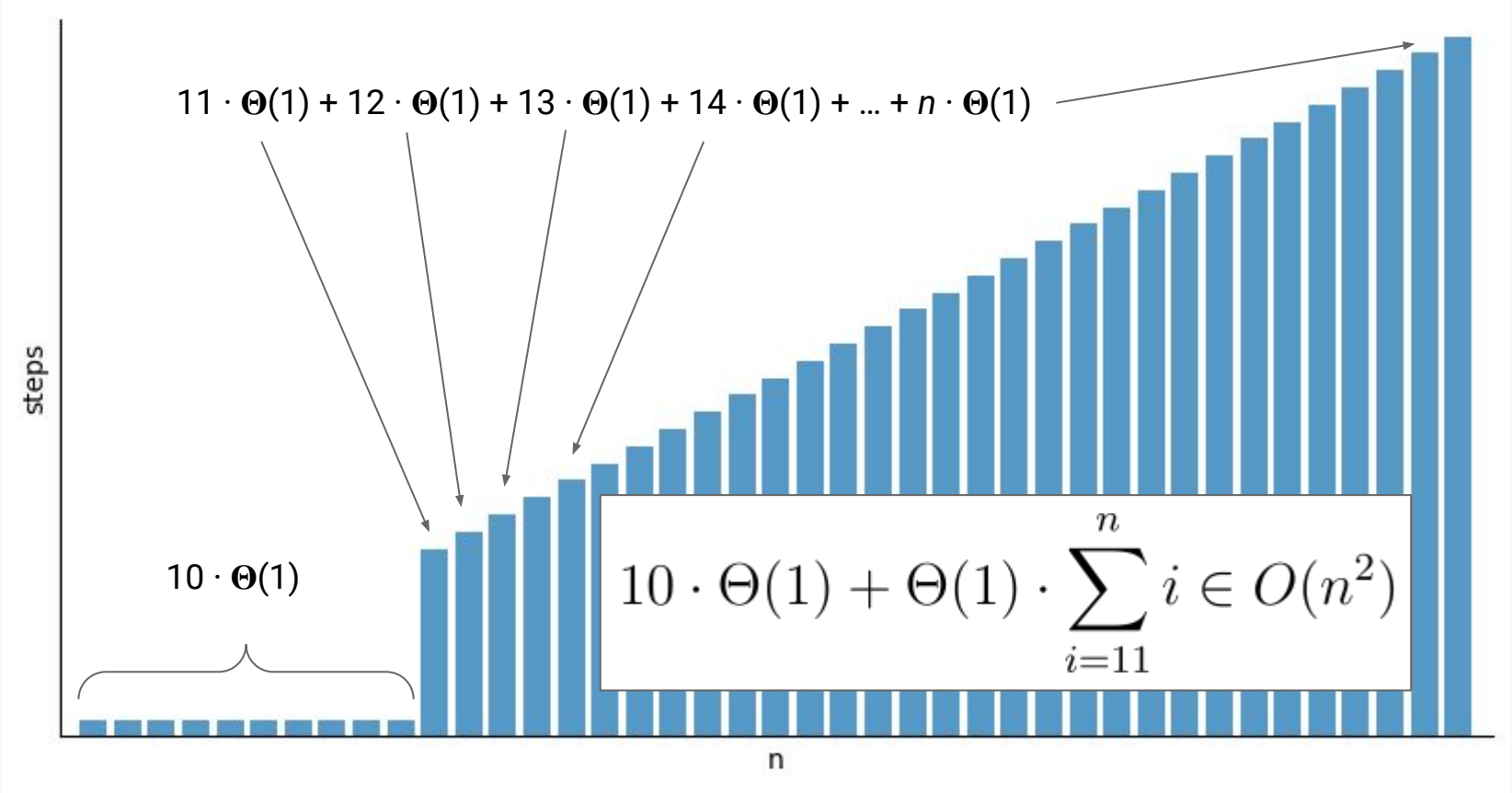
CSE-250 Recitation

September 23~24: Amortized Runtime

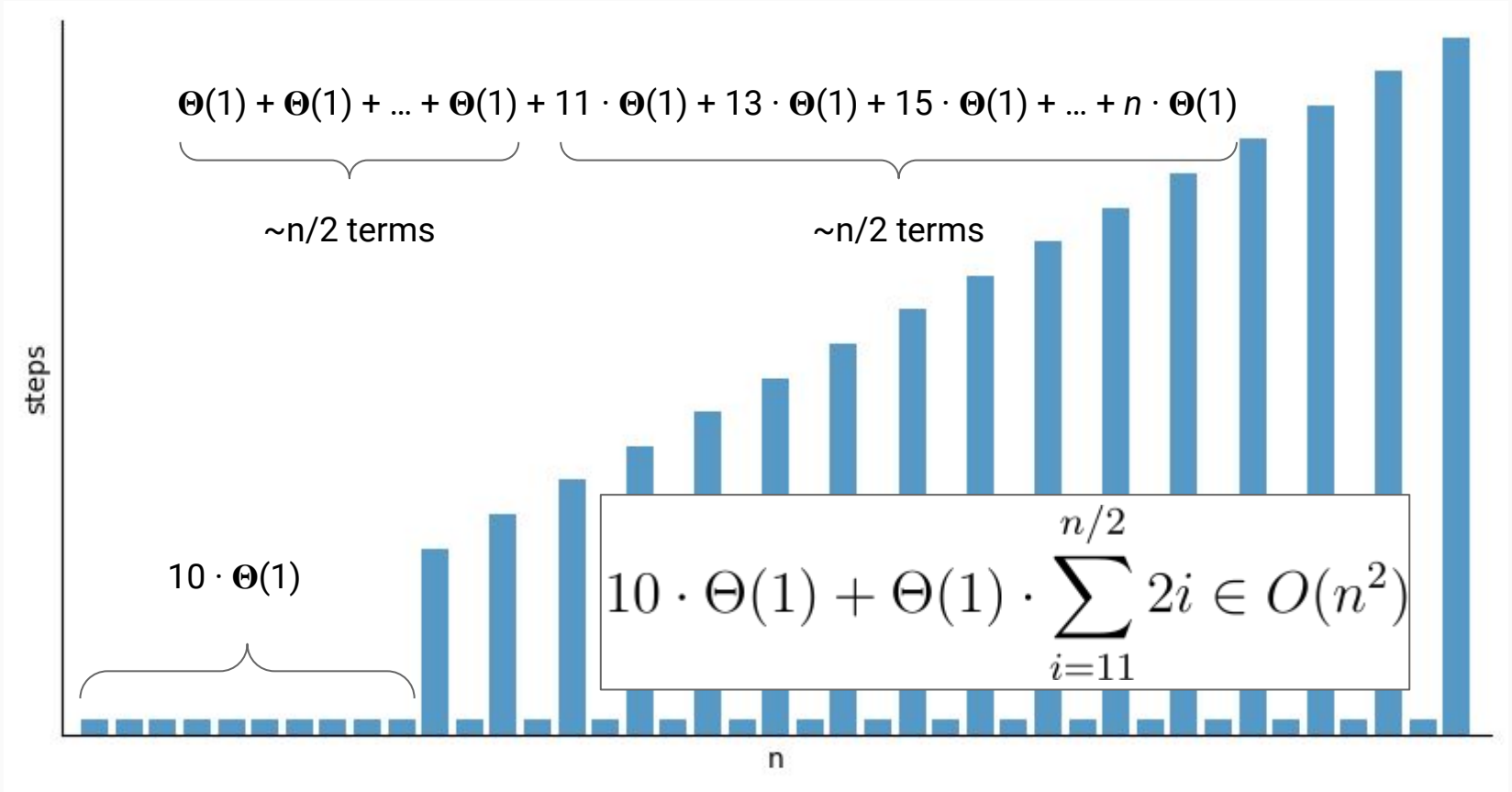
Amortized Runtime

If ***n*** calls to a function take **Θ(f(*n*))** steps

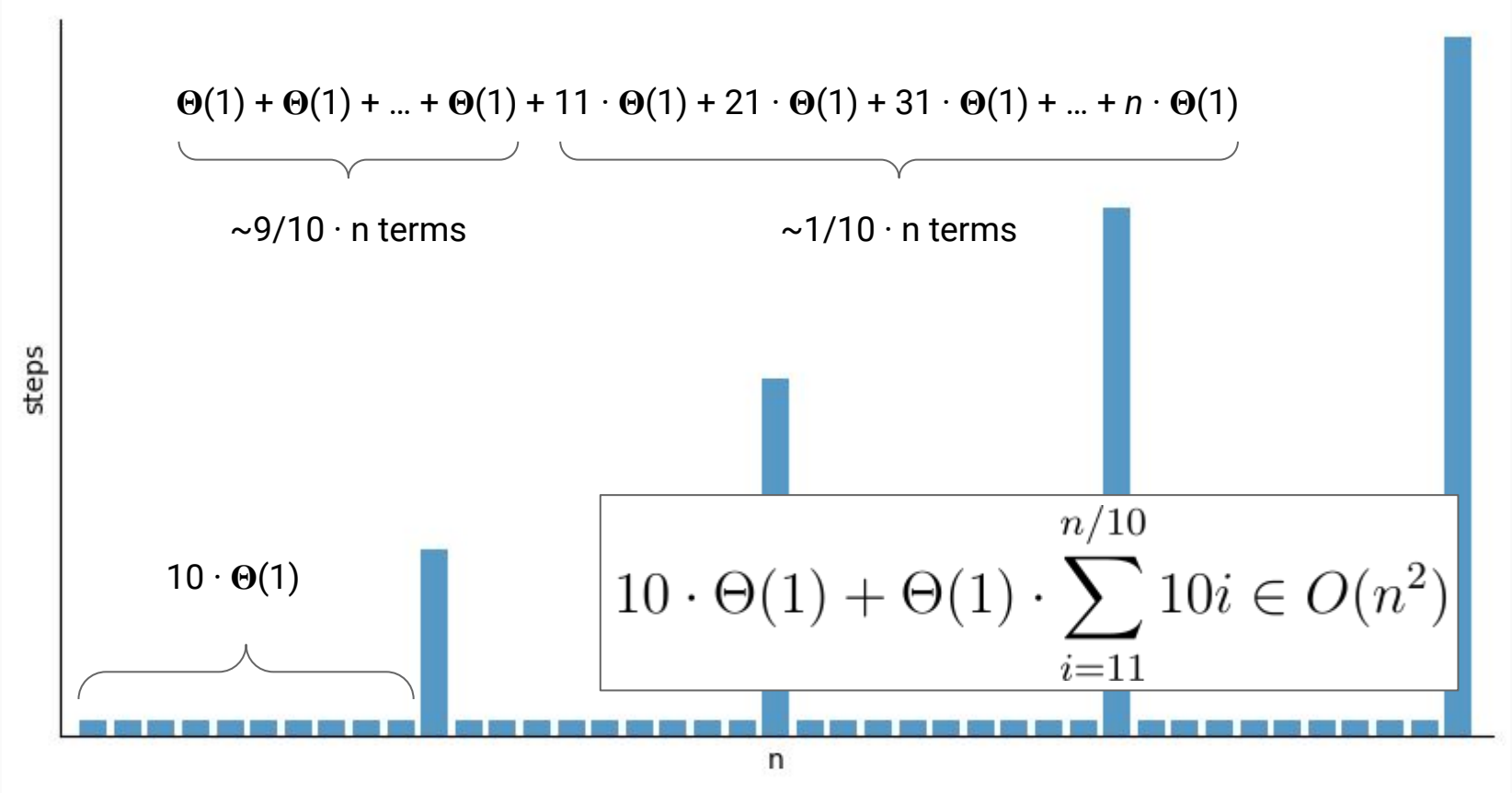
Then the **amortized runtime** of that function is **Θ(f(*n*)/*n*)**



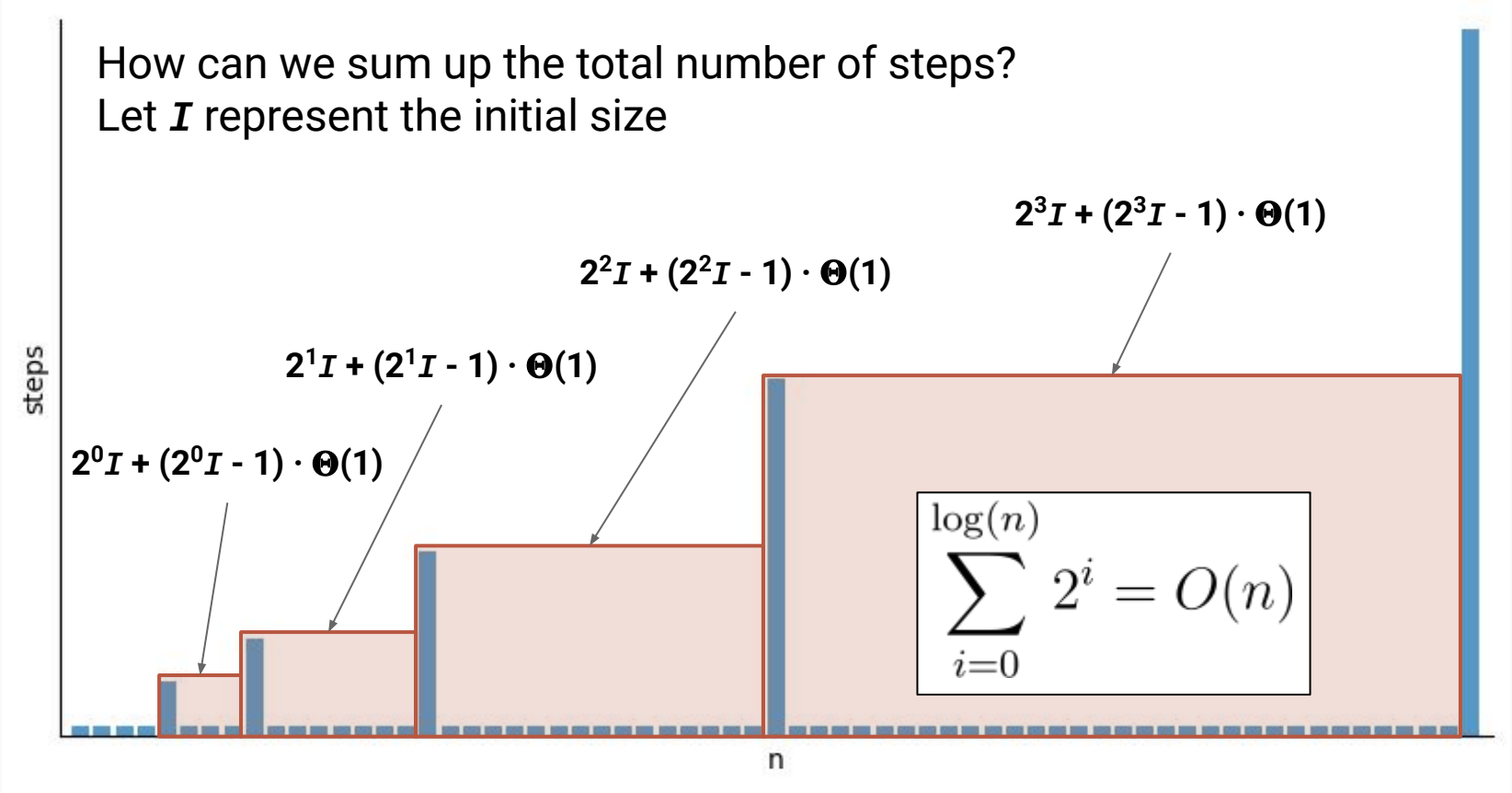
Cost of each call when the initial array size is 10, and newLength = data.length + 1



Cost of each call when the initial array size is 10, and newLength = data.length + 2



Cost of each call when the initial array size is 10, and newLength = data.length + 10



Cost of each call when the initial array size is 4, and newLength = data.length x 2

Amortized Runtime Analysis

```
1  public class Team {
2    private List<Player> players;
3
4    public void addPlayer(Player p) { /* ... */ }
5    public void importRoster(File f) { /* ... */ }
6    /* ... */
7  }
```

```
1  public void addPlayer(Player p) {
2    System.out.println("Welcome to the team " + p.name());
3    players.add(p);
4  }
```

What are the unqualified and amortized runtime bounds of the **`addPlayer`** method when **`List`** is a **`LinkedList`**? an **`ArrayList`**?

Amortized Runtime Analysis

```
1  public void importRoster(File f) {
2    BufferedReader br = new BufferedReader(new FileReader(f));
3    String line;
4    while (br.ready()) {
5      String line = br.readLine();
6      Player p = new Player(line);
7      players.add(p);
8    }
9  }
```

What are the unqualified and amortized runtime bounds of the **`importRoster`** method when **`List`** is a **`LinkedList`**? an **`ArrayList`**?

*(You can assume that opening the file, reading a line, and creating a **`Player`** are constant-time calls)*

Blooket Review:
https://dashboard.blooket.com/set/66f03f2994ef5bc1bcc80c79