transaction-logs

https://aonecode.com/amazon-online-assessment-transaction-logs

def processLogFile(logs, threshold):

"""

:type logs: List[str]

:type threshold: int

:rtype: List[str]

"""

temp=[]

for log in logs:

s, r, amt = log.split()

if(s==r):

temp.append(s)

continue

temp.append(s)

temp.append(r)

dict\_ele = {}

for i in temp:

if i in dict\_ele.keys():

dict\_ele[i]+=1

else:

dict\_ele[i]=1

result\_list = []

for k,v in dict\_ele.items():

if(v>=threshold):

result\_list.append(k)

result\_list.sort()

return result\_list

LRU

https://aonecode.com/amazon-online-assessment-lru

from collections import OrderedDict

def lruCacheMisses(num: int, pages: list, maxCacheSize: int) -> int:

cache = OrderedDict()

output = 0

for i in pages:

if i not in cache.keys():

output+=1

cache[i] = i

cache.move\_to\_end(i)

if len(cache) > maxCacheSize:

cache.popitem(last = False)

else:

cache.move\_to\_end(i)

return output

schedule-deliveries

https://aonecode.com/interview-question/schedule-deliveries

https://leetcode.com/discuss/interview-question/992156/Amazon-or-OA-or-Earliest-Time-To-Complete-Deliveries

int func(int numOfPiers, vector<int> pierOpenTime, vector<int> deliveryTime)

{

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

sort(deliveryTime.begin(), deliveryTime.end(), greater<int>());

int max=0;

for(int i=0; i<deliveryTime.size(); i=i+4)

{

if( (pierOpenTime[i/4]+deliveryTime[i]) > max)

{

max = pierOpenTime[i/4]+deliveryTime[i];

}

}

return max;

}

int main() {

int numOfPiers = 2;

vector<int> pierOpenTime;

pierOpenTime.push\_back(8);

pierOpenTime.push\_back(10);

vector<int> deliveryTime;

deliveryTime.push\_back(2);

deliveryTime.push\_back(2);

deliveryTime.push\_back(3);

deliveryTime.push\_back(1);

deliveryTime.push\_back(8);

deliveryTime.push\_back(7);

deliveryTime.push\_back(4);

deliveryTime.push\_back(5);

cout<<func(numOfPiers, pierOpenTime, deliveryTime);

//numOfPiers = 2

// pierOpenTime = [7, 9]

// deliveryTime = [7,6,3,4,1,1,2,0]

// Output:14

// numOfBuildings = 2

// buildingOpenTime = [8, 10]

// offloadTime = [2,2,3,1,8,7,4,5]

// Output: 16

}

tagging-system

https://aonecode.com/interview-question/tagging-system

https://discuss.codechef.com/t/amazon-coding-question-2020-help/81298

int main()

{

map<char,int>m;

priority\_queue<char> pq;

int k=2;

string original="baccc";

// cin>>original;

int n=original.size();

for(int i=0;i<n;i++)

m[original[i]]++;

for(auto x:m)pq.push(x.first);

string ans;

ans+=pq.top();

m[pq.top()]--;

if(m[pq.top()]==0){

pq.pop();

}

int same=1;

int c=1;

while(!pq.empty()){

if (same>=k){

char tp=pq.top();

if(tp!=ans[c-1]){

ans+=pq.top();

m[pq.top()]--;

if(m[pq.top()]==0){

m.erase(pq.top());

pq.pop();

}

}

else{

pq.pop();

ans+=pq.top();

m[pq.top()]--;

if(m[pq.top()]==0){

m.erase(pq.top());

pq.pop();

}

pq.push(tp);

same=1;

}

}

else{

ans+=pq.top();

m[pq.top()]--;

if(m[pq.top()]==0){

m.erase(pq.top());

pq.pop();

}

}

if (ans[c-1]==ans[c])same++;

c++;

}

cout<<ans;

// Input:

// originalLabel = baccc

// charLimit = 2

// Output:

// ccbca

// originalLabel = cbddd

// charLimit = 2

// Output:

// ddcdb

}

merge-two-sorted-lists

https://leetcode.com/problems/merge-two-sorted-lists/submissions/

/\*\*

\* Definition for singly-linked list.

\* struct ListNode {

\* int val;

\* ListNode \*next;

\* ListNode() : val(0), next(nullptr) {}

\* ListNode(int x) : val(x), next(nullptr) {}

\* ListNode(int x, ListNode \*next) : val(x), next(next) {}

\* };

\*/

class Solution {

public:

ListNode\* mergeTwoLists(ListNode\* l1, ListNode\* l2) {

ListNode \*head,\*prev,\*temp;

if(l1==NULL)

return l2;

if(l2==NULL)

return l1;

if(l1->val < l2->val)

{

temp = new ListNode();

temp->val = l1->val;

temp->next=NULL;

l1 = l1->next;

}

else

{

temp = new ListNode();

temp->val = l2->val;

temp->next=NULL;

l2 = l2->next;

}

head = temp;

prev = temp;

while(l1!=NULL && l2!=NULL)

{

if(l1->val < l2->val)

{

temp = new ListNode();

temp->val = l1->val;

temp->next=NULL;

l1 = l1->next;

}

else

{

temp = new ListNode();

temp->val = l2->val;

temp->next=NULL;

l2 = l2->next;

}

prev->next = temp;

prev=temp;

}

if(l1)

prev->next = l1;

if(l2)

prev->next = l2;

return head;

}

};

throttling-gateway

https://aonecode.com/amazon-online-assessment-throttling-gateway

int main()

{

int arriveTime[] = {1, 1, 1, 1, 2, 2, 2, 3, 3, 3, 4, 4, 4, 5, 5, 5, 6, 6, 6, 7,7,7,7, 11, 11, 11, 11};

int num = 27;

unordered\_map<char, int> m1;

int count=0;

for(int i=0;i<27;i++)

{

//cout<<"lol "<<arriveTime[i]<<endl;

if(m1.find(arriveTime[i]) != m1.end())

{

m1[arriveTime[i]]+=1;

}

else

{

m1[arriveTime[i]]=1;

}

int flag=0;

//more than 3 in a minute

if(m1[arriveTime[i]]>3)

{

flag=1;

}

// more than 20 in last 10 mins

int sum=0;

for(int j=max(arriveTime[i]-9,1);j<=arriveTime[i];j++)

{

if(m1.find(arriveTime[j]) != m1.end())

{

sum+=m1[j];

}

}

if(sum>20)

{

flag=1;

}

if(flag==1)

count++;

}

cout<<count;

}

Divisibility-of-Strings

https://leetcode.com/discuss/general-discussion/680341/Divisibility-of-Strings

int solve(string s1, string s2)

{

if(s1.length()%s2.length()!=0)

return -1;

int l2 = s2.length();

for(int i=0;i<s1.length();i++) {

if(s1[i] != s2[i%l2])

return -1;

}

int min=-1;

for(int i=s2.length()-1;i>=0;i=((i+1)/2)-1) {

int j=0;

for(;j<s2.length();j++) {

if(s2[j] != s2[j%(i+1)])

break;

}

if(j == s2.length()) {

min=i+1;

}

}

return min;

}

int main()

{

string a = "abcabcabcabc";

string b = "abcabc";

int flag=0;

cout<<solve(a,b);

}

schedule-tasks

https://aonecode.com/interview-question/schedule-tasks

void place(vector<int> &a)

{

int h = a.size()-1;

int temp = a[h];

h--;

while(a[h]>temp)

{

a[h+1] = a[h];

h--;

}

a[h+1] = temp;

}

int solve(vector<int> a, int n)

{

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

int count=0;

while(n>0)

{

n = n-a[a.size()-1];

a[a.size()-1] = a[a.size()-1]/2;

place(a);

count++;

}

return count;

}

int main()

{

vector<int> a;

a.push\_back(4);

a.push\_back(2);

a.push\_back(8);

a.push\_back(3);

a.push\_back(5);

cout<<solve(a,19);

}

slowest-key

https://aonecode.com/interview-question/key-pressed-times

https://leetcode.com/problems/slowest-key/

char solve(vector<pair<int, int> > vec1)

{

unordered\_map<int, vector<char>> m1;

int prev=0;

int diff;

int max=-1;

for(int i=0;i<vec1.size();i++)

{

diff = vec1[i].second - prev;

if(diff>max)

{

max=diff;

}

m1[diff].push\_back(vec1[i].first);

prev = vec1[i].second;

}

return m1[max][0]+'a';

}

int main()

{

vector<pair<int, int> > vec1 = { {0, 1}, {3,4}, {0,8}, {2,11}};

cout<<solve(vec1);

}

UniquePairs

https://leetcode.com/discuss/interview-question/372434

int countUniquePairs\_optimized(vector<int> nums, int target)

{

unordered\_set<int> seen;

unordered\_set<int> numsSet;

int count = 0;

for (int n : nums){

int complement = target - n;

if (numsSet.find(complement) != numsSet.end()){

if(seen.find(complement) == seen.end()){

count++;

}

seen.insert(n);

seen.insert(complement);

}

numsSet.insert(n);

}

return count;

}

int main()

{

vector<int> test1 = {1, 1, 2, 45, 46, 46};

int target1 = 47;

vector<int> test2 = {1, 1};

int target2 = 2;

vector<int> test3 = {1, 5, 1, 5};

int target3 = 6;

int res = countUniquePairs\_optimized(test1, target1);

cout << "result is " << res << endl;

return 0;

}

search-suggestions-system

https://leetcode.com/problems/search-suggestions-system/submissions/

class Solution {

public:

vector<vector<string>> suggestedProducts(vector<string>& products, string searchWord) {

vector<vector<string>> result;

vector<string> prev = products;

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

{

vector<string> temp;

vector<string> temp\_res;

for(int j=0; j<prev.size(); j++)

{

if(prev[j][i]==searchWord[i])

{

temp.push\_back(prev[j]);

}

}

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

if(temp.size()>3)

{

for(int k=0;k<3;k++)

{

temp\_res.push\_back(temp[k]);

}

}

else

{

temp\_res = temp;

}

result.push\_back(temp\_res);

prev = temp;

}

return result;

}

};

unique-device-names

https://aonecode.com/amazon-online-assessment-unique-device-names

void solve(vector<string>& test)

{

unordered\_map<string, int> m1;

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

{

if(m1.find(test[i]) != m1.end())

{

m1[test[i]]++;

test[i] += (m1[test[i]]-1+'0');

}

else

{

m1[test[i]]=1;

}

}

}

int main() {

vector<string> test = {"switch", "tv", "switch", "tv","switch", "tv"};

solve(test);

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

{

cout<<test[i]<<endl;

}

}

most-common-word

https://leetcode.com/problems/most-common-word/submissions/

class Solution {

public:

string mostCommonWord(string paragraph, vector<string>& banned) {

vector<string> words;

unordered\_map<string, int> m1;

int prev=0;

int i=0;

string max\_str;

int maxi=0;

for(i=0;i<paragraph.length();i++)

{

if(iswspace(paragraph[i]) || paragraph[i]==',' || paragraph[i]=='.' || paragraph[i]=='!' || paragraph[i]=='?' || paragraph[i]==';' || paragraph[i]=='\'')

{

string temp = paragraph.substr(prev,i-prev);

if(temp==" " || temp=="")

{

prev = i+1;

continue;

}

transform(temp.begin(), temp.end(), temp.begin(), ::tolower);

words.push\_back(temp);

prev = i+1;

}

}

if(isalpha(paragraph[i-1]))

{

string temp = paragraph.substr(prev,i-prev);

transform(temp.begin(), temp.end(), temp.begin(), ::tolower);

words.push\_back(temp);

}

for(int j=0;j<words.size();j++)

{

string temp = words[j];

if (std::find(banned.begin(), banned.end(), temp) == banned.end())

{

if(m1.find(temp) != m1.end())

{

m1[temp]++;

if(m1[temp] > maxi)

{

maxi = m1[temp];

max\_str = temp;

}

}

else

{

m1[temp]=1;

if(m1[temp] > maxi)

{

maxi = m1[temp];

max\_str = temp;

}

}

}

}

return max\_str;

}

};

longest-palindromic-substring

https://leetcode.com/problems/longest-palindromic-substring/

class Solution {

public:

string longestPalindrome(string s) {

if(s.length()==1)

return s;

int arr[s.length()][s.length()];

for(int i=0;i<s.length();i++)

{

for(int j=0;j<s.length();j++)

{

arr[i][j]=0;

}

}

for(int i=0;i<s.length();i++)

{

arr[i][i]=1;

}

int low=0;

int high=0;

int max = 1;

for(int l = 2; l <= s.length(); l++){

for(int i = 0; i < s.length()-l + 1; i++){

int j = i + l - 1;

if(l == 2 && s[i] == s[j]){

arr[i][j] = 2;

}else if(s[i] == s[j]){

if(arr[i+1][j-1]!=0)

arr[i][j] = arr[i + 1][j-1] + 2;

}else{

arr[i][j] = 0;

}

if(arr[i][j]>max)

{

max = arr[i][j];

low=i;

high=j;

}

}

}

return s.substr(low, high-low+1);

}

};

Balanced Parenthesis

<https://leetcode.com/playground/9N8nocBm>

char stacki[100];

int top=-1;

void push(char m)

{

top++;

stacki[top]= m;

}

char seek()

{

return stacki[top];

}

void pop()

{

top--;

}

int solve(string str)

{

int res=0;

for(int i=0;i<str.length();i++)

{

if(str[i]=='(')

{

// cout<<"hi";

push(str[i]);

}

if(str[i]==')')

{

if(seek()!='(')

return -1;

else

{

pop();

res++;

}

}

}

if(top!=-1)

return -1;

return res;

}

int main() {

string str = "(()(()";

cout<<solve(str);

}