HOMEWORK DAY: 16

28 September 2023 09:15

CLASS-16 - 27/09/2023

https://www.linkedin.com/in/manojoffcialmj/

CHAR ARRAYS & STRINGS LEVEL-3

Program 01: Decode the Message (Leetcode-2325)

EX:01

Key

T	H	E		D	O	G	T

Message

D	O	G	E

Output

D	E	F	C

Approach:
Step 01: *Create mapping for key*
Step 02: *Use mapping to decode the message*
Step 03: *Return decoded message*

Stup:01

Key

T	H	E	space	D	O	G	T

MAPPING

A	B	C	space	D	E	F	A

Stup:02

Message

D	O	G	E

Decoded Message

D	E	F	C

→ Return as output

DRY RUN

KEY

T	H	E		D	O	G	T

CHAR	ASCII
T	116
H	104
E	101
D	100
O	111
G	103
T	116
Space	0

It's Important to understand this problem

STEP:01

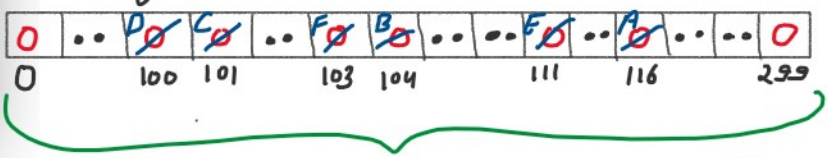
```
// Step 01: Create mapping for key
1 char start = 'a';
2 char mapping[300] = {0};
3 for(auto ch: key)
  {
    1 if(ch != ' ' && mapping[ch] == 0)
      {
        1 mapping[ch] = start;
        2 start++;
      }
  }
```

KEY

T	H	E		D	O	G	T

START = A

Mapping [300]



300 Blocks in mapping [300]

Iteration

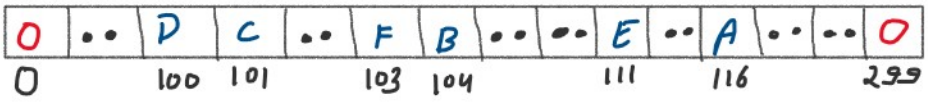
CH = T	→	mapping [116] == A	start = B
CH = H	→	mapping [104] == B	start = C
CH = E	→	mapping [101] == C	start = D
CH = Space	→	Ch != ' ' FALSH	start = D
CH = D	→	mapping [100] == D	start = E
CH = O	→	mapping [111] == E	start = F
CH = G	→	mapping [103] == F	start = F
CH = E	→	mapping [116] == A TRUE	

CH = G → mapping [103] == F

CH = E → mapping [116] == A TRUE

STEP:2

Mapping



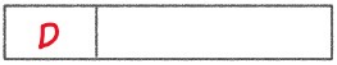

```
// Step 02: Use mapping to decode the message
string ans;
for(auto ch: message)
{
    if(ch == ' ')
    {
        ans.push_back(' ');
    }
    else
    {
        char decodeChar = mapping[ch];
        ans.push_back(decodeChar);
    }
}
```

Message

Situation

Ch = D → decodeChar = mapping [100]
= D

ans



Ch = O → decodeChar = mapping [111]
= E

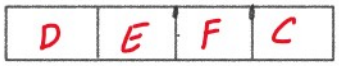
ans

Ch = G → decodeChar = mapping [103]
= F

ans

Ch = E → decodeChar = mapping [101]
= C

ans



STEP:03

Decoded Message

ANS

→ Return as output

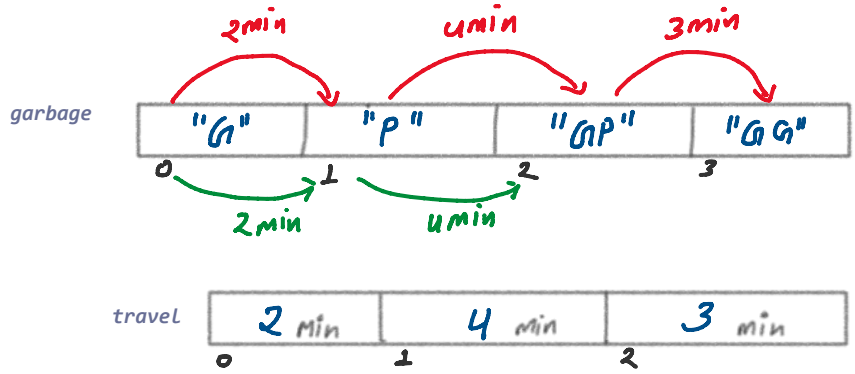
```
// Program 01: Decode the Message (Leetcode-2325)
class Solution {
public:
    string decodeMessage(string key, string message) {
        // Step 01: Create mapping for key
        char start = 'a';
        char mapping[300] = {0};

        for(auto ch: key){
            if(ch != ' ' && mapping[ch] == 0){
                mapping[ch] = start;
                start++;
            }
        }

        // Step 02: Use mapping to decode the message
        string ans;
```

```cpp
// Program 01: Decode the Message (Leetcode-2325)
class Solution {
public:
    string decodeMessage(string key, string message) {
        // Step 01: Create mapping for key
        char start = 'a';
        char mapping[300] = {0};

        for(auto ch: key){
            if(ch != ' ' && mapping[ch] == 0){
                mapping[ch] = start;
                start++;
            }
        }

        // Step 02: Use mapping to decode the message
        string ans;
        for(auto ch: message){
            if(ch == ' '){
                ans.push_back(' ');
            }
            else{
                char decodeChar = mapping[ch];
                ans.push_back(decodeChar);
            }
        }
        // Step 03: Return decoded message
        return ans;
    }
};

/*
Time Complexity: O(M)+ O(N) = O(M+N), where
M is a legth of key string and
N is a length of message string

Space Complexity: O(1) + O(N) = O(N), where
O(1) -> mapping size is constant
O(N) -> ans store the decode message
*/
```

Program 02: Minimum Amount of Time to Collect Garbage (Leetcode-2391)

Example 1:	*Example 2:*
Input: garbage = ["G","P","GP","GG"], travel = [2,4,3]	*Input: garbage = ["MMM","PGM","GP"], travel = [3,10]*
Output: 21	*Output: 37*

EXAMPLE : 01



M — TRAVEL 1 — PICK TIME = 0, TRAVEL TIME = 0 → 0+0 = 0 minutes

P — TRAVEL 2 — PICK TIME = 1 + 1 = 2, TRAVEL TIME = 2 + 4 = 6 → 2+6 = 8 minutes

G — TRAVEL 3 — PICK TIME = 1 + 0 + 1 + 2 = 4 → 4+9 = 13 minutes

G

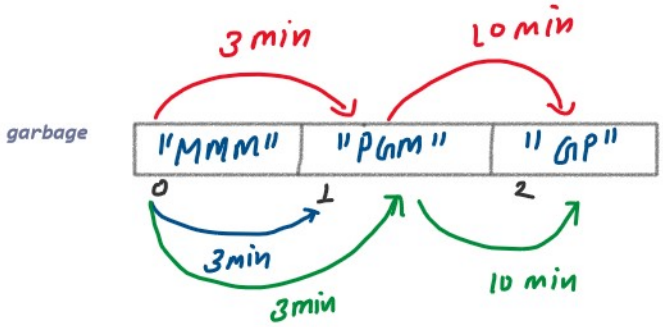
Truck 3

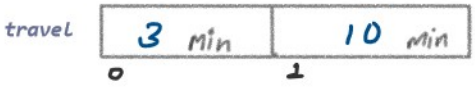
PickTime = 1 + 0 + 1 + 2 = 4
TravelTime = 2 + 4 + 3 = 9
> 4 + 9 = 13 minutes

Final Ans = 0 + 8 + 13

= 21 minutes

EXAMPLE : 02





M

Truck 1

Pick Time = 3 + 1 = 4
Travel Time = 3 = 3
> 4 + 3 = 7 min

P

Truck 2

Pick Time = 0 + 1 + 1 = 2
Travel Time = 3 + 10 = 13
> 13 + 2 = 15 min

G

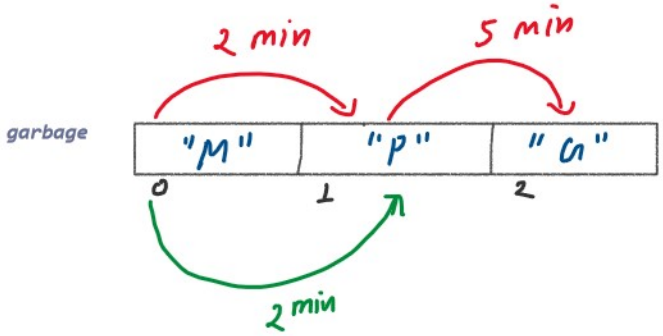
Truck 3

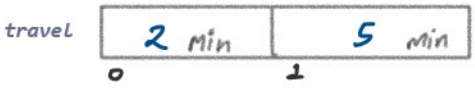
PickTime = 0 + 1 + 1 = 2
TravelTime = 3 + 10 = 13
> 2 + 13 = 15 min

Final Ans = 7 + 15 + 15

= 37 minutes

EXAMPLE : 03





M — Truck 1
Pick Time = 1 = 1
Travel Time = 0 = 0
> 1+0 = 1 min

P — Truck 2
Pick Time = 1 = 1
Travel Time = 2 = 2
> 1+2 = 3 min

G — Truck 3
PickTime = 1 = 1
Travel Time = 2+5 = 7
> 1+7 = 8 min

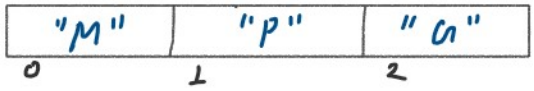
Final Ans = 1 + 3 + 8
= 12 minutes

DRY RUN

Step:01

```
// Step 01: Calculate pick time and find travel
time index from 0th index to last index

        for(int i=0; i<garbage.size();i++)
        {

            string str = garbage[i];

            for(auto ch: str)
            {
                if(ch == 'M')
                {
                    pickM++;
                    lastM = i;
                }
                if(ch == 'P')
                {
                    pickP++;
                    lastP = i;
                }
                if(ch == 'G')
                {
                    pickG++;
                    lastG = i;
                }
            }
        }
```

Example:03

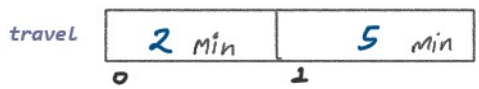


i=0 str = "M" ch = M
i=1 str = "P" ch = P
i=2 str = "G" ch = G

	Pick Time	Last Index
pickM	0 ~~1~~	lastM 0
pickP	~~0~~ 1	lastP ~~0~~ 1
pickG	~~0~~ 1	lastG ~~0~~ 2

STEP:02

```
// Step 02: Calculate travel time from 0th index to last index
        for(int i=0;i<lastM;i++)
1 C     {
            travelM += travel[i];
        }
        for(int i=0;i<lastP;i++)
2 C     {
            travelP += travel[i];
        }
        for(int i=0;i<lastG;i++)
3 C     {
            travelG += travel[i];
        }
```



1st Loop

travelM | 0

i=0 i<0 ✗

2nd Loop

i<1 travelP += travel[0]

2nd Loop

travelP: ~~0~~ 2

i=0 i<1

i=1 1<1 ✗

travelP += travel[0]
= 0+2
= 2

3rd Loop

pickG: ~~0~~ 2

pickG: ~~2~~ 7

i=0 i<2

i=1 i<2

i=2 2<2 ✗

travelG += travel[0]
= 0+2
= 2

travelG += travel[1]
= 5+2
= 7

Step: 03

```
// Step 03: Return sum of travel and pick time
                      1 + 0          1 + 2         1 + 7
    int finalAns = (pickM + travelM) + (pickP + travelP) + (pickG + travelG);

    return finalAns; // 12 Minutes
```

Complete Program

```
// Program 02: Minimum Amount of Time to Collect Garbage (Leetcode-2391)
class Solution {
public:
    int garbageCollection(vector<string>& garbage, vector<int>& travel) {

        // initial pick time of M, P and G
        int pickM = 0, pickP = 0, pickG = 0;

        // initial travel time of M, P and G
        int travelM = 0, travelP = 0, travelG = 0;

        // i have to count travel time from 0th index to lastM, lastP, and lastG index
        int lastM = 0, lastP = 0, lastG = 0;

        // Step 01: Calculate pick time and find travel time index from 0th index to last index
        for(int i=0; i<garbage.size();i++){

            // mujhe yanha ek string mil jayegi to mujhe us string ko traverse karna padega each character par
            //  to calculate the pick time and find last index to traverse the travel array
            string str = garbage[i];

            for(auto ch: str){
                if(ch == 'M'){
                    pickM++;
                    lastM = i;
                }
                if(ch == 'P'){
                    pickP++;
                    lastP = i;
                }
                if(ch == 'G'){
                    pickG++;
                    lastG = i;
                }
            }
        }

        // Step 02: Calculate travel time from 0th index to last index
        for(int i=0;i<lastM;i++){
            travelM += travel[i];
        }
        for(int i=0;i<lastP;i++){
            travelP += travel[i];
        }
        for(int i=0;i<lastG;i++){
            travelG += travel[i];
        }

        // Step 03: Return sum of travel and pick time
        int finalAns = (pickM + travelM) + (pickP + travelP) + (pickG + travelG);
        return finalAns;
    }
};

/*
Time Complexity: O(N)+ O(lastM) + O(lastP) + O(lastG) = O(N), where N is a length of garbage vector

Space Complexity: O(1), where no extra space used
```

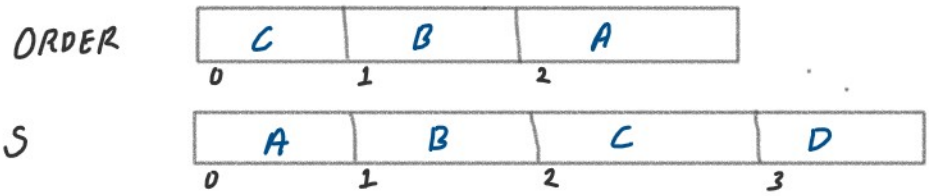
```
};

/*
Time Complexity: O(N)+ O(lastM) + O(lastP) + O(lastG) = O(N), where N is a length of garbage vector

Space Complexity: O(1), where no extra space used
*/

/*
Example 1:
Input: garbage = ["G","P","GP","GG"], travel = [2,4,3]
Output: 21

Example 2:
Input: garbage = ["MMM","PGM","GP"], travel = [3,10]
Output: 37

Example 3:
Input: garbage = ["M","P","G"], travel = [2,5]
Output: 12
*/
```

Program 03: Custom Sort String (Leetcode-791)

Example 1:	Example 2:
Input: order = "cba", s = "abcd" Output: "cbad"	Input: order = "cbafg", s = "abcd" Output: "cbad"

Approach:
Step 01: Use custom comparator to arrange character of string s according to string order
Step 02: in custom comparator, find character's position of s string and compare b/w both a and b positions in string order
Step 03: Return arranged string s

Example: 01



Output ⇒ "CBAD", "CDBA", "CBDA", or "DCBA"

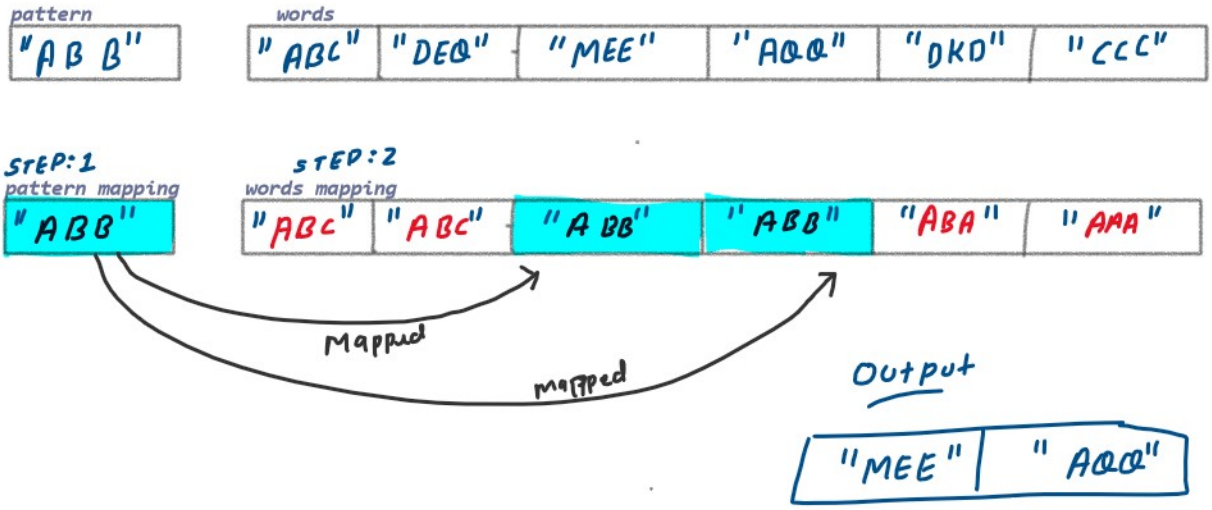
```cpp
// Program 03: Custom Sort String (Leetcode-791)
class Solution {
public:
    // Step 02: in custom comparator, find character's position of s string in string order
    static string orderStr;
    static bool compare(char ch1, char ch2){
        char a = orderStr.find(ch1);
        char b = orderStr.find(ch2);

        // Compare b/w both a and b positions to arrange character in string s
        // If a > b = b pahle ayega or bad me a ayega
        // If a < b = a pahle ayega or bad me b ayega

        // output string
        return a < b;
    }

    string customSortString(string order, string s) {

        orderStr = order;

        // Step 01: Use custom comparator to arrange character of string s according to string order
        sort(s.begin(), s.end(), compare);

        // Step 03: Return arranged string s
        return s;
    }
};
string Solution::orderStr;

/*
Time Complexity: O(L) + O(NlogN) = (NlogN), where
L is the length of the string order
N is a length of string s
```

```
/*
Time Complexity: O(L) + O(NlogN) = (NlogN), where
L is the length of the string order
N is a length of string s

Space Complexity: O(1), where no extra space used
*/
```

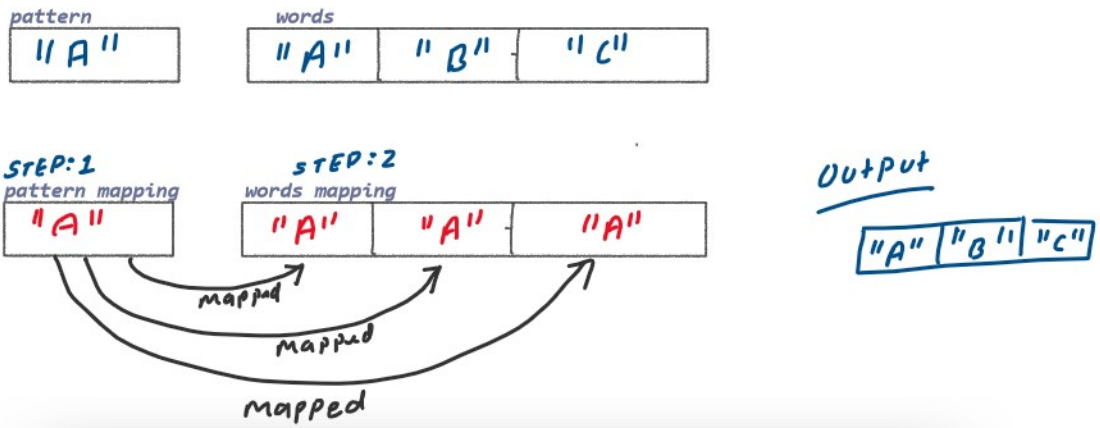
Program 04: Find and Replace Pattern (Leetcode-890)

Example 1:	Example 2:
Input: words = ["abc","deq","mee","aqq","dkd","ccc"], pattern = "abb" Output: ["mee","aqq"]	Input: words = ["a","b","c"], pattern = "a" Output: ["a","b","c"]

EX:01



EX:02




```cpp
// Program 04: Find and Replace Pattern (Leetcode-890)
class Solution {
public:
    // Create mapping and update string
    void createUpdateMapping(string& str) {
        // 1. find mappping
        char start = 'a';
        char mapping[300] = {0};

        for(auto ch: str) {
            if(mapping[ch] == 0) {
                mapping[ch] = start;
                start++;
            }
        }

        // 2. update the string
        for(int i=0; i<str.length(); i++) {
            char ch = str[i];
            str[i] = mapping[ch];
```

```
        }

        // 2. update the string
        for(int i=0; i<str.length(); i++) {
            char ch = str[i];
            str[i] = mapping[ch];
        }
    }

    // Find and replace pattern
    vector<string> findAndReplacePattern(vector<string>& words, string pattern) {

        vector<string> ans;

        // Step 01: Create mapping and update string for string pattern
        createUpdateMapping(pattern);

        // Step 02: Create mapping and update string for all strings of string type array words
        for(string str: words){

            string tempStr = str;
            // Normalise tempStr
            createUpdateMapping(tempStr);

            //  compare b/w temp string and mapped pattern
            if(tempStr == pattern){
                // push correct str in ans
                ans.push_back(str);
            }
        }
        // Step 03: Return vector ans
        return ans;
    }
};

/*
Time Complexity: ?

Space Complexity: ?
*/
```