DSA SERIES

- Learn Coding

Topic to be Covered today

Greedy Algorithms

1833. Maximum Ice Cream Bars

```
class Solution {
public:
    int maxIceCream(vector<int>& costs, int
coins) {
        sort(costs.begin(), costs.end());
        int count = 0;
        for (int& cost : costs) {
            if (cost > coins) {
                return count;
            } else {
                count++;
                coins -= cost;
            }
        }


        return count;
    }
};
```

2279. Maximum Bags With Full Capacity of Rocks

```
class Solution {
public:
    int maximumBags(vector<int>& capacity, vector<int>& rocks,
                    int additionalRocks) {
        int n = rocks.size();


        vector<int> diff(n, 0);


        for (int i = 0; i < n; i++) {
            diff[i] = capacity[i] - rocks[i];
        }


        sort(diff.begin(), diff.end());


        int count = 0;
        for (int i = 0; i < n; i++) {
            if (diff[i] == 0) {
                count++;
            }


            else {
                if (additionalRocks >= diff[i]) {
                    additionalRocks -= diff[i];
                    count++;
                } else {
                    break;
                }
            }
        }
        return count;
    }
};
```

1323. Maximum 69 Number

```
class Solution {
public:
    int maximum69Number (int num) {
        string s = to_string(num);
        int n = s.length();


        for(int i=0;i<n;i++){
            if(s[i]=='9'){
                continue;
            } else{
                s[i]='9';
                break;
            }
        }


        return stoi(s);
    }
};
```

1328. Break a Palindrome

```
class Solution {
public:
    string breakPalindrome(string palindrome) {
        int n = palindrome.length();


        if(n==1) return "";


        for(int i =0;i<n/2;i++){
            if(palindrome[i] != 'a'){
                palindrome[i]='a';
                return palindrome;
            }
        }


        palindrome[n-1]= 'b';
        return palindrome;
    }
};
```

881. Boats to Save People

```
class Solution {
public:
    int numRescueBoats(vector<int>& people, int
limit) {
        int n = people.size();

        sort(begin(people), end(people));

        int i = 0, j = n - 1;

        int count = 0;

        while (i <= j) {
            if (people[j] + people[i] <= limit)
{
                i++;
                j--;

            } else {
                j--;

            }
            count++;
        }

        return count;
    }
};
```

Thank You