sorted T."

    >>> fast_overlap([3, 4, 6, 7, 9, 10], [1, 3, 5, 7, 8])
    2
    """
    i, j, count = 0, 0, 0
    while i < len(s) and j < len(t):
        if s[i] == t[j]:
            count, i, j = count + 1, i + 1, j + 1
        elif s[i] < t[j]:
            i = i + 1
        else:
            j = j + 1
    return count
```
Given two sorted lists with no repeats, return the number of elements that appear in both.

```
def fast_overlap(s, t):
    """Return the overlap between sorted S and sorted T."

    >>> fast_overlap([3, 4, 6, 7, 9, 10], [1, 3, 5, 7, 8])
    2
    ""
    i, j, count = 0, 0, 0
    while i < len(s) and j < len(t):
        if s[i] == t[j]:
            count, i, j = count + 1, i + 1, j + 1
        elif s[i] < t[j]:
            i = i + 1
        else:
            j = j + 1
    return count
```