Simple array problems

1. Best time to buy and sell stocks https://leetcode.com/problems/best-time-to-buy-and-sell-stock/ (https://leetcode.com/problems/best-time-to-buy-and-sell-stock/)
2. In an array replace every element with a greatest element on the right side https://leetcode.com/problems/replace-elements-with-greatest-element-on-right-side/ (https://leetcode.com/problems/replace-elements-with-greatest-element-on-right-side/)
3. https://leetcode.com/problems/kids-with-the-greatest-number-of-candies/description/ (https://leetcode.com/problems/kids-with-the-greatest-number-of-candies/description/)
4. https://leetcode.com/problems/build-array-from-permutation/description/ (https://leetcode.com/problems/build-array-from-permutation/description/)
5. Reverse a part of array (arr, start, end) Then solve https://leetcode.com/problems/rotate-array/description/ (https://leetcode.com/problems/rotate-array/description/)
6. https://leetcode.com/problems/trapping-rain-water/ (https://leetcode.com/problems/trapping-rain-water/)

In [ ]:

In [ ]:
```
https://leetcode.com/problems/best-time-to-buy-and-sell-stock/



[7,5,3,6,4]
   1
maxProfit = 0
buy = 7
    ->buy
    sell

calcMaxProfit againa.. see if it is larger

update buy price if current price is lower
```

```
class Solution {
    public int maxProfit(int[] prices) {
        int min = prices[0];
        int maxProfit = 0;

        for(int i=0; i<prices.length; i++){
            if(prices[i] < min){
                min = prices[i];
            }
            maxProfit = Math.max(maxProfit, prices[i]-min);
        }
        return maxProfit;
    }
}

[7,5,3,6,4]
i         0 1 2 3 4
maxProfit 0 0 0 3 3
min       7 5 3 3 3


[]
[1]
[1 2 3 4]
[4 3 2 1]
[1 1 1 1]

TC: O(N)
SC: O(1)
```

```java
class Solution {
    public int maxProfit(int[] prices) {
        int minIndex = 0;
        int maxProfit = 0;

        for(int i=0; i<prices.length; i++){
            if(prices[i] < prices[minIndex]){
                minIndex = i;
            }
            if (maxProfit < prices[i]- prices[minIndex] ){
                    maxProfit = prices[i]- prices[minIndex];
                    // i // selling point
                    // minIndex // buy point
            }
        }
        return profit;
    }
}
```

```python
class Solution:
    def maxProfit(self, prices: List[int]) -> int:
        left = 0
        right = 1 # left is buying, right is selling
        maxP = 0

        while right < len(prices):
            if prices[left]<prices[right]:
                profit  = prices[right] - prices[left]
                maxP = max(profit,maxP)
            else:
                left = right
            right = right + 1

        return maxP
```

In [ ]:

https://leetcode.com/problems/replace-elements-with-greatest-element-on-right-side/ (https://leetcode.com/problems/replace-elements-with-greatest-element-on-right-side/)

[17,18,5,4,6,1]

```java
// Brute force solution
class Solution {
public:
    vector<int> replaceElements(vector<int>& arr) {
        for(int i = 0; i < arr.size(); i++) {

            int maxOnRight = -1;
            for (int j = i + 1; j < arr.size(); j++) {
                if (arr[j] > maxOnRight) maxOnRight = arr[j];
            }

            arr[i] = maxOnRight;
        }
    }
};
```

```
[17,18,5,4,6,1]
  0  1 2 3 4 5
i          0           1
j          1 2 3 4 5    2 3 4 5
maxOnRight -1 18     -1 5 5 6 6
```

TC: O(N^2)
SC: O(1)

```python
def replaceElements(self, arr: List[int]) -> List[int]:
        for i in range(len(arr)-1):
            arr[i] = max(arr[i+1:])
        arr[-1] = -1
        return arr
// O(N^2)
```

Optimal Solution

TC: O(N)
SC: O(1)
[17,18,5,4,6,1]

maxOnRight = -1 1 6 6 6 18 18

```cpp
// Brute force solution
class Solution {
public:
    vector<int> replaceElements(vector<int>& arr) {
        int maxOnRight = -1;
        for(int i = arr.size() - 1; i >= 0; i--){

            int curr = arr[i];
            arr[i] = maxOnRight;

            if (curr > maxOnRight) {
                maxOnRight =  curr;
            }
        }

        return arr;
    }
};
```

```java
class Solution {
    public int[] replaceElements(int[] arr) {
        int n = arr.length;
        int maxFromRight = arr[n-1];
        arr[n-1]=-1;
        for(int i = n-2;i>=0;i--)
        {
            int currVal = arr[i];
            arr[i]=maxFromRight;
            maxFromRight = Math.max(maxFromRight, currVal);
        }

        return arr;
    }
}
```

In [ ]:

In [ ]: https://leetcode.com/problems/kids-with-the-greatest-number-of-candies

```python
class Solution:
    def kidsWithCandies(self, candies: List[int], extraCandies: int) -> List[bool]:
        maxCandies = max(candies)

        for i in range(len(candies)):
            candies[i] = (candies[i] + extraCandies) >= maxCandies

        return candies
```

TC: O(N)
SC: O(1)

```python
class Solution:
    def kidsWithCandies(self, candies: List[int], extraCandies: int) -> List[bool]:
        max_val = max(candies) # O(N)

        result = []
        for curr in candies:
            result.append( True if curr + extraCandies >= max_val else False)

        return result
TC: O(N)
SC: O(1) # excluding the output array
```

```java
class Solution {
    public List<Boolean> kidsWithCandies(int[] candies, int extraCandies) {

        List<Boolean> list = new ArrayList<>();
        int max = Integer.MIN_VALUE;
        for(int i=0;i<candies.length;i++)
        {
            if(candies[i]>max)
            {
                max = candies[i];
            }
        }
        for(int i=0;i<candies.length;i++)
        {
            if(candies[i]+extraCandies>=max)
            {
                 list.add(true);
            }
            else
            {
                list.add(false);
            }
        }
        return list;


    }
}
```

```javascript
/**
 * @param {number[]} candies
 * @param {number} extraCandies
 * @return {boolean[]}
 */
var kidsWithCandies = function(candies, extraCandies) {
    let max=Math.max(...candies);
    console.log(max)
    const finalArr = [];
    for(let i=0; i<candies.length; i++){
        const exraCandiesSum = extraCandies +  candies[i];
        if(exraCandiesSum>=max){
            finalArr.push(true)
        }else{
            finalArr.push(false)
        }
    }
    return finalArr;
};
```

In [ ]:

In [ ]:
```
class Solution {
    public List<Boolean> kidsWithCandies(int[] candies, int extraCandies) {
      List<Boolean> bol = new ArrayList();
      int maxArray= Arrays.stream(candies).max().getAsInt();
     for (int i=0;i<candies.length;i++){
         bol.add(candies[i]+extraCandies>=maxArray ? true : false);
     }
    return bol;
    }
}
```

In [ ]:

In [ ]:
```
https://leetcode.com/problems/build-array-from-permutation/description/
```

In [ ]:
```
[0,2,1,5,3,4]
```

```
            int[] res = new int[nums.length];
            for(int i=0;i<nums.length;i++){
                res[i] = nums[nums[i]];
            }
            return res;

        // tc: o(n)
        // sc: o(n) # including the output array for result
```

In [ ]:

In [ ]:
```
[1, 2, 3] Range(1-99)
x100
[100, 200, 300]/100 = [1, 2, 3]

some data is multiplied by 100
[100, 2, 300] => if >= 100 divide else use value => [1, 2, 3]


[1, 2, 3]
store curr and next element data in current position
[102, 203, 300]
102/100 => 1
102%100 => 2


[1, 2, 3]
[201, ]
```

In [ ]:

In [2]:
```
n = 1000

[0,2,1,5,3,4]

for(i = 0; i < arr.size(); i++) {
    curr = arr[i]

    next = arr[arr[i]]
    if next > 1000:
        next = int(next/1000)

    arr[i] = (curr * 1000) + next
}

for ... {

}
```

```
i    0             1                   2                      3                         4                              5
arr [0,2,1,5,3,4]  [0, 2001,1, 5,3,4]  [0, 2001,1002, 5,3,4]  [0, 2001,1002, 5004,3,4] [0, 2001,1002, 5004,3005,4] [0, 2001,1002,
     0 1 2 3 4 5
```

```
  File "C:\Users\LEANGA~1\AppData\Local\Temp/ipykernel_16304/3863041984.py", line 5
    for(i = 0; i < arr.size(); i++) {
          ^
SyntaxError: invalid syntax
```

Constraints
Int size: 2*31 - 1
Problem size: 9991000

In [ ]:
```
```

```
[5,0,1,2,3,4]
 0 1 2 3 4 5
5004, 5, 1005,  2001, 3002, 4003

[4005, 5000, 1, ]

// C++
class Solution {
public:
    vector<int> buildArray(vector<int>& arr) {
        for (int i = 0; i < arr.size(); i++) {
            int curr = arr[i];
            int next = arr[curr];
            if (next > 1000) {
                next = int(next%1000);
            }
            arr[i] = (curr) + next*1000;

            cout << arr[i] << "\t";
        }

        for (int j = 0; j < arr.size(); j++) {
            arr[j] = arr[j]%1000;
        }

        return arr;
    }
};

TC: O(N)
SC: O(1)
```

```
class Solution:
    def buildArray(self, nums: List[int]) -> List[int]:
        for index in range(len(nums)):
            nums[index] = nums[index] + 1000*(nums[nums[index]] % 1000)

        for index in range(len(nums)):
            nums[index] = int(nums[index] / 1000)

        return nums



n = 1000

[0,2,1,5,3,4]

for(i = 0; i < arr.size(); i++) {
    curr = arr[i]

    next = arr[arr[i]]
    if next > 1000:
        next = int(next%1000)

    arr[i] = curr + next * 1000
}

for ... {

}

# i    0             1                           2
# arr [5,0,1,2,3,4]  [4005,5000,1,1002,2003,3004]
# [4,5,0,1,2,3]
```

In [ ]:
```
var buildArray = function(nums) {

    for(i = 0; i < nums.length; i++) {
            let curr = nums[i];
            let next = nums[curr];
            if (next >= 1000) {
                next = parseInt(next%1000);
            }
            nums[i] = curr + (1000*next);


    }
   for (let j = 0; j < nums.length; j++) {
        nums[j] = nums[j]/1000;
    }

        return nums;


};
```

In [ ]:
```
reverse(arr, start, end):
i = 0
j = len(arr) - 1
while i < j:
    arr[i], arr[j] = arr[j], arr[i]
    i++1
    j-=1
```