CRIBA

```
#include <stdio.h>
#include <vector>
#include <iostream>
using namespace std;

#define vi vector<int>
#define lli long long int

lli n = 100000;
vi criba(n, true);
vector<int> makeCriba(){
   criba[0] = criba[1] = false;
   for(int i = 2; i*i<n; i++){
      if(!criba[i]) continue;
      for(int j = i*i; j<n; j+=i){
         criba[j] = false;
      }
   }
   vector<int> primes;
   for(int i = 2; i<n; i++){
      if(criba[i]){
         primes.push_back(i);
      }
   }
   return primes;
}
```

CRIBA,SET

```
#include <stdio.h>
#include <stdlib.h>
#include <string>
#include <queue>
#include <map>
#include <vector>
#include <iostream>
#include <cmath>
#include <set>
using namespace std;
set<int> midiv;
vector<int> makeCriba(long long int n){
   vector<bool> criba(n, true);
   criba[0] = criba[1] = false;
   for(int i = 2; i*i<n; i++){
      if(!criba[i]) continue;
      for(int j = i*i; j<n; j+=i){
         criba[j] = false;
      }
   }
   vector<int> primes;
   for(int i = 2; i<n; i++){
      if(criba[i]){
         primes.push_back(i);
         midiv.insert(i);
      }
   }

   map<int,int> mymap;
   vector<int> res;
```

```
    //cout<<"TAMANIO"<<primes.size()<<endl;
    for(int j = 0; j< primes.size(); j++){
        for(int i = 1; i<= n; i++){
            if(midiv.count(i)) continue;
            if(i%primes[j] == 0){
                if(mymap.find(i) == mymap.end()) mymap[i] = 1;
                else mymap[i] = mymap[i]+1;
            }
            //cout<<"MAP"<<mymap[i]<<endl;
            if(j==primes.size()-1 && mymap[i] == 2) res.push_back(i);
        }

    }

    cout<<res.size();
    return primes;
}
int main(){
    long long int n;
    cin>>n;
    makeCriba(n);



}
```

TRIE

```
#include <map>
#include <vector>
#include <iostream>
using namespace std;

#define lli long long int

struct Node{
   map<char, lli> next;
   bool end = false;

};

vector<Node> trie;
lli curNode = 0;

lli newNode(){
   trie.push_back(Node());
   return curNode++;
}

void add(string& s){
   lli pt = 0;
   for(lli i = 0; i < s.size(); i++){
      if(not trie[pt].next.count(s[i])){
         trie[pt].next[s[i]] = newNode();
      }
      pt = trie[pt].next[s[i]];
   }
   trie[pt].end = true;
```

```
}

void remove(string& s){
   lli pt = 0;
   for(lli i = 0; i < trie.size(); i++){
      pt = trie[pt].next[s[i]];
   }
   trie[pt].end = false;
}

void DFS(lli pt = 0, string s = ""){
   if(trie[pt].end){
      cout<<s<<endl;
   }
   for(auto i: trie[pt].next){
      s.push_back(i.first);
      DFS(i.second, s);
      s.pop_back();
   }
}
```

DFS GRAFO

```
#include <stdio.h>
#include <stdlib.h>
#include <string>
#include <queue>
#include <map>
#include <vector>
#include <iostream>
using namespace std;

int main(){
   long long int n, m, L, ini, fin;
   long long int pos;
   scanf("%llu %llu %llu %llu %llu", &n, &m, &L, &ini, &fin);

   queue<long long int> myqueue;
   queue<long long int> level;

   // matrix hash
   //map<int, vector<vector<int> > > mymap;
   map<long long int, vector<long long int>> weight;
   map<long long int, vector<long long int>> node;
   vector<vector<long long int>>mat;

   //cout<<L<<endl;
   //int mat[n][2];

   for(int i = 0; i<m; i++){
      // ini fin weight
      vector<long long int>temp;
      long long int a,b,c;
```

```
    scanf("%lli %lli %lli", &a, &b, &c);
    temp.push_back(a);
    temp.push_back(b);
    temp.push_back(c);
    if(c == 0){
        pos = i;
    }
    node[a].push_back(b);
    weight[a].push_back(c);
    mat.push_back(temp);
}
myqueue.push(ini);
level.push(0);
while(!myqueue.empty()){

    int curr = myqueue.back();
    int l = level.back();
    printf("%llu\n",curr);
    if(curr == fin){

        printf("YES\n");
        mat[pos][2] = L-l;
        // print matrix
        for(int i = 0; i<m; i++){
            printf("%llu %llu %llu\n", mat[i][0], mat[i][1], mat[i][2]);
        }
        return 0;
      }
      for(int k = 0; k<node[curr].size(); k++){
        if(l+weight[curr][k] > L){
            continue;
```

```
            }
            else{
               myqueue.push(node[curr][k]);
               level.push(l+weight[curr][k]);
            }

         }

      if(node.find(curr) != node.end()){
         cout<<"entre!"<<endl;



      }

   }
   printf("NO\n");
}
```

DP1

```
#include <vector>
#include <iostream>
#include <string>
#include <math.h>
#include <cmath>
using namespace std;

int main(){
   int n,temp;
   cin>>n;
   vector<int> vecdp(n+1,9999);
   vector<int> entrada(n,0);
   //entrada[0] = 0;
   for(int i = 0; i < n;i++){
      cin>>temp;
      entrada[i]=temp;
      //10 30 40 20
      // 0 inf inf inf inf
      //
   }
   vecdp[0] = 0;
   for(int i = 0; i < n ;i++){
      if(i+1<n) vecdp[i+1] = min(vecdp[i+1],(abs(entrada[i]-entrada[i+1])+vecdp[i]));
      if(i+2<n) vecdp[i+2] = min(vecdp[i+2],(abs(entrada[i]-entrada[i+2])+vecdp[i]));
      //cout<<vecdp[i+1]<<endl;
      //cout<<vecdp[i+2]<<endl;
   }
   cout<<vecdp[n-1];

}
```

DP2

```
#include <string>
#include <math.h>
#include <cmath>
using namespace std;

int main(){
   int n,numPiedras,menos,temp;
   menos = 10000;
   cin>>n;
   cin>>numPiedras;
   vector<int> vecdp(n+1,9999);
   vector<int> entrada(numPiedras,0);
   for(int i = 0; i < numPiedras;i++){
      cin>>temp;
      if(temp<menos) menos = temp;
      entrada[i]=temp;
   }
   vecdp[0] = 0;
   for(int i = 0; i < n ;i++){
      for(int j = 1; j<=saltos; j++){
         if((i+j)<n) vecdp[i+j] = min(vecdp[i+j],(abs(entrada[i]-entrada[i+j])+vecdp[i]));
         //if(i+2<n) vecdp[i+2] = min(vecdp[i+2],(abs(entrada[i]-entrada[i+2])+vecdp[i]));
         //cout<<vecdp[i+j]<<endl;

      }


      //cout<<vecdp[i+2]<<endl;
   }
   cout<<vecdp[n-1];
```

```
}
```

DP3

```
#include <string>
#include <math.h>
#include <cmath>
using namespace std;

int main(){
   int n,t;
   cin>>n;

   vector<vector<int>> entrada;
   vector<vector<int>> dp;
   for(int i = 0; i < n; i++){
      vector<int> temp;
      vector<int> cero;
      for(int j = 0; j < 3; j++){
         cin>>t;
         temp.push_back(t);
         cero.push_back(0);
      }
      if(i!=0) dp.push_back(cero);
      else dp.push_back(temp);
      entrada.push_back(temp);
   }
   if(n==1){
      int a;
      a = max(entrada[0][0],entrada[0][1]);
      a = max(a,entrada[0][2]);
      cout<<a;
      return 0;
   }
```

```
    for(int i = 1; i < n; i++){
        for(int j = 0; j < 3; j++){
            //A
            if(j == 0){
                dp[i][1] = max(dp[i][1],entrada[i][1]+dp[i-1][0]);
                dp[i][2] = max(dp[i][2],entrada[i][2]+dp[i-1][0]);
            }
            //B
            if(j == 1){
                dp[i][0] = max(dp[i][0],entrada[i][0]+dp[i-1][1]);
                dp[i][2] = max(dp[i][2],entrada[i][2]+dp[i-1][1]);
            }
            //C
            if(j == 2){
                dp[i][0] = max(dp[i][0],entrada[i][0]+dp[i-1][2]);
                dp[i][1] = max(dp[i][1],entrada[i][1]+dp[i-1][2]);
            }
        }

    }
    int a;
    a = max(dp[n-1][0],dp[n-1][1]);
    a = max(a,dp[n-1][2]);
    cout<<a;
    return 0;

}
```

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* addTwoNumbers(ListNode* l1, ListNode* l2) {

       int stay = 0;
       ListNode *res=NULL;
       ListNode **s=&res;

      // ListNode* res=l1;
       //cout <<l1->val<<endl;
       while(l1!= NULL || l2 != NULL ){

          //cout <<l1->val<<endl;
          int temp = 0;
          if(l1 == NULL){
             temp = l2->val;

          }
          else if(l2 == NULL){
             temp = l1->val;
          }
          else{
```

```
    temp = l1->val+l2->val;
}

cout <<temp<<endl;
if(stay != 0){
    temp = temp+stay;
    stay = 0;
}
if(temp >= 10){
    stay = temp/10;
    temp = temp - (stay*10);

}
//ut <<temp<<endl;

//res = res->next;
if(l1 == NULL){
    l2 = l2->next;

}
else if(l2 == NULL){
    l1 = l1->next;
}
else{
    l1 = l1->next;
    l2 = l2->next;

}
```

```
        (*s)= new ListNode(temp);
        s = &((*s)->next);


    }
    if(stay > 0){
       (*s)= new ListNode(stay);
    }
    //ultimo ciclo normal
    /*t temp = l1->val+l2->val;

    if(stay != 0){
       temp = temp+stay;
       stay = 0;
    }
    //cout <<temp<<endl;
    if(temp >= 10){
       stay = temp/10;
       temp = temp - (stay*10);
    }
    //cout <<temp<<endl;
    (*s)= new ListNode(temp);
    s = &((*s)->next);

      */

    return res;

  }
};
```

TREE

```
struct TreeNode {
    int val;
    TreeNode *left;
    TreeNode *right;
    TreeNode() : val(0), left(nullptr), right(nullptr) {}
    TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
    TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 };

class Solution {
public:
   TreeNode* pruneTree(TreeNode* root) {

   }
};
```

FORMULAS

Gauss = n*(n+1)/2

Inverso modular = (a^m % m) = a

(a^m-1 % m) = 1

(a^m-2 % m) = 1

Combinacion sin repeticion = [n x].T = (n!)/x!(n-x)!

Total de palabras con n letras con no dos letras Iguales 26*25 a la n-1