**Day-4**

**Problem1**- **Word Break (Trie)**

Solution-

//{ Driver Code Starts

//Initial template for C++

#include <bits/stdc++.h>

using namespace std;

// } Driver Code Ends

//User function template for C++

class Solution{

public:

// A : given string to search

// B : vector of available strings

int solve( int i,string &s,set<string>&set,vector<int>&dp)

{

if(i==s.size())return 1;

string temp;

if(dp[i]!=-1)return dp[i];

for(int j=i;j<s.size();j++)

{ temp+=s[j];

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

{

// cout<<temp<<endl;

if(solve(j+1,s,set,dp)) return dp[i]=1;

}

}

return dp[i]=0;

}

int wordBreak(string A, vector<string> &B) {

//code here

int n=A.size();

set<string>set;

vector<int>dp(n,-1);

for(auto i:B)

set.insert(i);

return solve(0,A,set,dp);

}

};

**Problem2**- **PALINDROMIC PAIRS**

Solution-

class Solution {

public:

 bool ispalindrome(const string& s) {

        int left = 0, right = s.size() - 1;

        while (left < right) {

            if (s[left++] != s[right--]) return false;

        }

        return true;

    }

    vector<vector<int>> palindromePairs(vector<string>& w) {

        vector<vector<int>>ans;

        int n=w.size();

        if(n<2)return ans;

        unordered\_map<string,int>m;

        for(int i=0;i<n;i++)

        {

            string z=w[i];

            reverse(z.begin(),z.end());

            m[z]=i;

        }

        for(auto i:m)

        cout<<i.first<<i.second<<endl;

        for(int i=0;i<n;i++)

        {

            for(int j=0;j<=w[i].size();j++)

            {

                string pre=w[i].substr(0,j);

                string suff=w[i].substr(j);

                if(!pre.empty()&&ispalindrome(pre)&& m.count(suff)&& m[suff]!=i)

                {

                    ans.push\_back({m[suff],i});

                }

                if(ispalindrome(suff)&& m.count(pre)&& m[pre]!=i)

                {

                    ans.push\_back({i,m[pre]});

                }

            }

        }

        return ans;

    }

};

**Problem 3**- [**11. Container With Most Water**](https://leetcode.com/problems/container-with-most-water/)

class Solution {

public:

    int maxArea(vector<int>& height) {

        int n=height.size();

        int i=0;

        int j=n-1;

        int p=0;

        int maxi=INT\_MIN;

        while(i<j)

        {

            int m=min(height[i],height[j]);

            p=m\*abs(j-i);

            maxi=max(maxi,p);

            if(height[i]>height[j])

            {

                j--;

            }

            else if(height[i]<height[j])

            {

                i++;

            }

            else

            {

                i++;

                j--;

            }

        }

        return maxi;

    }

};