While you wait

- Read the Alcohol APT and think about how to solve it.
- Open Eclipse

Problem Statement

The Stanford Office of Student Affairs has been placing student groups (including the one, the only, the truly incomparable, Leland Stanford Junior University Marching Band) on alcohol probation. However, due to some legal nitpicking, they are only allowed to place a group on alcohol probation if it throws 3 parties in a given year.

Given a String[] schedule, which represents which group throws a party on each day, return a string of the first group that becomes legally eligible for alcohol probation. Each element of schedule will represent one day. If the same group throws 3 or more parties, it becomes legally eligible for alcohol probation.

If no group becomes legally eligible for alcohol probation, return "".

From last time

- Array – ordered, indexed, fixed length
- List – ordered, indexed, adjustable length
- Set – unordered, adjustable length, no doubles
- Map – unordered, pairs (key, value)
Maps

- Alcohol APT

Problem Statement

The Stanford Office of Student Affairs has been placing student groups (including the one, the only, the truly incomparable, Leland Stanford Junior University Marching Band) on alcohol probation. However, due to some legal nitpicking, they are only allowed to place a group on alcohol probation if it throws 3 parties in a given year.

Given a string() schedule, which represents which group throws a party on each day, return a string of the first group that becomes legally eligible for alcohol probation. Each element of schedule will represent one day. If the same group throws 3 or more parties, it becomes legally eligible for alcohol probation.

If no group becomes legally eligible for alcohol probation, return "".
Jotto

- Do you want help getting started?
  - Assignment walkthrough
    - When: Wednesday, January 29 7:00-8:30pm
    - Where: LSRC B101
    - Who: Owen Astrachan

Array vs. ArrayList
Big-Oh

- Big-Oh
  - Estimate time required for a program
  - No units of time!!!!!!!
  - Count operations

Assign costs to operations
- Declarations cost 0 units
  - double aDouble;
- Operations cost 1 unit
  - aDouble = 4.56 //assignment
  - aDouble * 5 //mathematical operation
  - return aDouble; //returns
public double getArea(double r) {
    double pi;
    pi = 3.14;
    double area;
    area = pi * r * r;
    return area;
}
public double getArea(double r) {
    double pi;                  0
    pi = 3.14;                  1
    double area;               0
    area = pi * r * r;         3
    return area;              1
}

Total: 5

O(5)

Assign costs to operations
- Declarations cost 0 units
- Operations cost 1 unit

Write in Big-Oh notation

Simplify
- Remove constants
  - O(6) = O(1)
  - O(4N) = O(N)
  - O(3N^2 + 5) = O(N^2)
- Remove lower order terms
  - O(N^2 + N) = O(N^2)
Big-Oh

```
1 public double getArea(double r){
2    double pi;          0
3    pi = 3.14;        1
4    double area;      0
5    area = pi * r * r; 3
6    return area;      1
7 }
```

Total: 5

\[ O(1) \]

\[ O(5) \]

---

**Assign costs to operations**
- Declarations cost 0 units
- Operations cost 1 unit

**Write in Big-Oh notation**

**Simplify**
- Remove constants
  - \( O(6) = O(1) \)
  - \( O(4N) = O(N) \)
  - \( O(3N^2 + 5) = O(N^2) \)
- Remove lower order terms
  - \( O(N^2 + N) = O(N^2) \)
Your turn

```java
public static int sum(int n) {
    int partialSum;
    partialSum = 0;
    for (int i = 1; i <= n; i++)
        partialSum += i * i * i;
    return partialSum;
}
```

O(1 + N*4 + 1) = O(4N + 2)
Big-Oh

• Rules
  • Consecutive statements
    • Add them
  • for-loops
    • (statements in for-loop) * iterations
  • Nested for-loops (inside-out)
    • (statements in innermost for-loop) * iterations * iterations
  • If/else
    • Test + max(if, else)

Big-Oh

• Rules
  • Consecutive statements
    • Add them
  • for-loops
    • (statements in for-loop) * iterations
  • Nested for-loops (inside-out)
    • (statements in innermost for-loop) * iterations * iterations
  • If/else
    • Test + max(if, else)
Consecutive statements

int foo = 0;
foo = foo + 7;
int bar = 2;
bar = bar * bar;

Big-Oh

• Rules
  • Consecutive statements
    • Add them
  • for-loops
    • (statements in for-loop) * iterations
  • Nested for-loops (inside-out)
    • (statements in innermost for-loop) * iterations * iterations
  • If/else
    • Test + max(if, else)
for-loops

for(int i = 0; i < foo; i++){
    bar = bar * 2;
}

Big-Oh

• Rules
  • Consecutive statements
    • Add them
  • for-loops
    • (statements in for-loop) * iterations
  • Nested for-loops (inside-out)
    • (statements in innermost for-loop) * iterations * iterations
  • If/else
    • Test + max(if, else)
for (int i = 0; i < foo; i++){
    for(int j = 0; j < bar; j++){
        System.out.println("hello");
    }
}

**Big-Oh**

- **Rules**
  - Consecutive statements
    - Add them
  - for-loops
    - (statements in for-loop) * iterations
  - Nested for-loops (inside-out)
    - (statements in innermost for-loop) * iterations * iterations
  - If/else
    - Test + max(if, else)
if(foo < 100){
    System.out.println("foo < 100");
}
else{
    for(int i = 0; i < foo; i++){
        System.out.println("foo > 100");
    }
}
}
<table>
<thead>
<tr>
<th>Function</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c$</td>
<td>Constant</td>
</tr>
<tr>
<td>$\log N$</td>
<td>Logarithmic</td>
</tr>
<tr>
<td>$\log^2 N$</td>
<td>Log-squared</td>
</tr>
<tr>
<td>$N$</td>
<td>Linear</td>
</tr>
<tr>
<td>$N \log N$</td>
<td></td>
</tr>
<tr>
<td>$N^2$</td>
<td>Quadratic</td>
</tr>
<tr>
<td>$N^3$</td>
<td>Cubic</td>
</tr>
<tr>
<td>$2^N$</td>
<td>Exponential</td>
</tr>
</tbody>
</table>
• Do you want help getting started?
  • Assignment walkthrough
    • When: Wednesday, January 29 7:00-8:30pm
    • Where: LSRC B101
    • Who: Owen Astrachan