

## HOUSE ROBBER 2

Similar to House Robber 1 only difference is that the houses are all connected and our circular which indicates the first index & the last index are also adjacent elements.

Sol

We can solve the answer with the same logic as house robber 1 with just a small update which is

a) We'll leave out the last element & apply the same house robber 1 to all the remaining elements (or)

b) We can leave out the first element & apply the same house robber 1 logic to all the remaining elements

Let's say a) is ans 1 & (b) is ans 2  
(leaving last) (leaving first)

∴ can just return the answer to this question is  
 $\max(\text{ans1}, \text{ans2})$

Sol ~~int~~



code

```
HoweRobber1(List<Integer> nums) {
```

```
    int n = nums.size();
```

```
    if (n == 0) return 0;
```

```
    if (n == 1) return nums.get(0);
```

```
    int prev1 = nums.get(0);
```

```
    int prev2 = 0;
```

```
    for (int i = 1; i < n; i++) {
```

```
        int take = nums.get(i);
```

```
        if (i > 1) take += prev2;
```

```
        int notTake = 0 + prev1;
```

```
        int curr = Math.max(take, notTake);
```

```
        prev2 = prev1;
```

```
        prev1 = curr;
```

```
    }
```

```
    return prev1;
```

```
}
```

```
HoweRobber2(int[] Howes) {
```

```
    int n = Howes.length;
```

```
    if (n == 1) return Howes[0];
```

```
    List<Integer> temp1 = new ArrayList<>(1);
```

```
    List<Integer> temp2 = new ArrayList<>(1);
```

```
    for (int i = 0; i < n; i++) {
```

```
        if (i != 0) temp1.add(Howes[i]);
```

```
        if (i != n - 1) temp2.add(Howes[i]);
```

```
    }
```

```
    return Math.max(
```

```
        max HoweRobber1(temp1),
```

```
        HoweRobber1(temp2)); }
```