**TA Lecture Notes**
Oct 5, 2015

**Puzzle #1 :**

Linear Search: Start from beginning, check every single element until find the target.

**Puzzle #2:**

Binary Search: Start with the middle of a **sorted** list.

**Puzzle #3:**

**Worse case running time**

| **Amount of Data** | **Linear Search** | **Binary Search** | |
|---|---|---|---|
| 7 | 7 | 3 | |
| 15 | 15 | 4 | |
| 31 | 31 | 5 | |
| n | n | $\log_2 n$ | |
| **Advantage** | Data don't has to be sorted | Data has to be sorted | |
| **Disadvantage** | Less efficient | More efficient | |

**Puzzle #4:**

```
var findOpponent = function(opponent, games) {
    for (var i = 0;i<games.length;i++){
        if (opponent==games[i].opponent){
            return games[i];
        }
    }
};
```

**Puzzle #7:**

| **Data Size** | **Linear Search** | **Binary Search** | **Selection Sort** |
|---|---|---|---|
| **9** | 9 | About 4 | 9*9 = 81 |
| **100** | 100 | About 7 | 100*100 = 10,000 |
| **n** | n | $\log_2 n$ | $n^2$ |

**Puzzle #8:**

```
var sortByOpponent = function(games) {

  // Loop through the array:
    for (var i=0;i<games.length;i++){
    // Declare a variable to store the smallest element:
        var min = i;

    // Loop through the array again, looking for the smallest
    // element that has not been put in the correct position:
        for (var j = i+1;j<games.length;j++){
            if (games[i].opponent<games[min].opponent){
                min = i;
            }
        }
    // Swap the smallest element with the current element:
        var temp = games[i];
        games[i]=games[min];
        games[min]=temp;
    }
};
```