UDvorak's notebook (2017)

Contents

1 algorithms

1.1 HornersRule

```
#include <iostream>
using namespace std;
/* Example
 * given the polynomial f(x) = 2x^3 - 6 x^2 - 2x -1
 * we want to know f(8)
 *    -the traditional form in evaluate it
 *  by the horners method is by syntetic division
 *  8  | X^3 X^2  X^1 X^0
 *     |  2  -6   -2  -1
 *     |      16  80  624
 *     -------------------
 *        2   10  78   623
 * With these we can say that the remainder is 623
 * f(8) = 623
 * Wow a pretty good ALGORITHM
 */
typedef long long ll;
```

```
ll Horner( ll a[], ll n, ll x ){
  ll result = a[n];
  for(ll i=n-1; i >= 0 ; --i)
    result = result * x + a[i];
  return result;
}
int main(){
  ll grade = 3;
            //-1 -2x -6x^2 +2x^3
  ll a[] = {-1,-2,-6,2};
  ll x = 8;
  cout << Horner (a, grade, x);
  return 0;
}
```

1.2 Kadane

```
#include <bits/stdc++.h>
#define forn(i,j,k) for(int i=j; i<k; i++)
using namespace std;
typedef long long ll;
/*
 * Largest Sum Contiguous Subarray
 * Kadane Algorithm
 * Complexity O(n)
 */
inline ll kadane(ll data[8], int size){
  ll m1= data[0];
  ll m2 = data[0];
  forn(i, 1, size){
    m2 = max(data[i], data[i] + m2);
    m1 = max(m1, m2);
  }
  return m1;
}
int main(){
  int size = 8;
  ll data[8] = {-1,2,4,-3,5,2,-5,2};
  ll res = kadane(data, size);
  printf("The max sum that can be done with \n \
    Contiguous elements is: %lld \n", res);
  return 0;
}
```

1.3 PickTheorem

```
#include <stdio.h>
using namespace std;
/*
 *  Pick's theorem is a useful method for determining the area
 *  of any polygon whose vertices are points on a lattice,
 *  a regularly spaced array of points.
 */
/*
 * b boundary point : a lattice point on the polygon including
     vertices
 * i interior point : a lattice points on the polygon's interior
     region
 */
double area_poligon(double b, double i){
  return (b/2) + i -1;
}
int main(){
  printf("%f",area_poligon(5,5));
  return 0;
}
```

1.4 Zalgorithm

```
#include <bits/stdc++.h>
#define endl '\n'
#define MAX 1000001
#define pb push_back
using namespace std;
typedef long long lld;
int z[MAX]; //zarray
vector<int> matches;

//Complexity: O(N + M)

inline void zAlgorithm(string s, int m){
  int len = s.length();
  int l = 0, r = 0;
  for (int i = 1 ;i < len; i++){
    if (i > r) {
      l = r = i;
      while (r < len && s[r-l] == s[r]) r++;
      z[i] = r-l;
      r--;
    }else {
      int k = i - l;
      if (z[k] < r-i+1) z[i] = z[k];
      else {
        l = i;
        while (r < len && s[r-l] == s[r]) r++;
        z[i] = r - l;
        r--;
      }
    }
    if (z[i] == m) matches.pb(i - m - 1);
  }
}

int main() {
  string haystack = "abcabc", needle = "abc";
  int n = haystack.size(), m = needle.size();
  zAlgorithm(needle + "#" + haystack, m);
  cout << ("locations where start to match. \n");
  for (int i=0;i<matches.size();i++) cout << matches[i] << " ";
  cout << endl <<"Number of matches: " << matches.size() << endl;
  return 0;
}
```

2 data-structures

2.1 Fenwick

```
#include<bits/stdc++.h>
#define endl '\n'
using namespace std;
typedef long long ll;
typedef vector < ll> vll;
struct fw {
  int n; vll data;
  fw(int _n) : n(_n), data(vll(_n)) { }
  void update(int at, ll by) {
   while(at < n) {
      data[at] += by;
      at |= at + 1;
    }
  }
  void update_range( int l, int r, ll by){
    update(l, by);
    update(r+1, -by);
  }
  ll query(int at) {
   ll res = 0LL;
   while(at >= 0) {
     res += data[at];
     at = (at & (at + 1)) - 1;
   }
   return res;
  }
};

int main(){
#ifdef LOCAL
  freopen("in","r", stdin);
#endif
  ios::sync_with_stdio(0);cin.tie(0);
  int n, q ,a, b;
  char op;
  cin >> n >> q;
  fw *fen  = new fw(n+1);
  for( int i = 0; i < q ; i++){
    cin >> op;
    if( '+' == op){
      cin >> a >> b;
      fen->update(a,b);
    }else{
      cin >>a; a--;
      cout <<  fen->query(a) << endl;
    }
  }

}
```

2.2 Kmp

```
#include<bits/stdc++.h>
#define debug(x) cout <<#x << " = " << x << endl
#define rep(i, a, b) for( __typeof(a) i = a; i < b ; i++)

using namespace std;

int* compute(const string &t) {
  int m = t.size();
  int *p = new int[m];
  p[0]= 0;
  rep( i , 1 , m){
    p[i] = p[ i - 1 ];
    while( p[i] > 0 && t[i] != t[ p[i] ] ){
      p[i] = p[ p[i] -1 ];
    }
    if( t[i] == t[ p[i] ] ) p[i]++;
  }
 return p;
}


int match( const string &ne, const string &ha ){
  debug( ne ); debug( ha);
  int m = ne.size(),   n = ha.size();
  int *p = compute( ne  );
  int s = 0;
  rep( i, 0, n){
    while( s > 0 && ha[ i ] != ne[  s ] )
      s = p[ s - 1];
    if( ha[i] == ne[s] ) s++;
    if( s == m) return i - m + 1;
  }
  delete[] p;
  return -1;
}

int main(){
#ifdef LOCAL
  freopen("in", "r" , stdin);
#endif

  string needle = "abcaby";
  string haystack  = "abcabcabyid";

  cout << match (  needle , haystack ) <<endl;


  return 0;
}
```

2.3 SegmentTree

```
#include <bits/stdc++.h>
using namespace std;
const int N = 100000;
int n;
int tree[2 * N];
void build( int arr[]){
  // insert leaf nodes in tree
  for (int i=0; i<n; i++) tree[n+i] = arr[i];
  // build the tree by calculating parents
  for (int i = n - 1; i > 0; --i)
  tree[i] = tree[i<<1] + tree[i<<1 | 1];
}
// function to update a tree node
void updateTreeNode(int p, int value){
  // set value at position p
  tree[p+n] = value;
  p = p+n;
  // move upward and update parents
  for (int i=p; i > 1; i >>= 1)
  tree[i>>1] = tree[i] + tree[i^1];
}
// function to get sum on interval [l, r)
int query(int l, int r)  {
  int res = 0;
  // loop to find the sum in the range
  for (l += n, r += n; l < r; l >>= 1, r >>= 1) {
    if (l&1)  res += tree[l++];
    if (r&1) res += tree[--r];
  }
  return res;
}
int main(){
  int a[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12};
  // n is global
  n = sizeof(a)/sizeof(a[0]);
  // build tree
  build(a);
  // print the sum in range(1,2) index-based
```

```
    cout << query(0, 3)<<endl;
    // modify element at 2nd index
    updateTreeNode(2, 1);
    // print the sum in range(1,2) index-based
    cout << query(1, 3)<<endl;
    return 0;
}
```

2.4 Trie

```
#include <bits/stdc++.h>
using namespace std;
/*
 * Struct for a trie
 */
struct node {
  node * son[26];
  bool is_end;
  int num_times;
  node(){
   memset(son, 0, sizeof(son));
   is_end =false;
   num_times =0;
  }
};
/*
 * insert a word in the trie
 */
void insert(node* nd, char *s){
  if(*s){
   int pos = *s - 'a';
   if(!nd->son[pos]) nd->son[pos]=new node();
   insert(nd->son[pos], s+1);
  }else{
   nd->is_end = true;
  }
}
/*
 * Check if the word is in the trie
 */
int contains(node *nd, char *s){
  if(*s){
   int pos = *s - 'a';
   if(!nd->son[pos]) return false;
   return contains(nd->son[pos], s+1);
   }else{
   return nd->is_end;
  }
}
int main(){
  node * trie = new node();
  string  a = "word";
  char *cstr = new char[a.length() + 1];
  strcpy(cstr, a.c_str());
  insert (trie, cstr);
  string b = "banani";
  strcpy(cstr, b.c_str());
  insert (trie, cstr);
  if (contains(trie, cstr)){
    cout << "ohh holly xx." << endl;
  }else{
    cout << "mother ..." << endl;
  }
  return 0;
}
```

2.5 UnionFind

```
#include <bits/stdc++.h>
using namespace std;
typedef vector<int> vi;
struct union_find {
  vi p;
  //initialize all elements with -1
  union_find(int n) : p(n, -1) { }
  int find(int x) {
    return p[x] < 0 ? x : p[x] = find(p[x]);
  }
  bool unite(int x, int y) {
    int xp = find(x), yp = find(y);
    if (xp == yp) return false;
    if (p[xp] > p[yp]) swap(xp,yp);
    p[xp] += p[yp], p[yp] = xp; //add -1 if merge
    return true;
  }
  int size(int x) {
     return -p[find(x)];
  }
};

int main() {
    union_find uf(10);
    uf.unite(0, 2);
    cout << uf.find(0) << endl;
    cout << uf.find(2) << endl;
    assert(uf.find(0) == uf.find(2));
    assert(uf.find(0) != uf.find(1));
    return 0;
}
```

3 geometry

3.1 CenterCircle

```
#include <bits/stdc++.h>
using namespace std;
const double PI = acos(-1);
#define show(x) cout << #x << " = " << x << endl;
struct pt {
  double x;
  double y;
  pt (){}
  pt (double _x, double _y){
    x = _x;
    y = _y;
  }
};
inline pt getCenter(pt p1, pt p2, pt p3){
  pt center;
  float m1 = (p2.y - p1.y)/(p2.x - p1.x);
  float m2 = (p3.y - p2.y)/(p3.x - p2.x);
  center.x = ( m1 * m2 * (p1.y - p3.y) + m2 * ( p1.x + p2.x)
                 - m1 * (p2.x + p3.x) )
               / (2 * (m2 - m1) );
  center.y = -1 * (center.x - (p1.x + p2.x) / 2) / m1 +  (p1.y +
      p2.y) / 2;
  return center;
}
```

```
int main(){
  pt p1(1,1), p2(2,4), p3(5,3);
  pt res = getCenter(p1, p2, p3);
  show(res.x)
  show(res.y)
  return 0;
}
```

3.2 PolygonArea

```
#include <bits/stdc++.h>
#define f first
#define s second
#define mp make_pair
#define pb push_back
using namespace std;
typedef long double ld;
typedef pair <ld, ld> point;
typedef vector < point > polygon;
inline  point diff(point o, point d){
  return mp(d.f-o.f, d.s - o.s) ;
}
inline ld crossProduct(point o, point d){
  ld cross  = (o.f * d.s)  - ( o.s * d.f);
  return cross > 0 ? cross :  cross * -1;
}
inline ld area(polygon p){
  int num_points = p.size();
  ld area = 0;
  for (int i = 1; i < num_points -1 ; i++){
    point l1 = diff(p[0],p[i]);
    point l2 = diff(p[0],p[i+1]);
    area += crossProduct(l1,l2);
  }
  return abs(area/2.0);
}
int main(){
  polygon p;
  p.pb(mp(1,0)); p.pb(mp(2,1));
  p.pb(mp(1,2)); p.pb(mp(0,1));
  cout << area(p);
  return 0;
}
```

3.3 RayCasting

```
#include <bits/stdc++.h>
#define pb push_back
#define mp make_pair
using namespace std;
/*
 * This program implements the ray casting algorithm to check
 * if a point is inside or outside of a simple polygon
 */
typedef double ld;
struct point {
  ld x, y;
  point(){}
  point(ld x, ld y){
    this->x = x;
    this->y = y;
  }
};
struct vert {
    point o,d;
};
typedef vector < point > polygon;

inline ld cross(point o, point d){
  return(o.x * d.y)  - ( o.y * d.x); }
inline ld dot(point o, point d){
  return (o.x * d.x)  + ( o.y * d.y); }
inline point diff(point o, point d){
  return {d.x-o.x, d.y - o.y} ;}
inline ld dist(point o, point d){
  return  sqrt(dot(diff(o,d) , diff(o,d))); }

inline  bool segments_parallel(point a, point b, point c){
    return abs(cross(diff(c,a),diff(b,a)))  == 0;
}
inline bool point_on_segment(polygon v,  point c){
  int cant = v.size();
  for (int i=0;i<cant;i++){
    if (dist(v[i],c)==0) return true;
    if (dist(v[(i+1)%cant],c)==0) return true;
    if(segments_parallel(v[i], v[(i+1)%cant], c) &&
        dot(diff(c,v[i]), diff(c,v[(i+1)%cant])) < 0) {
          return true;
    }
  }
  return false;
}

/* Ray Casting algorithm
 * true inside
 * false outside
 */
bool point_in_polygon(point p, polygon a){
  bool inside = false;
  int cant = a.size();
  for (int i=0;i<cant;i++){
    int j = (i+1) % cant;
    point aux = a[i];
    point nxt = a[j];
    bool cond1 = (p.y < aux.y != p.y < nxt.y);
    bool cond2 = (p.x < aux.x + (nxt.x - aux.x) * (p.y - aux.y) /
        (nxt.y - aux.y));
    if ( cond1 && cond2 ){
      inside = !inside;
    }
  }
  return inside;
}
inline void test_point(polygon v, point pun){
  if(point_on_segment(v,pun)){
      cout << "on"<<endl;
  }else if (point_in_polygon(pun, v)){
      cout << "in"<<endl;
  }else{
      cout <<"out"<<endl;
  }
}
int main(){
    polygon p;
    p.pb(point(1,0)); p.pb(point(2,1));
    p.pb(point(1,2)); p.pb(point(0,1));
    test_point(p, point(0,0));
    test_point(p, point(1,1));
    test_point(p, point(1.5,0.5));
    return 0;
  }
```

3.4 Struct

```
#include <bits/stdc++.h>
#define INF 1e9
#define EPS 1e-9
#define PI acos(-1.0)
#define debug(x) cout << #x << " " << x << endl;

using namespace std;

struct point {
  double x, y;
  point() {}
  point(double _x, double _y){
    x = _x;
    y = _y;
  }
  point operator + (point p) const {
    return point(p.x + x, p.y +y);  }
  point operator - (point p) const {
    return point(x - p.x, y -p.y);  }
  point operator *(double d) const {
    return point(x*d, y*d); }
  bool operator ==(point p) const {
    return p.x == x && p.y==y; }
  double dot(point p) {
    return x*p.x + y*p.y;};
  double cross2(point p) {
    return x*p.y - p.x*y;};
  double mag () {
    return sqrt(x*x + y*y);};
  double norm() {
    return x*x + y*y;};
  double dist(point p2){
    return hypot(x - p2.x, y - p2.y); };
  void show(){ printf("x= %lf, y=%lf\n", x, y);}
};

struct line {
  point o;//origin
  point d;//destiny
  double m;
  line (){}
  line( point _o, point _d){
    o = _o;
    d = _d;
  }
  double slope(){
    if (o.x != d.x){
      m = (double)(d.y - o.y) / (double)(d.x - o.x);
      return m;
    }
    m = INF;
    return INF;
  }

};
double cross(point &o, point &a, point &b) {
    return (a.x - o.x)*(b.y - o.y) - (a.y - o.y)*(b.x - o.x);
}
bool areParallel(line l1, line l2) {
  return fabs(l1.slope()-l2.slope())<EPS ;
}
double distToLine(point p, line l1) {
  // formula: c = a + u * ab
  point a = l1.o, b = l1.d, c;
  point ap = p-a, ab = b-a;
  double u = ap.dot(ab) / ab.norm();
  c = a + ab*u;
  return c.dist(p);
}

double distToSegment(point p, line l1) {
  point a = l1.o, b = l1.d, c;
  point ap = p-a, ab = b-a;
  double u = ap.dot(ab) / ab.norm();
  if (u < 0.0) {
    c = point(a.x, a.y);  // closer to a
    return p.dist(a);
  }
  if (u > 1.0) {
    c = point(b.x, b.y);  // closer to b
    return p.dist(b);
  }
  return distToLine(p, line(a, b));
}

int main(){
  point a(0,4);
  point b(5,0);
  point c(7,0);
  a.show();
  b.show();
  c.show();
  line l1(a,b), l2(a,c);
  cout << "m1= "<< l1.slope() << endl;
  cout << "m2= "<< l2.slope() << endl;
  cout << "parallel l1 || l2? = " << (areParallel(l1, l2)?"true":
     "false") << endl;
  cout << "dist from point to line= " << distToLine(c, l1) << endl
     ;
  cout << "dist from point to segment= " << distToSegment(c, l1)
     << endl;
  return 0;
}
```

4 graph

4.1 BFS

```
#include <bits/stdc++.h>
#define pb push_back
using namespace std;
typedef vector < int > vi;
vi dis;
vector < vi > graph;
void show_distances(){
  for( int i = 0; i< dis.size(); i++){
    cout << i << " : " << dis[i] << "\n";
  }
}
void bfs(int origin){
  queue < int > q;
  dis[origin] = 0;
  q.push(origin);
  while( q.size() > 0){
    int front = q.front(); q.pop();
    for(int son: graph[front]){
      if(dis[son] == -1){
        dis[son] = dis[front] +1;
        q.push(son);
      }
    }
  }
}

int main(){
  int num_nodes = 5;
  dis.assign(num_nodes, -1);
  graph.resize(num_nodes);
  graph[0].pb(1);
  graph[0].pb(2);
  graph[0].pb(3);
  graph[1].pb(4);
  bfs(0);
  show_distances();
  return 0;
}
```

4.2 BFSMaze

```
#include <bits/stdc++.h>
#define pb push_back
#define ROWS 10
#define endl '\n'
#define COLS 10
#define mp make_pair
using namespace std;
struct point {
  int x, y;
  point() {}
  point(int _x, int _y){
    x = _x;
    y = _y;
  }
  point operator + (point p) const {
    return point(p.x + x, p.y +y);  }
  point operator - (point p) const {
    return point(x - p.x, y -p.y);  }
  bool operator == (point p) const {
    return x==p.x && y ==p.y; }
  bool operator <(point p) const {
    return p.x < x || p.y<y; } // set vis
  void show(){ printf("x= %d, y=%d\n", x, y);}
};

vector < point > dir = {point(1,0), point(-1,0), point(0,1), point
    (0,-1)};;
int r , c, salx, saly;

char mat[ROWS][COLS];
set < pair < int , int > > vis;
point sal;

bool inBoundaries(point &p){
  if(p.x >= 0 && p.y >=0 && p.x < r &&  p.y<c) return true;
  return false;
}

bool sol(point in){
  if (in == sal) return true;
  queue < point > q;
  q.push(in);
  while (!q.empty()){
    point actual = q.front();
    vis.insert(mp(in.x, in.y));
    q.pop();
    actual.show();
    for (int i= 0; i < 4; ++i){
      point p = actual + dir[i];
      if( inBoundaries(p) && (vis.count(mp(p.x, p.y))==0)){
        q.push(p);
        vis.insert(mp(p.x, p.y));
        cout  << '\t';
        p.show();
        if (p == sal) return true;
      }
    }
  }
  return false;
}


int main(){
  #ifdef LOCAL
   freopen("in.c", "r", stdin);
  #endif
  vis.clear();
  //rows and cols
  cin >> r  >> c;
  for ( int row =0 ; row < r ; row++){
    for (int col = 0; col < c; col++){
      cin >> mat[row][col];
    }
  }
  sal.x = sal.y =2;
  point ini(1,1);
  sol(ini);
  return 0;
}
```

4.3 DFS

```
#include <bits/stdc++.h>
#define pb push_back
#define NUM_NODES 20
using namespace std;
vector < int > g[NUM_NODES];
int vis[NUM_NODES];
enum {WHITE, GRAY, BLACK};
void dfs(int o){
  vis [o] = GRAY; //semi-visited
  for (int i = 0; i < g[o].size(); i++){
    int v = g[o][i];
    if (vis[v] == GRAY)
      cout << "Cycle to " << o <<  endl;
    else if (vis[v] == WHITE) dfs(v);
  }
  cout << o << endl;
  vis[o] = BLACK; //visited;
}
int main(){
  g[0].pb(1);  g[0].pb(2);
  g[0].pb(3);  g[1].pb(4);
  g[1].pb(5);  g[2].pb(6);
  g[3].pb(7);  g[4].pb(0);
  g[6].pb(0);
  dfs(0);
  return 0;
}
```

4.4 Dijkstra

```
#include <bits/stdc++.h>
#define V 9
int minDis(int dist[], bool is_set[]){
  int min = INT_MAX, min_index;
  for (int v = 0; v < V; v++){
    if (is_set[v] == false && dist[v] <= min){
      min = dist[v], min_index = v;
    }
  }
  return min_index;
}

inline  void dijkstra(int graph[V][V], int src){
  int dist[V];
  bool is_set[V];
  for (int i = 0; i < V; i++){
    dist[i] = INT_MAX, is_set[i] = false;
  }
  dist[src] = 0;
  for (int count = 0; count < V-1; count++){
   int u = minDis(dist, is_set);
   is_set[u] = true;
   for (int v = 0; v < V; v++){
     if (!is_set[v] && graph[u][v]
             && dist[u] != INT_MAX
             && dist[u]+graph[u][v] < dist[v])
         dist[v] = dist[u] + graph[u][v];
   }
  }
  for( int i= 0; i < V; i++)
    cout << i << " " << dist[i] <<endl;
}
int main(){
   int graph[V][V] =
    {{0, 4, 0, 0, 0, 0, 0, 8, 0},
     {4, 0, 8, 0, 0, 0, 0, 11, 0},
     {0, 8, 0, 7, 0, 4, 0, 0, 2},
     {0, 0, 7, 0, 9, 14, 0, 0, 0},
     {0, 0, 0, 9, 0, 10, 0, 0, 0},
     {0, 0, 4, 14, 10, 0, 2, 0, 0},
     {0, 0, 0, 0, 0, 2, 0, 1, 6},
     {8, 11, 0, 0, 0, 0, 1, 0, 7},
     {0, 0, 2, 0, 0, 0, 6, 7, 0}
    };
    //distances from all points to 1
    dijkstra(graph, 1);
    return 0;
}
```

4.5 DijkstraHeap

```
#include <bits/stdc++.h>
#define pb push_back
using namespace std;
#define forn(i,a) for (int i=0; i<a ; i++)
#define INF 2e7
struct edge{
        int to, weight;
        edge(){}
        edge(int _to, int _weight){
                to = _to;
                weight = _weight;
        }
        bool operator < (edge e) const {
                return weight > e.weight;
        }
};
typedef vector < edge > ve;
typedef vector < ve > vve;
typedef vector < int > vi;
typedef priority_queue< edge> pq;
inline void dijkstra(vve &adj, int src, int num_nodes){
  vi dist = vi(num_nodes+1,INF);
        pq  q;
  //by default
  q.push(edge(src,0));
  dist[src] = 0;
  //apply bfs
  while(!q.empty()){
    edge top = q.top();
    q.pop();
    int u = top.to;
    for(int i=0;i<adj[u].size();i++){
      int v = adj[u][i].to;
      if(dist[u] + adj[u][i].weight < dist[v]){
        dist[v] = dist[u] + adj[u][i].weight;
        q.push(edge(v,dist[v]));
      }
    }
  }
  //Show results of distances
  cout << "Distancias desde el origen ";
  cout << src << endl;
  forn(i, num_nodes){
    cout <<"Costo al nodo: " << i;
    cout << " ="<< dist[i] << endl;
  }
}

int main(){
        int nodes =5;
  vve adj(nodes);
        //from             to - weight
        adj[0].pb(edge(1, 6));
        adj[0].pb(edge(2, 2));
        adj[1].pb(edge(3, 5));
        adj[1].pb(edge(4, 7));
  int src = 1;
  dijkstra(adj, src, nodes);
  return 0;
}
```

4.6 FloydWarshal

```
#include<iostream>
#include<stdio.h>
using namespace std;
/*
 * Floyd-Warshall gives us the shortest paths
 * from all sources to all target nodes.
 */
#define V 4  //number of vertex
#define INF 9999999

void show(int dist[][V]){
  printf ("shortest distances \n");
  for (int i = 0; i < V; i++){
      for (int j = 0; j < V; j++){
        if (dist[i][j] == INF)
          printf("%7s", "INF");
        else
          printf ("%7d", dist[i][j]);
        }
         printf("\n");
    }
  }
  void floyd (int graph[][V]){
    int dist[V][V], i, j, k;
    for (i = 0; i < V; i++)
      for (j = 0; j < V; j++)
        dist[i][j] = graph[i][j];
    for (k = 0; k < V; k++){
      for (i = 0; i < V; i++){
        for (j = 0; j < V; j++){
          if (dist[i][k] + dist[k][j] < dist[i][j])
              dist[i][j] = dist[i][k] + dist[k][j];
        }
      }
    }
    show(dist);
  }
  int main(){
      int graph[V][V] =
      { {0,   5,  INF, 10},
        {INF, 0,   3, INF},
        {INF, INF, 0,   1},
        {INF, INF, INF, 0}
      };
      floyd(graph);
      return 0;
  }
```

4.7 RecoveryTree

```
#include <iostream>
using namespace std;
/**Build a binary tree form a inorder and preoder string **/
int preIndex = 0;
struct node {
  char key;
  node *left, *right;
  node(int k) {
    key = k;
    left = NULL;
    right = NULL;
  }
};
int search(string word, int b, int e, char c) {
  for(int i=b; i<=e; i++) {
    if(word[i] == c) return i;
  }
  return -1;
}
//Set preIndex to 0 to build another tree
node* build(string in, string pre, int b, int e) {
  if(b > e) return NULL;
  node *root = new node(pre[preIndex++]);
  if(b == e)return root;
  int inIndex = search(in, b, e, root->key);
  root->left = build(in, pre, b, inIndex - 1);
  root->right = build(in, pre, inIndex + 1, e);
  return root;
}

int main() {
  string pre, in;
  node *tree;
  while(cin >> pre >> in) {
    tree = build(in, pre, 0, pre.size() - 1);
    preIndex = 0;
  }
  return 0;
}
```

4.8 Tsort

```
#include<bits/stdc++.h>
#define debug(x) cout << #x << " = " << x <<endl;
#define PB push_back
using namespace std;

typedef vector < bool > vb;
typedef vector < int > vi;

enum { NV, SV, V};
vb vis;
int N;
vector < vi > G;

void dfs( int src, stack < int > &S ){
  vis[src] = SV;
  debug( src );
  for( int son: G[src]){
    if( vis[ son ] == NV){
        dfs( son, S );
    }
  }
  vis[src] = V;
  S.push( src );
}

void tsort( ){
  stack< int > S;
  vis.resize( N );
  vis.assign( N, NV);
  for( int i = 0; i < N; i++){
    if( vis[i]  == NV){
      dfs( i, S);
    }
  }

  while(!S.empty()){
    cout << S.top() <<endl;
    S.pop();
  }
}

int main(){

  N = 6;
  G.resize(N);
  G[0].PB( 1 );
  G[0].PB( 2 );
  G[0].PB( 3 );
  G[1].PB( 4 );
  G[4].PB( 3 );
  G[5].PB( 2 );
  G[3].PB( 2 );
```

```
    tsort( );

    return 0;
}
```

5 mathematics

5.1 Binomial

```
#include <iostream>
using namespace std;
const int MAXN = 66;
unsigned long long ch[MAXN+5][MAXN+5];
void binomial(int N){
  for (int n = 0; n <= N; ++n)
    ch[n][0] = ch[n][n] = 1;
  for (int n = 1; n <= N; ++n){
    for (int k = 1; k < n; ++k){
      ch[n][k] = ch[n-1][k-1] + ch[n-1][k];
    }
  }
}
int main(){
  binomial(10);
  cout << ch[10][2] << endl;
}
```

5.2 Binomial

```
import math, sys
MAXN = 431
choose = []
for i in range (0, MAXN+5):
  choose.append([0]*(MAXN+5))
def binomial(N):
  for n in range (0, N+1):
    choose[n][0] = choose[n][n] = 1
  for n in range(1, N+1):
    for k in range(1, n):
       choose[n][k] = choose[n-1][k-1] + choose[n-1][k]
if __name__ == "__main__":
  N = 431
  binomial(N)
  n, k = 10, 4
  print(choose[n][k])
```

5.3 ChangeBases

```
#include<bits/stdc++.h>
#define endl '\n'
#define show(x) cout <<#x << " =" <<x <<endl;
using namespace std;
typedef long long ll;
string chars = "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ";

ll to10(ll n , ll b, ll mul){
  if (n ==0 ) return 0;
  return (n % 10)*mul + to10(n / 10, b, mul*1LL*b);
}
string tob(ll n, ll b){
  if (n == 0) return "";
  return  tob(n / b, b) + chars[n % b];
}
/*
 * ob -> origin base
 * db -> destiny base
 */
string changeBase(ll num, ll ob, ll db){
  if (ob == 10) return tob(num, db);
  return tob(to10(num, ob, 1LL), db);
}
int main(){
  cout << changeBase(1000,2,10) <<endl;
}
```

5.4 CoinChange

```
#include <bits/stdc++.h>
#define MAXCOINS (10005)
#define MAXVALUE (105)
using namespace std;
typedef vector < int > vi;
int dp[MAXVALUE][MAXCOINS];
vi coins;
//recursive
int ways(int tg, int n){
  if ( 0 == tg) return 1;
  if ( 0 > tg) return 0;
  if ( n <= 0 && tg >0) return 0;
  return ways(tg, n-1) +
    ways(tg - coins[n -1], n);
}
//by dp
int waysdp(int tg, int n){
  for( int i=0; i< coins.size(); i++) dp[0][i] = 1;
  for(int i = 1 ; i<= tg; i++){
   for (int c = 0; c < n; c++){
      int x =0 , y = 0;
      if(i-coins[c] >= 0) x = dp[i - coins[c]][c];
      if( c >=1) y = dp[i][c-1];
      dp[i][c] = x + y;
          }
      }
  return dp[tg][n-1];
}

int main(){
  coins.insert(coins.end(), {1,3,9,27});
  cout <<  ways(47, coins.size()) <<endl;
  cout << waysdp(47, coins.size()) <<endl;
  return 0;
}
```

5.5 Combination

```
array = [3,2,4,0]
n = len (array)
r = 3

def combination(data, start, index):
  if (index == r):
    print (data)
    return
  for i in range(start, n):
    """  n - i + 1 >= r - index  makes sure that
         including one element  at index will
         make a combination with remaining
         elements at remaining positions
    """
    if (n - i + 1 >= r - index):
      data[index] = array[i]
      combination(data, i+1 , index + 1)

def get_combinations(r):
  combination([0] * r, 0,  0)

if __name__ == "__main__":
  get_combinations(r=3)
```

5.6 CompareDoubles

```
#include <stdio.h>
using namespace std;
const double EPS = 1e-15;
/*
 * Return
 * -1  if x < y
 *  0  if x == y
 *  1  if x > y
 */
int cmp (double x, double y){
    return (x <= y + EPS) ? (x + EPS < y) ? -1 : 0 : 1;
}
int main(){
  double d1 = 0.00000000000212;
  double d2 = 0.00000000000213;
  int res = cmp(d1,d2);
  if (res == 0){
    printf("Equal \n");
  }else if(res == 1){
    printf("Greater\n");
  }else {
    printf("Less \n");
  }
}
```

5.7 Divisors

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef set<ll> si;
/* Get the divisors of a number */
si divisores(ll n) {
  si d;
  ll r = sqrt(n);
  for(ll i = 1; i <= r; i++) {
    if(n % i == 0) {
      d.insert(i);
      d.insert(n / i);
    }
  }
  return d;
}
int main() {
  si divi = divisores(10);
  for(ll el: divi){
        cout << el <<endl;
        }
  printf("\n");
}
```

5.8 Exponentiation

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
// (a^b)%c
ll expo(ll a, ll b, ll c){
  if (b == 0) return 1;
  if (b % 2 == 0) {
    ll temp = expo(a, b/2, c);
    return (temp * temp) % c;
  } else {
    ll temp = expo(a, b-1, c);
    return (temp * a) % c;
  }
}
int main(){
  cout << expo(2, 100, 1025);
  return 0;
}
```

5.9 ExtendedEuclides

```
#include <bits/stdc++.h>

using namespace std;
typedef long long ll;
typedef vector < ll > vl;

vl arr(3);
/*
  returs gcd(a,b) and find the coeficcients of bezout
  such that d = ax + by
  arr[0] gcd
  arr[1] x
  arr[2] y
*/
void extended(ll a, ll b){
  ll y =0;
  ll x =1;
  ll xx =0;
  ll yy =1;
  while(b){
    ll q = a / b;
    ll t = b;
    b = a%b;
    a = t;

    t = xx;
    xx = x-q*xx;
    x = t;

    t = yy;
    yy = y -q*yy;
    y = t;
  }
  arr[0] = a;
  arr[1] = x;
  arr[2] = y;
}
```

```
/*
  ax + by  = c
  mcd(a,b) = d
  ax0 + by0 = d
  c = d * c'

  Bezouts identity
   X = x0 * c' - (b/d) * k
   Y = y0 * c' + (a/d) * k
*/
int main(){
  ll a = 20, b = 50;
  extended(a,b);
  printf("gcd(%lld, %lld) = %lld = %lld * %lld + %lld * %lld\n",
       a, b, arr[0], a,arr[1], b, arr[2]);
  return 0;
}
```

5.10 ExtendEuclidesSample

```
#include<bits/stdc++.h>
#define debug(x) cout << #x << " = "<< x <<endl
using namespace std;
typedef long long ll;
typedef vector <ll > vl;
vl arr(3);
/*
       13250 - Balance Game
*/
//Extended euclidean theorem
void extended(ll a, ll b){
  ll y =0, x =1, xx =0, yy =1;
  while(b){
   ll q = a / b, t = b;
   b = a%b;
   a = t;

   t = xx;
   xx = x-q*xx;
   x = t;

   t = yy;
   yy = y -q*yy;
   y = t;
  }
  arr[0] = a;
  arr[1] = x;
  arr[2] = y;
}

int main(){
  #ifdef LOCAL
   freopen("in.c", "r", stdin);
  #endif
  ll m, n;
  double a,b,c3;

  while(cin >>  m >> n){
    cin >> a >> b >> c3;
    extended(a, b);
    ll mcd = arr[0];
    ll res =0;

    for (ll i=-m; i <= m; i++   ){
      ll val = n - c3*i;
      if(val % mcd != 0)  continue;
      ll x0 = arr[1], y0 = arr[2], d = mcd;
      ll cp =  val / mcd;
      //Compute values range kx
      double kminx = (d*(x0*cp -m))/b, kmaxx =  (d*(x0*cp +m))/b;
      //Compute  values range ky
      double kminy = d*(-m -y0*cp)/a, kmaxy = d*(m -y0*cp)/a;
      //Now the intersection betewen kx and ky
      ll kmin = max(ceil(kminx), ceil(kminy));
      ll kmax = min(floor(kmaxx), floor(kmaxy));
      //Add the number of values in the range
      if(kmin<=kmax) res += abs(kmax-kmin)+1;
      //Debugging purposes
      /*
        printf("Cofficient = %d \n", val);
        printf("\tx = %d - %.0fk\n",x0*cp, b/d);
        printf("\ty = %d + %.0fk\n",y0*cp, a/d);
        printf("\tRanges: \n\t\tkx = [%.2f, %.2f]", kminx, kmaxx);
        printf(", ky = [%.2f, %.2f]", kminy, kmaxy);
        printf(", intersection = [%d, %d]\n", kmin, kmax);
        for(int k=kmin; k <=kmax; k++){
          //values of x and y that sattisfy the equation
          int x_ = x0*cp - (b*k)/d ;
          int y_ = y0*cp + (k*a)/d;
          printf("\tx = %d, y = %d \n",x_ , y_);
        }
        printf("\n\n");
      */
    }
    cout << res <<endl;
  }
  return 0;
}
```

5.11 Factorize

```
#include <bits/stdc++.h>
#define pb push_back
#define show(x) cout << #x << " = " << x << endl;
using namespace std;
const int MAXN = 1000000;
bool sieve[MAXN + 5];
typedef long long ll;
vector <ll> pri; //primes

void build_sieve(){
  memset(sieve, false, sizeof(sieve));
  sieve[0] = sieve[1] = true;
  for (ll i = 2LL; i * i <= MAXN; i ++){
    if (!sieve[i]){
      for (ll j = i * i; j <= MAXN; j += i){
        sieve[j] = true;
      }
    }
  }
  for (ll i = 2; i <= MAXN; ++i){
    if (!sieve[i]) pri.pb(i);
  }
}
//before call this call build_sieve
vector <ll> fact(long long a){
  vector <ll> ans;
  ll b = a;
  for (int i = 0; 1LL * pri[i] * pri[i] <= a; ++i){
    int p = pri[i];
    while (b % p == 0){
      ans.push_back(p);
      b /= p;
      }
    }
    if (b != 1) ans.push_back(b);
    return ans;
  }
  int main(){
    build_sieve();
    ll num_to_fact= 128234234LL;
    vector < ll > vll = fact(num_to_fact);
    for (int x=0; x< vll.size(); x++){
      cout << vll[x] << " ";
    }
    cout << endl;
  }
```

5.12 FastPow

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
ll modular_pow(ll  base, int exponent, ll  modulus){
 ll result = 1;
 while (exponent > 0){
 /* if y is odd, multiply base with result */
               if (exponent & 1)
                       result = (result * base) % modulus;
               /* exponent = exponent/2 */
               exponent = exponent >> 1;
               /* base = base * base */
               base = (base * base) % modulus;
        }
        return result;
}
int main(){
        ll exp = 1023;
        cout << modular_pow (2, exp, 999) << endl;
}
```

5.13 GcdLcm

```
#include<cstdio>
using namespace std;
typedef long long ll;
ll mod( ll a, ll b){
  return (b + (a %b )) %b;
}
ll gcd ( ll a, ll b){
  if (b == 0 ) return a;
  return gcd( b, mod( a , b) );
}
ll lcm(ll n1, ll n2){
    return (n1 *1LL* n2) / gcd(n1,n2);
}
int main(){
    ll n1=2366, n2=273;
    printf("gcd(%ld, %ld) = %ld\n",
            n1, n2, gcd(n1,n2));
    return 0;
}
```

5.14 IsFibo

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
bool isPerfectSquare(long long x){
  ll s = sqrt(x);
  return (s*1LL*s == x);
}
bool isFibonacci(int n){
  // n is Fibinacci if one of 5*n*n + 4 or 5*n*n - 4 or both
  // is a perferct square, this is deduced of the discriminant
  //of binnets formule
  return isPerfectSquare(5*n*1LL*n + 4) || isPerfectSquare(5*1LL*n
      *n - 4);
}

// A utility function to test above functions
int main() {
  for (int i = 1; i <= 10; i++)
     isFibonacci(i)? cout << i << " is a Fibonacci Number \n":
                     cout << i << " is a not Fibonacci Number \n"
                         ;
  return 0;
}
```

5.15 IsPrime

```
import java.math.BigInteger;
import java.util.Scanner;
public class prime {
  public static void main(String[] args) {
   BigInteger a = new BigInteger("1299827");
   //User miller rabin & Lucas Lehmer
   boolean res = a.isProbablePrime(10);
   System.out.println(res? "It's prime":"It's not prime");
 }
}
```

5.16 knapsack

```
#include<bits/stdc++.h>
#define MAX (int) 1e3
using namespace std;
int v[5] = {60, 100, 120, 30, 5};
int w[5] = {10, 20, 30, 30, 5};

int memo[MAX][MAX];

int knapsack( int n , int W){
  if( n  == 0 || W == 0 ) return 0;
  int &ans = memo[n][W];
  if( ans != -1) return ans;
  if( w[n] > W ) { //not include too heavy
    ans =  knapsack(n-1, W);
  }else{
    //Include
    int a1 = v[n]+ knapsack( n-1, W-w[n] );
    //Not include
    int a2 = knapsack( n -1, W);
    ans =  max(a1, a2);
  }
  return ans;
}

int main() {
    for( int i = 0; i < MAX; i++)
      for( int j = 0; j < MAX; j++)
        memo[i][j] =  -1;

    cout << knapsack ( 5, 50) << endl;
    return 0;
  }
```

5.17 Knapsack

```
#include <bits/stdc++.h>
using namespace std;

typedef vector < int > vi;
typedef vector < vi  > vii;
// w[i] = peso del objeto i (i comienza en 1)
vi w;
vi v;
// dp[i][j] m xima ganancia si se toman un subconjunto de los
// objetos 1 .. i y se tiene una capacidad de j
int ** dp;

int knapsack(int n, int W){
  for (int j = 0; j <= W; ++j) dp[0][j] = 0;
  for (int i = 1; i <= n; ++i){
    for (int j = 0; j <= W; ++j){
      dp[i][j] = dp[i-1][j];
      if (j - w[i] >= 0){
        dp[i][j] = max(dp[i][j],
          dp[i-1][j-w[i]] + v[i]);
      }
    }
  }
  return dp[n][W];
}

int main(){
  int numObjects = 10;
  int maxCapacity = 100;
  dp = new int*[numObjects];
  for (int i =0; i < maxCapacity; i++)
    dp[i]= new int[maxCapacity];

  w.resize(numObjects);
  v.resize(numObjects);
  int cont = numObjects;
  for( int i = 1; i < numObjects; i++){
    w[i] = i;
    v[i] = cont--;
  }
  cout << knapsack(10, 100);
}
```

5.18 MatrixFibo

```
#include <bits/stdc++.h>
using namespace std;

const int MAX = 1000;
int f[MAX] = {0};
// Returns n'th fuibonacci number using table f[]
int fib(int n){
  // Base cases
  if (n == 0) return 0;
  if (n == 1 || n == 2) return (f[n] = 1);
  // If fib(n) is already computed
  if (f[n])   return f[n];
  int k = (n & 1)? (n+1)/2 : n/2;
  // Applyting above formula [Note value n&1 is 1
  // if n is odd, else 0.
  f[n] = (n & 1)? (fib(k)*fib(k) + fib(k-1)*fib(k-1))
         : (2*fib(k-1) + fib(k))*fib(k);

  return f[n];
}

/* Driver program to test above function */
int main(){
  int n = 9;
  printf("%d ", fib(n));
  return 0;
}
```

5.19 MillerTest

```
#include <bits/stdc++.h>
using namespace std;
typedef unsigned long long ll;
ll power(ll x, ll y, ll p){
  ll res = 1;
  x = x % p;
  while (y > 0){
   if (y & 1) res = (res*1LL*x) % p;
   y = y >> 1;
   x = (x * x) % p;
  }
  return res;
}
bool miillerTest(ll d, ll n){
  ll a = 2 + rand() % (n - 4);
  ll x = (ll)power(a, d, n);
  if (x == 1 || x == n-1)
  return true;
  while (d != n-1){
   x = (x *1LL* x) % n;
   d *= 2;
   if (x == 1)   return false;
   if (x == n-1) return true;
  }
  return false;
}
bool isPrime(ll n, ll k){
  if (n <= 1 || n == 4) return false;
  if (n <= 3) return true;
  ll d = n - 1;
  while (d % 2 == 0) d /= 2;
  // Iterate given nber of 'k' times
  for (ll i = 0; i < k; i++)
  if (miillerTest(d, n) == false)
   return false;
  return true;
}
int main(){
  ll k = 4; // Number of iterations
  ll n = 982451653;
  cout << (isPrime(n, k)?"True":"False") << endl;
  return 0;
}
```

5.20 NaiveFind

```
#include <bits/stdc++.h>
using namespace std;
int main(){
  string needle = "CD", haystack ="MANICD";
  if(haystack.find(needle) != string::npos) cout << "Gotcha!!!";
  else cout << "Not Gotcha";
  cout << endl;
  return 0;
}
```

5.21 PollarRho

```
#include<bits/stdc++.h>
using namespace std;

typedef long long ll;
ll num;

int modular_pow(ll  base, int exponent, ll  modulus){
        ll result = 1;
        while (exponent > 0){
                if (exponent & 1)
                        result = (result * base) % modulus;
                exponent = exponent >> 1;
                base = (base * base) % modulus;
        }
        return result;
}
//take care if its' prime infinite loop
ll  PollardRho(ll  n){
        srand (time(NULL));
        if (n==1) return n;
        if (n % 2 == 0) return 2;
        ll  x = (rand()%(n-2))+2;
        ll y = x;
        ll  c = (rand()%(n-1))+1;
        ll d = 1;
        cout << n << endl;
        while (d==1){
                cout << d<<endl;
                x = (modular_pow(x, 2, n) + c + n)%n;
                y = (modular_pow(y, 2, n) + c + n)%n;
                y = (modular_pow(y, 2, n) + c + n)%n;
                d = __gcd(abs(x-y), n);
                if (d==n) return PollardRho(n);
        }
        return d;
}

int main(){
        num = 982451653;
        printf("One of the divisors for %lld is %lld.",num,
            PollardRho(num));
        return 0;
}
```

5.22 PollarRho

```
import random as r
def gcd( a, b):
    if(b == 0): return a;
    return gcd(b, a % b);
def pollardRho(N):
    if N%2==0: return 2
    x = r.randint(1, N-1)
    y = x
    c = r.randint(1, N-1)
    g = 1
    while g==1:
        x = ((x*x)%N+c)%N
        y = ((y*y)%N+c)%N
        y = ((y*y)%N+c)%N
        g = gcd(abs(x-y),N)
    return g
if(__name__=="__main__"):
    print(pollardRho(10967535067))
    print(pollardRho(113))
```

5.23 PrimalyTest

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
bool isPrime(ll n){
  if (n < 2) return false;
  if (n < 4) return true;
  if (n % 2 == 0 || n % 3 == 0) return false;
  if (n < 25) return true;
  for(int i = 5; i*i <= n; i += 6){
    if(n % i == 0 || n % (i + 2) == 0)
        return false;
  }
  return true;
}
int main(){
    cout << isPrime(23234) << endl;
    cout << isPrime(2) << endl;
    cout << isPrime(7454) << endl;
    cout << isPrime(976) << endl;
    cout << isPrime(1973) << endl;
    return 0;
}
```

5.24 RotateMatrix

```
#include<bits/stdc++.h>
using namespace std;
#define R 4
#define C 4
int arr[R][C];

void reverseColumns(){
  for (int i=0; i<C; i++)
    for (int j=0,  k=C-1; j<k; j++,k--)
      swap(arr[j][i], arr[k][i]);
}
void transpose() {
  for (int i=0; i<R; i++)
        for (int j=i; j<C; j++)
      swap(arr[i][j], arr[j][i]);
}
/* anticlockwise rotate matrix by 90 degree*/
void rotate90(){
  transpose();
  reverseColumns();
}

int main() {
  int aux [R][C]=
   { {1, 2, 3, 4},
     {5, 6, 7, 8},
     {9, 10, 11, 12},
     {13, 14, 15, 16}
   };
   rotate90();
   return 0;
}
```

5.25 Sieve

```
#include <bits/stdc++.h>
#define tam 1000
using namespace std;
typedef long long ll;
typedef vector< bool > vbool;
void show (vbool primes){
  ll cap = primes.size();
  for(ll i = 0; i< cap; i++){
    cout << i << " : " << primes[i] << endl;
  }
}
vbool sieve(ll n){
  vbool sieve (tam);
  for (ll i = 0; i < tam; i++)
    sieve[i] = true;
    sieve [0] = sieve[1] = false;
    ll root = sqrt(n);
    for (ll i = 2; i < root; i++){ //find primes
      if(sieve[i]){
        //removes all the multiples
        //of the current prime
        for (ll k = i*1LL*i; k<= n; k+=i){
          sieve[k] = false;
        }
      }
    }
    return sieve;
}
int main(){
    vbool primes = sieve(1000);
    show(primes);;
    primes.clear();
    return 0;
}
```

5.26 Sum

```
/*
  Summatories
*/
int main(){
  sum(i) from 1 to n = n(n+1)/2
  sum(i^2) from 1 to n = n(n+1)(2n+1)/6
  sum(i^3)  from 1 to n = (n^2(n+1)^2)/4

  //Geometric serie
  a * sum(r^k) from 0 to n = a * (1-r ^(n+1)) / (1 -r)
 //  ar + ar^2 + ar^3 ...
}
```

5.27 toBin

```
#include <bits/stdc++.h>
using namespace std;
void toBin(int x){
  for (int i =31; i>=0; --i)
    cout << ((x&(1LL<<i))!=0);
}
int main (){
  toBin(10);
  return 0;
}
```

6 other

6.1 MergeSortPY

```
def merge_sort(arr):
    if (len(arr)>1):
        mid = len(arr) // 2
        lefthalf, righthalf = arr[:mid] , arr[mid:]
        merge_sort(lefthalf)
        merge_sort(righthalf)

        merge(lefthalf, righthalf, arr)


def merge(lh, rh, arr):
    il = 0
    ir = 0
    k = 0
    while il < len(lh) and ir < len(rh):
        if (lh[il] < rh[ir]):
            arr[k] = lh[il]
            il = il+1
        else:
            arr[k] = rh[ir]
            ir = ir+1
        k = k+1

    while il < len (lh):
        arr[k]= lh[il]
        il = il +1
        k = k+1

    while ir < len(rh):
        arr[k] = rh[ir]
        ir = ir +1
        k = k+1

def main():
    array = [-10, 37, 98 , 0 ,12, 192, 5]
    print("Original Array")
    print(array)
    merge_sort(array)

    print("Sorted Array")
    print(array)
main()
```

6.2 Partitions

```
#include<iostream>
using namespace std;
/*
  Generate all unique
  partitions of a given integer
*/

void partitions(int n){
  int p[n];
  int k = 0;
  p[k] = n;

  while (true){
   for( int i =0; i <=k ; i++) cout <<p[i] << " ";
   cout <<endl;
   int rem_val = 0;
   while (k >= 0 && p[k] == 1){
    rem_val += p[k];
    k--;
   }
   if (k < 0) return;
   p[k]--;
   rem_val++;
   // If rem_val is more, then the sorted order is violated.
       Divide
   // rem_val in different values of size p[k] and copy these
       values at
   // different positions after p[k]
   while (rem_val > p[k]){
    p[k+1] = p[k];
    rem_val = rem_val - p[k];
    k++;
   }
   // Copy rem_val to next position and increment position
   p[k+1] = rem_val;
   k++;
  }
}
int main(){
  cout << "All Unique Partitions of 7 \n";
  partitions(7);
  return 0;
}
```

6.3 TemplateC

```
#include <bits/stdc++.h>
using namespace std;

// INT_MAX  -> limits.h
typedef long long ll;
typedef long double ld;
typedef vector < int > vi;
typedef vector < vi > vii;
struct point {int x, y;};

#define show(x) cout << #x << " = " << x << endl;
#define isOdd(x) (x & 0x01)
#define mod(a,b) (b + (a % b)) % b)

const double PI = acos(-1);
const ld INF = 1e18;
const double EPS = 1e-15;

void input(){
  /*
  scanf("%ld",&value); //long y long int
  scanf("%c",&value); //char
  scanf("%f",&value); //float
  scanf("%lf",&value); //double
  scanf("%s",&value); //char*
  scanf("%lld",&value); //long long int
  scanf("%x",&value); //int hexadecimal
  scanf("%o",&value); //int octal
  */
}

void tricks(){
  int a=21,b=16,c=8;
  //if the numbers are long and long long end with and l or two l
  /*ie
    int
    __builtin_popcount
    long
    __builtin_popcountl
    long long
    __builtin_popcountll
    */
  //log2 floor
  show(__lg(21));
  show(__lg(16));
  show(__lg(8));
  cout << endl;
  //count the number of ones
  show(__builtin_popcount(16));
  show(__builtin_popcount(15));
  show(__builtin_popcount(0));
  cout << endl;

  //count the trailing zeros zer
  show(__builtin_ctz(16));
  show(__builtin_ctz(5));
  cout << endl;

  //count the leading zeros
  show(__builtin_clz(32));
  show(__builtin_clz(1024));
  cout << endl;
  //Returns one plus the index of the least significant
  //1-bit of x, or if x is zero, returns zero.
  show(__builtin_ffs(5));
  cout << endl;
  //Is a number   x   power of 2?
  show(((a & (a-1))==0));
  show(((b & (b-1))==0));
  cout << endl;

  //turn on the first n bits of a mask
  show(((1LL<<10)-1));
}

//Main
int main(){
  ios::sync_with_stdio(false);
  cin.tie(0);

  tricks();
  #ifdef LOCAL
    freopen("in.txt", "r", stdin);
    freopen("out.txt", "w", stdout);
  #endif
}
```

6.4 TemplateP

```
from sys import stdin
lines = stdin.read().splitlines()
for line in lines:
  a, b = [int(y) for y in line.split()]
```

6.5 UpperLowerBound

```
#include <bits/stdc++.h>
using namespace std;
int main () {
  int myints[] = {10,20,30,30,20,10,10,20};
  vector<int> v(myints,myints+8);           // 10 20 30 30 20 10
      10 20
  sort (v.begin(), v.end());                // 10 10 10 20 20 20
      30 30
  vector<int>::iterator low,up;
  low=lower_bound (v.begin(), v.end(), 20); //          ^
  up= upper_bound (v.begin(), v.end(), 20); //                   ^
  cout << "lower_bound at position " << (low- v.begin()) << '\n';
  cout << "upper_bound at position " << (up - v.begin()) << '\n';
  return 0;
}
```

6.6 XIncludes

```
#include <vector>      vector<
#include <queue>       queue< priority_queue<
#include <set>         set< multiset<
#include <map>         map< multimap<
#include <bitset>      bitset<
#include <list>        list<
#include <deque>       deque<
#include <stack>       stack<
#include <complex>     complex<
#include <hash_map.h>  hash_map<
#include <hash_set.h>  hash_set<
#include <string>      string
#include <algorithm>   sort( stable_sort( make_heap( push_heap(
    pop_heap(
                       lower_bound( upper_bound( equal_range(
                           binary_search(
                       find( find_first_of( count( min( max( swap(
                           fill( copy(
                       next_permutation( prev_permutation(
                       remove( replace( reverse( rotate(
                           random_shuffle(
                       min_element( max_element( nth_element(
                           mismatch(
                       set_difference( set_intersection( set_union(
                       set_symmetric_difference( merge( unique(
                           adjacent_find(
                       lexicographical_compare(
                           lexicographical_compare_3way(
                       equal( includes(
#include <numeric>     accumulate( partial_sum( adjacent_difference
    (
                       inner_product(
#include <iostream>    cin cout cerr istream ostream
#include <fstream>     ifstream ofstream ifstream( ofstream(
#include <sstream>     istringstream ostringstream
#include <cassert>     assert(
#include <cmath>       sin( cos( tan( asin( acos( atan( atan2( sinh
    ( cosh( tanh(
                       sqrt( hypot( abs( exp( pow( ceil( floor(
                           fmod( log( log10(
                       fabs( M_PI
#include <cstdio>      printf( scanf( fprintf( fscanf( sprintf(
    sscanf(
                       getc( fgetc( putc( fputc( getchar( putchar(
                           ungetc(
                       FILE stdin stdout stderr feof( fclose(
                           fflush(
#include <cstdlib>     rand( srand(
#include <cstring>     memcpy( memmove( memchr( memset(
                       strcpy( strncpy( strcat( strncat( strcmp(
                           strncmp(
                       strchr( strrchr( strstr( strtok( strlen(
#include <ctime> time( clock( CLOCKS_PER_SEC
```

6.7 YGenerator

```
#include <bits/stdc++.h>
using namespace std;
int main(){
        #ifdef LOCAL
        freopen("new.c", "w", stdout);
  #endif
  srand (time(NULL));

  int numRandom = 1000;
  cout << numRandom <<endl;
  for( int i=1 ; i<=numRandom ; i++)
        int cant = rand() % 100 +2;
  return 0;

}
```

7 strings

7.1 LIS

```
#include<bits/stdc++.h>
#define debug(x) cout <<#x << " = " << x << endl
using namespace std;

typedef vector < int > vi;
typedef pair < int , int > ii;


void lis( vi & arr){
  int l = arr.size();
  vi m(l, 1);
  for( int i = 1 ; i < l; i++){
    for( int j = 0 ; j < i ; j++){
      if( arr[j] < arr[i])
      m[i] = max( m[i], m[j]+1);
    }
  }
  for( int &i : m) cout << i << " ";
  cout <<endl;
}
int main(){
```

```
#ifdef LOCAL
  freopen("in", "r", stdin);
#endif

  vector < int > A = { 1, 3, 32 ,2 ,78, 9,2};

  lis ( A );

  return 0;
}
```

7.2 LRSubs

```
#include<bits/stdc++.h>
using namespace std;

// Returns the longest repeating non-overlapping substring
string longestRepeatedSubstring(string str){
    int n = str.length();
    int LCSRe[n+1][n+1];
    // Setting all to 0
    memset(LCSRe, 0, sizeof(LCSRe));
    string res; // To store result
    int res_length  = 0; // To store length of result

    // building table in bottom-up manner
    int i, index = 0;
    for (i=1; i<=n; i++){
      for (int j=i+1; j<=n; j++){
        // (j-i) > LCSRe[i-1][j-1] to remove
        if (str[i-1] == str[j-1] &&
          LCSRe[i-1][j-1] < (j - i)){
          LCSRe[i][j] = LCSRe[i-1][j-1] + 1;
          if (LCSRe[i][j] > res_length){
            res_length = LCSRe[i][j];
            index = max(i, index);
          }
        }
        else
          LCSRe[i][j] = 0;
      }
    }
    if (res_length > 0){
            cout <<  (index  - res_length +1)<<endl;
          for (i = index - res_length + 1; i <= index; i++)
          res.push_back(str[i-1]);
            }
    return res;
}

// Driver program to test the above function
int main(){
    string str = "hello,p23puoeouhello,oues";
    cout << longestRepeatedSubstring(str); //hello,
    return 0;
}
```

7.3 StringUtil

```
#include <bits/stdc++.h>
#define pb push_back
using namespace std;
typedef vector <string> vs;
int toNum(string a){
        stringstream toNum(a);
        int num;
        toNum >> num;
        return num;
}
string toString(double d){
  stringstream ss;
  ss << fixed << setprecision(10) << d;
  string num = ss.str();
  return num;
}
void tolowers(string &data){
 transform(data.begin(), data.end(), data.begin(), ::tolower);
}
void replace(string &a,  string &from, string &to){
        int pos=0;
        while((pos = a.find(from,pos)) != string::npos){
                a.replace(pos, to.size(), to);
                pos+=to.size();
        }
}
vs split(string line, char d){
        vector < string > elements;
        stringstream ss(line);
        string item;
        while(getline(ss, item, d))     elements.pb(item);
        return elements;
}

int main(){
  vs d1 = split("1990/10/5", '/');
  for (string s: d1){
    cout << toNum(s) << endl;
  }

  char a = 'a';
  cout << (isalnum(a)?"true":"false")  << endl;
  cout <<( isalpha(a)?"true":"false") << endl;
  cout << (isblank(a)?"true":"false")  << endl;
  cout << (isdigit(a)?"true":"false")  << endl;
  cout << (islower(a)?"true":"false")  << endl;
  cout << (ispunct(a)?"true":"false")  << endl;
  cout << (isupper(a)?"true":"false")  << endl;
  cout << (isxdigit(a) ?"true":"false") << endl;
  cout << (char)tolower(a) << endl;
  cout << (char)toupper(a) << endl;
  string hay ="hellohowareyouhow",  ned ="whatare", from= "how";
  replace(hay, from, ned);
  cout << hay <<endl;
  return 0;
}
```

7.4 SubstrK

```
#include<bits/stdc++.h>
#define debug(x) cout << #x << " = "<< x << endl
#define pb push_back
/*
        Algorithm to find all possible
  substrings of size k given a set of values
*/
using namespace std;
set<string> subs;
//print all possible substrings of size k
void substringSizek(char set[], string prefix, int n, int k){
  //Base case
```

```
    if( 0 == k){
      cout << prefix <<endl;
      subs.insert(prefix);
      return;
          }
    for( int i=0;  i < n ; ++i){
           string newprefix = prefix + set[i];
           //k is decreased because we add a new caracter
           substringSizek(set, newprefix, n, k-1);
          }
  }
```

```
  void init(char set[], int k){
    int n = strlen(set);
          substringSizek(set, "", n, k);
  }
  int main(){
    char set[3] ={'a', 'b'};
    int k = 3;
    init(set, k);
  }
```