UDvorak's notebook (2017)

Contents

1 Snippets

1.1 Foreach

```
#include <iterator>
#define foreach (x,v) for (typeof(v).begin() x=(v).begin(); x!=(v)
    .end(); ++x)
using namespace std;
int main(){
    return 0;
}
```

1.2 Show

```
#include <iostream>
#define show(x) cout << #x << " = " << x << endl;
using namespace std;
int main(){
    int e =32;
    show(e);
}
```

1.3 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;
}
```

1.4 FreOpen

```
#include <iostream>
#include <stdio.h>
using namespace std;
int main (){
  freopen("data.in", "r", stdin);
  freopen("data.out", "w", stdout);
  return 0;
}
```

1.5 Time

```
#include <chrono>
#include <iostream>
using namespace std;
int main(){
  auto start = chrono::high_resolution_clock::now();
  for(long long i = 0; i < 10000000; i++)

  auto end = chrono::high_resolution_clock::now();
  chrono::duration<double> diff = end-start;
  cout << diff.count() << endl ;
  return 0;
}
```

1.6 Size

```
#include <vector>
#include <string>
#include <iostream>
#define sz(a) ((int)(a).size())
using namespace std;
int main(){
    string t = "Hello, what's up";
    vector<int> c (10);
    cout << sz(t) << endl;
    cout << sz(c) << endl;
}
```

1.7 IsOdd

```
#include <iostream>
#define isOdd(x) (x & 0x01)
using namespace std;
int main (){
    int a =57, b= 32;
    cout << isOdd(a) << endl;
    cout << isOdd(b) << endl;
    return 0;
}
```

1.8 Assert

```
#include <iostream>
#include <assert.h>
#define isOdd(x) (x & 0x01)
using namespace std;

void test(int num){
    assert(isOdd(num) == 0);
    cout << "Hello: " << num << endl;
}

int main(){
    int a=10, b=21;
    test(a);
    test(b);
}
```

1.9 StructPriorityQueue

```
#include <iostream>
#include <queue>
using namespace std;
typedef priority_queue<edge> pq;
struct edge{
  int to, weight;
  edge(){}
  edge(int _to, int _weight){
    to = _to;
    weight = _weight;
  }
  bool operator < (edge e) const {
    return weight > e.weight;
  }
};
int main(){
  pq edges;
  edges.push(edge(1, 23));
  edges.push(edge(2, 3));
  edges.push(edge(3, 10));
  edges.push(edge(4, 11));
  edges.push(edge(5, 4));
  while(!edges.empty()){
    edge a = edges.top();
    edges.pop();
    cout << a.to << endl;
  }
}
```

1.10 utilities Directives1

```
#include <iostream>
using namespace std;
#define concat(a, b) a ## b

int main() {
   int xy = 100;
   cout << concat(x, y);
   return 0;
}
```

1.11 utilities Namespace1

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

/*REFERENCE
 *https://www.tutorialspoint.com/cplusplus/cpp_namespaces.htm
 */

// first name space
namespace first_space{
   void func(){
      cout << "Inside first_space" << endl;
   }
}

// second name space
namespace second_space{
   void func(){
      cout << "Inside second_space" << endl;
   }
}

int main () {

   // Calls function from first name space.
   first_space::func();

   // Calls function from second name space.
   second_space::func();

   return 0;
}
```

1.12 utilities ClassPointers

```cpp
#include <iostream>
using namespace std;

class Person {
   public:
      Person() {
         cout << "Constructor called!" <<endl;
      }

      ~Person() {
         cout << "Destructor called!" <<endl;
      }
};

int main( ) {
   Person* myBoxArray = new Person[4];
   delete [] myBoxArray; // Delete array
   return 0;
}
```

1.13 utilities Debug

```cpp
#include <iostream>
using namespace std;
#define DEBUG

#define MIN(a,b) (((a)<(b)) ? a : b)

int main () {
   int i, j;
   i = 100;
   j = 30;

   #ifdef DEBUG
      cerr <<"Trace: Inside main function" << endl;
   #endif

   #if 0
      /* This is commented part */
      cout << MKSTR(HELLO C++) << endl;
   #endif

      cout <<"The minimum is " << MIN(i, j) << endl;

   #ifdef DEBUG
      cerr <<"Trace: Coming out of main function" << endl;
   #endif
      return 0;
}
```

1.14 utilities Directives2

```cpp
#include<iostream>

using std::cout;
using std::cin;
using std::endl;


/*
 * g++ -E test.cpp  > sal.out
 * compile with that command and see how the compiler replace the
     constant
 */
#define PI 3.141516
#define MIN(a,b) (((a)<(b)) ? a : b)

int main(){

    cout << "The number PI is " << PI << endl;
    cout <<"The minimum is " << MIN(i, j) << endl;
    return 0;

}
```

1.15 utilities PointersDeclaration

```cpp
#include <iostream>

using namespace std;

int main(){

    double * data;
    data = new double;

    *data = 123.34;

    cout <<  *data << endl;

    delete data;

    return 0;

}
```

1.16 utilities PredefinedMacros

```cpp
#include <iostream>
using namespace std;

int main () {
   cout << "Value of __LINE__ : " << __LINE__ << endl;
   cout << "Value of __FILE__ : " << __FILE__ << endl;
   cout << "Value of __DATE__ : " << __DATE__ << endl;
   cout << "Value of __TIME__ : " << __TIME__ << endl;

   return 0;
}
```

1.17 utilities Template

```cpp
#include <iostream>
#include <string>


/*REFERENCES
 * https://www.tutorialspoint.com/cplusplus/cpp_templates.htm
 */
using namespace std;

template <typename T>
inline T const& Max (T const& a, T const& b)  {
   return a < b ? b:a;
}

int main () {

   int i = 39;
   int j = 20;
   cout << "Max(i, j): " << Max(i, j) << endl;

   double f1 = 13.5;
   double f2 = 20.7;
   cout << "Max(f1, f2): " << Max(f1, f2) << endl;
```

```
    string s1 = "Hello";
    string s2 = "World";
    cout << "Max(s1, s2): " << Max(s1, s2) << endl;

    return 0;
};
```

1.18 utilities Namespace2

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

/*REFERENCES
 *https://www.tutorialspoint.com/cplusplus/cpp_namespaces.htm
 */
// first name space
namespace first_space{
    void func(){
        cout << "Inside first_space" << endl;
    }
}

// second name space
namespace second_space{
    void func(){
        cout << "Inside second_space" << endl;
    }
}

using namespace first_space;
int main () {

    // This calls function from first name space.
    func();

    return 0;
}
```

1.19 utilities CommaOperator

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

int main() {
    int i, j;

    j = 10;
    i = (j++, j+100, 999+j);

    cout << i;

    return 0;
}
```

1.20 utilities ArrayPointers

```
#include <iostream>

using namespace std;

inline void example_1(){
    char * name;
    name = new char[10];

    delete [] name;
}

inline void example_2(){
    int row = 4;
    int col = 3;

    //Allocate memory for rows
    double **pvalue  = new  double* [row];
    //Now allocate memory for columns
    for (int i=0; i<col; i++){
        pvalue[i] = new double[col];
    }


    //Now release memory
    for(int i = 0; i < row; i++) {
        delete [] pvalue[i];
    }
    delete [] pvalue;
}

int main(){
    example_1();
    example_2();


    return 0;

}
```

1.21 template

```
//Constants and defines
#define show(x) cout << #x << " = " << x << endl;
#define endl '\n'
#define f first
#define s second
#define mp make_pair
#define pb push_back
const double PI = acos(-1);
const ld INF = 1e18;
const double EPS = 1e-15;
// INT_MAX  -> limits.h
typedef long long ll;
typedef vector < int > vi;
typedef vector < vi > vii;

//Geometry
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(r(o,d) , r(o,d)
    )); }

//Input
scanf("%d",&value); //int
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
```

```
//Main
int main(){
  ios::sync_with_stdio(false);
  cin.tie(NULL);
  #ifdef UDVORAK
    freopen("in.txt", "r", stdin);
    freopen("out.txt", "w", stdout);
  #endif
}
```

1.22 For

```
#include <iostream>
#define forn(i, n) for(int i = 0 ; (i) < (n) ; ++i)
using namespace std;
int main(){
  forn(_,10){
    cout << "with out variable" << endl;
  }
  forn(i,10){
    cout << "with variable: " << i << endl;
  }
  return 0;
}
```

1.23 Swap

```
#include <iostream>
#define swap(x,y) (x^=y, y^=x, x^=y)
using namespace std;
int main(){
  int x=324, y=232;
  cout << x << " " << y << endl;
  swap(x,y);
  cout << x << " " << y << endl;
  return 0;
}
```

1.24 StringStream

```
#include <bits/stdc++.h>
using namespace std;
int main(){
    string line;
    while (getline(cin, line)){
        stringstream ss(line);
        string word;
        int count = 0;
        while ( ss >> word) count ++;
        cout << endl << "# Words: " << count << endl;
    }
}
```

1.25 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;
}
```

1.26 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");
  }
}
```

2 BasicOperations

2.1 Exponentiation

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
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;
}
```

2.2 SumArray

```
#include <bits/stdc++.h>
using namespace std;
typedef vector<int> vi;
inline void show(vi &n2){
    for (int i = 0; i < n2.size() ; i++){
        cout << n2[i];
    }
    cout << endl;
}
inline vi sum_arrays(vi &a1, vi &a2){
    int tam = a1.size();
    vi result(tam + 1,0);
    int carry = 0, aux = 0;
    for (int i = tam; i>= 0; i--){
        aux = a1[i] + a2[i]+ carry;
        result[i+1] = aux % 10;
        carry  = aux >= 10 ? 1 : 0;
    }
    result[0] = carry;
    return result;
}

int main(){
    int vals[] =
        {3,4,1,0,9,8,7,3,4,9,4,3,5,9,2,3,9,0,4,5,8,7,0,2,4,5,2};
    int val2[] =
        {9,4,1,0,2,8,7,3,4,9,4,3,5,9,2,3,9,0,4,5,8,7,0,2,4,5,2};
    vi n1(begin(vals), end(vals));
    vi n2(begin(val2), end(val2));
    vi result1 = sum_arrays(n1, n2);
    show(result1);
    return 0;
}
```

3 Structures

3.1 DisjointSets

```
#include <bits/stdc++.h>
using namespace std;
typedef vector<int> vi;
struct union_find {
  vi data, pe;
  union_find(int n) : data(vi(n)), pe(vi(n)) {
    for(int i=0; i<data.size(); i++)
      data[i] = i;
  }
  int find(int x) {
    if(x == data[x]) return x;
    data[x] = find(data[x]);
    return data[x];
  }
  bool unite(int x, int y) {
    int px = find(x);
    int py = find(y);
    if(px == py) return false;
    if(pe[px] > pe[py]) swap(px, py);
    pe[px] += pe[py];
    data[py] = px;
    return true;
  }
};
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;
}
```

3.2 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;
}
```

3.3 Prim

```
#include <bits/stdc++.h>
using namespace std;
const int MAXN = 10005;
#define pb push_back
typedef pair <int, int> edge;
// Pareja (nodo, peso)
typedef pair <int, int> weight_node; // Pareja (peso, nodo)
```

```
vector <edge> g[MAXN];
// Lista de adyacencia
bool visited[MAXN];
// Retorna el costo total del MST
int prim(int n){ // n = n mero de nodos
    for (int i = 0; i <= n; ++i) visited[i] = false;
    int total = 0;
    priority_queue<weight_node, vector <weight_node>,
    greater<weight_node> > q;
    // Empezar el MST desde 0 (cambiar si el nodo 0 no existe)
    q.push(weight_node(0, 0));
    while (!q.empty()){
        int u = q.top().second;
        int w = q.top().first;
        q.pop();

        if (visited[u]) continue;
        visited[u] = true;
        total += w;
        for (int i = 0; i < g[u].size(); ++i){
            int v = g[u][i].first;
            int next_w = g[u][i].second;
            if (!visited[v]){
                q.push(weight_node(next_w, v));
            }
        }
    }
    return total;
}

int main(){
    //Nodo 0 se une al 1 con peso 1
    g[0].pb(edge(1,1));
    //Nodo 0 se une al 2 con peso 2
    g[0].pb(edge(2,2));
    //Nodo 0 se une al 3  con peso 3
    g[0].pb(edge(3,3));
    g[1].pb(edge(5,4));
    g[2].pb(edge(4,5));
    g[3].pb(edge(4,1));
    cout << prim(4);
    return 0;
}
```

3.4 SegmentTree

```
#include <iostream>
#define left(x) x << 1
#define right(x) x << 1 | 1
#define ROOT 1
using namespace std;

void build(int *T, int *A, int node, int start, int end) {
  if(start == end) {
      T[node] = A[start];
  } else {
      int mid = (start + end) / 2;
      build(T, A, left(node), start, mid);
      build(T, A, right(node), mid + 1, end);
      // Merging the children
      T[node] = T[left(node)] + T[right(node)];
  }
}

void update(int *T, int *A, int node, int start, int end, int i,
    int val ) {
  if(start == end) {
    A[i] = val;
    T[node] = val;
  } else {
    int mid = (start + end) / 2;
    if(start <= i && i <= mid)
      update(T, A, left(node), start, mid, i, val);
    else
      update(T, A, right(node), mid + 1, end, i, val);
    T[node] = T[left(node)] + T[right(node)];
  }
}

int query(int *T, int node, int start, int end, int a, int b) {
  if(b < start || end < a)  // out of he boundaries
    return 0;
  if(a <= start && end <= b)
    return T[node];
  int mid = (start + end) / 2;
  int p1 = query(T, left(node), start, mid, a, b);
  int p2 = query(T, right(node), mid + 1, end, a, b);
  return p1 + p2;
}
int main() {
  int size = 5;
  int *a = new int[size];
  int *stree = new int[4 * size];
  for (int i = 0;  i < size ; i++) a[i] = i + 1;
  build(stree, a, ROOT, 0, size - 1);
  int from = 0, to = 4;
  cout << query(stree, ROOT, 0, size-1, from, to) << endl;
  return 0;
}
```

3.5 FenwickTree

```
#include <bits/stdc++.h>
using namespace std;
#define flag(x) printf("[%d]\n", x)
typedef vector<int> vi;
struct fenwick_tree {
  vi data;
  fenwick_tree(int _n) : data(vi(_n + 1, 0)) {}
  void update(int i, int val)  {
    while(i < data.size()) {
      data[i] += val;
      i += i & (-i);
    }
  }
  int query(int i) {
    int sum = 0;
    while(i > 0) {
      sum += data[i];
      i -= i & (-i);
    }
    return sum;
  }
  int query_segment(int a, int b) {
    return query(b) - query(a - 1);
  }
};
int main() {
  int x[5] = {1, 2, 3, 4, 5};
  fenwick_tree *fq = new fenwick_tree(8);
  for(int i=0; i<5; i++)
    fq->update(i + 1, x[i]);
```

```
    //Node 0 -> dummy node
    for(int i=1; i<fq->data.size(); i++) {
        cout << fq->data[i] << ' ';
    }cout << endl;
    //Sum interval [1 - 4]
    flag(fq->query(4));
    //Sum interval [3 - 5]
    flag(fq->query_segment(3, 5));
    return 0;
}
```

3.6 MaxMinPHeap

```
/**Utility STL Data Structures*/
/**Max Heaps*/
priority_queue <int> pq;
/**Min Heaps*/
priority_queue <int, vector<int>, greater<int> > pq;
```

3.7 BinaryTree

```
#include <iostream>
using namespace std;
struct node {
  int val = 0;
  node * l = nullptr;
  node * r = nullptr;
};
inline node* build(node *head, int value){
  node *son = new node;
  son->val = value;
  if (head == nullptr) return son;
  node * aux = head, * nxt = head;
  while(nxt != nullptr){
    aux = nxt;
    if (value > nxt->val) nxt = nxt->r;
      else nxt = nxt->l;
  }
  if(value > aux-> val) aux->r = son;
  else aux->l = son;
  return head;
}
inline void show(node* head){
  if (head==nullptr) return;
  show(head->l);
  cout << head->val << endl;
  show(head->r);
}
int main(){
  node *head = new node;
  head->val = 5;
  head = build(head, 45);
  head = build(head, 20);
  show(head);
  return 0;
}
```

3.8 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;
        }
}

//This is just the driver program
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;
}
```

3.9 Kruskals

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

typedef vector<int> vi;
```

```
typedef vector<bool> vb;
typedef pair<int, int> pii;
typedef pair<long long, pii> edge;
typedef vector<pii> vpii;
typedef vector<edge> E;

long long weight;
int vertex;

vpii mst(int n, E &edges, vb &vis) {
  weight = 0; vertex = 0;
  union_find uf(n);
  sort(edges.begin(), edges.end());
  vpii res;
  for(int i=0; i<edges.size(); i++) {
    int x = edges[i].second.first;
    int y = edges[i].second.second;
    if(uf.find(x) != uf.find(y)) {
      if(!vis[x]) {
        vertex++;
        vis[x] = true;
      }
      if(!vis[y]) {
        vertex++;
        vis[y] = true;
      }
      weight += edges[i].first;
      res.push_back(pii(min(x, y), max(x, y)));
      uf.unite(x, y);
    }
  }
  return res;
}

int main() {
  int v, e, x, y, w;
  while(scanf("%d %d", &v, &e) && (v + e)){
    E list(e);
    vb vis(v);
    for(int i=0; i<e; i++) {
      scanf("%d %d %d", &x, &y, &w);
      list[i] = edge(w, pii(x, y));
    }
    vpii answ = mst(v, list, vis);
    if(vertex == v) {
      printf("%lld\n", weight);
      sort(answ.begin(), answ.end());
      for(int i=0; i<answ.size(); i++){
        printf("%d %d\n", answ[i].first, answ[i].second);
      }
    }
    else printf("Impossible\n");
  }
  return 0;
}
```

3.10 MaxFlow

```
#include <bits/stdc++.h>
using namespace std;
#define V 6
bool bfs(int rGraph[V][V], int s, int t, int parent[]){
        bool visited[V];
        memset(visited, 0, sizeof(visited));
        queue <int> q;
        q.push(s);
        visited[s] = true;
        parent[s] = -1;
        while (!q.empty()){
                int u = q.front();
                q.pop();
                for (int v=0; v<V; v++){
                        if (visited[v]==false && rGraph[u][v] > 0)
                            {
                                q.push(v);
                                parent[v] = u;
                                visited[v] = true;
                        }
                }
        }
        return (visited[t] == true);
}

int fordFulkerson(int graph[V][V], int s, int t){
        int u, v;
        int rGraph[V][V]; // Residual graph where rGraph[i][j]
            indicates
                                        // residual capacity of
                                            edge from i to j (if
                                            there
                                        // is an edge. If rGraph[i
                                            ][j] is 0, then there
                                            is not)
        for (u = 0; u < V; u++)
                for (v = 0; v < V; v++)
                        rGraph[u][v] = graph[u][v];
        int parent[V];
        int max_flow = 0; // There is no flow initially
        while (bfs(rGraph, s, t, parent)){
                int path_flow = INT_MAX;
                for (v=t; v!=s; v=parent[v]){
                        u = parent[v];
                        path_flow = min(path_flow, rGraph[u][v]);
                }
                for (v=t; v != s; v=parent[v]){
                        u = parent[v];
                        rGraph[u][v] -= path_flow;
                        rGraph[v][u] += path_flow;
                }
                max_flow += path_flow;
        }
        return max_flow;
}

int main(){
        int graph[V][V] = { {0, 16, 13, 0, 0, 0},
                                                {0, 0, 10, 12, 0,
                                                    0},
                                                {0, 4, 0, 0, 14,
                                                    0},
                                                {0, 0, 9, 0, 0,
                                                    20},
                                                {0, 0, 0, 7, 0,
                                                    4},
                                                {0, 0, 0, 0, 0, 0}
                                            };
  int origen = 0;
  int dest = 5;
        cout << "The maximum possible flow is "
       << fordFulkerson(graph, origen, dest);

        return 0;
}
```

4 NP*problem*

4.1 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 Combinatory

5.1 BinomialCPP

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

5.2 BinomialPYTHON

```
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])
```

6 Primes

6.1 PollarRhoCPP

```
#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;
}
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){
                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 = 124554;
        printf("One of the divisors for %lld is %lld.",num,
            PollardRho(num));
        return 0;
}
```

6.2 PollarRhoPYTHON

```
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))
```

6.3 MillerTest

```
#include <bits/stdc++.h>
using namespace std;
typedef unsigned long long ll;
int power(ll x, ll y, ll p){
        int res = 1;
        x = x % p;
        while (y > 0){
                if (y & 1) res = (res*x) % p;
                y = y >> 1;
                x = (x * x) % p;
        }
        return res;
}

bool miillerTest(long long d, long 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 = (ll)(x * 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;
}
```

6.4 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");
        }
}
```

6.5 Sieve

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

```
    return 0;
}
```

6.6 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;
}
```

6.7 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 < long long > vll = fact(num_to_fact);
    for (int x=0; x< vll.size(); x++){
        cout << vll[x] << " ";
    }
    cout << endl;
}
```

7 Search

7.1 BinarySearch

```
#include <bits/stdc++.h>
using namespace std;
const int TAM = 5;
int arr[TAM];
/* Recursive
 * l -> left
 * r -> right
 * x -> element to search
 */
int binarySearchRec(int l, int r, int x){
  if (r >= l){
    int mid = l + (r - l)/2;
    // The element in the middle
    if (arr[mid] == x)  return mid;
    // Smaller of the middle element
    if (arr[mid] > x) return binarySearchRec( l, mid-1, x);
    // Greater that the middle element
    return binarySearchRec(mid+1, r, x);
  }
 return -1;
}

/* Iterative
 * l -> left
 * r -> right
 * x -> element to search
 */
int binarySearchIte( int l, int r, int x){
  while (l <= r){
    int m = l + (r-l)/2;
    // The element in the middle
    if (arr[m] == x)  return m;
    // Smaller of the middle element
    if (arr[m] < x)  l = m + 1;
    // Greater that the middle element
    else r = m - 1;
  }
  // if we reach here, then element was not present
  return -1;
}

int main(void){
    arr[0] = 2;
    arr[1] = 3;
    arr[2] = 4;
    arr[3] = 10;
```

```
    arr[4] = 40;
    int x = 10;
    int result = binarySearchIte(0, TAM-1, x);
    (result == -1)? printf("Element is not present in array")
                  : printf("Element is present at index %d \n",
                     result);
    return 0;
}
```

8 Sorting

8.1 SelectionSort SelectionSortPYTHON

```
def show(elements):
    for element in elements:
        print(element, end =" ")
    print("")

def sort(elements, array_length):
    for i in range(0, array_length):
        smallest = elements[i]
        pos_smallest = i
        for index in range(i+1,array_length):
            if elements[index] < elements[pos_smallest]:
                pos_smallest = index
        aux = elements[i]
        elements[i] = elements[pos_smallest]
        elements[pos_smallest] =aux

if __name__ == "__main__":
    elements = [23, -3, 85, 0, 21, -10, 40]
    array_length =  len(elements)
    print("original array")
    show(elements)
    print("sorted array")
    sort(elements, array_length)
    show(elements)
```

8.2 SelectionSort SelectionSortCPP

```
#include<iostream>
using namespace std;
int show(int array[], int length_array){
    for (int index = 0 ; index < length_array; index++)
        cout << array[index] << " ";
    cout << endl;

}
int sort(int array[], int length_array){
  for (int index = 0; index < length_array; index++){
    int pos_smallest = index;
    for(int index_aux = index+1; index_aux < length_array;
        index_aux++){
      if (array[pos_smallest] > array[index_aux]){
        pos_smallest = index_aux;
      }
    }
    //Swap
    if( pos_smallest != index){
      int aux = array[index];
      array[index] = array[pos_smallest];
      array[pos_smallest] = aux;
    }
  }
}
int main(){
  int length_array = 7;
  int array [] = {58, -5, 10, 8, 78 ,234, 43};
  cout << "Original Array" << endl;
  show(array, length_array);
  sort(array, length_array);
  cout << "Sorted Array" << endl;
  show(array, length_array);
}
```

8.3 InsertionSort InsertionSortPYTHON

```
def show(array):
    for element in array:
        print(element, end = " ")
    print("")
def sort(array, length_array):
    for index in range(1,length_array):
        key = array[index]
        index_aux = index -1
        while index_aux >=0 and array[index_aux]>key:
            array[index_aux+1] = array[index_aux]
            index_aux = index_aux -1
        array[index_aux+1]=key
def main():
    array = [50, 885, 1 , -8 , 54 , 2, 54, 0]
    print("Original array")
    show(array)
    print("Sorted array")
    sort(array, len(array))
    print(array)
main()
```

8.4 InsertionSort InsertionSortCPP

```
#include <iostream>
using namespace std;
void show(int array[], int length_array){
  for (int index = 0; index < length_array; index ++)
    cout << array[index] << " ";
  cout<< endl;
}
void sort(int array[], int length_array){
  for (int index = 1; index < length_array; index ++){
    int key = array[index];
    int index_aux = index - 1;
    while (index_aux >= 0 && array[index_aux] > key){
      array[index_aux + 1 ] = array[index_aux];
      index_aux = index_aux -1;
    }
    array[index_aux + 1] = key;
  }
}

int main(){
  int length_array = 8;
  int array[]  = {50, 885, 1 , -8 , 54 , 2, 54, 0};
  show(array, length_array);
  sort(array, length_array);
  show(array, length_array);
}
```

8.5 BubbleSort Bubble

```cpp
#include <bits/stdc++.h>
#define forn(i,j,k) for (int i=j; i<k; i++)
using namespace std;
typedef long long ll;
inline void sort(ll *arr, int size){
  forn(i,0,size-1)
    forn(j,0, size-i-1)
      if (arr[j] > arr[j+1])
        swap(arr[j], arr[j+1]);
}

int main(){
    int size =8;
    ll *data  = new ll[size];
    forn(i, 0, size)
        scanf("%lld", &data[i]);
    sort(data, size);
    forn(i, 0, size)
        printf("%lld ",data[i]);
    return 0;
}
```

8.6 MergeSort MergeSortCPP

```cpp
#include<bits/stdc++.h>
using namespace std;
void show(int array [], int length_array){
  int index = 0;
  // cout<< "size : "<< array.size() <<endl;
  while(index<length_array){
    printf("%d ",array[index]);
    index = index +1;
  }
  printf("\n");
}
void sort(int array[], int pos_ini, int pos_final){
  /*In this condition the len of the array
  left and right half arrays will be
  of 1 element both */
  if(pos_final > pos_ini){
   //first calc the half point
    int pos_mid = (pos_ini+pos_final)/2;
    sort(array, pos_ini, pos_mid);
    sort(array, pos_mid +1, pos_final);
    merge(array, pos_ini, pos_mid, pos_final);
    }
}

void merge( int array[], int pos_ini, int pos_mid,int pos_final){
  int size_left = pos_mid - pos_ini + 1;
  int size_right =  pos_final - pos_mid;
  /* create temp arrays */
  int lefthalf[size_left], righthalf[size_right];
  for (int i = 0; i < size_left; i++)
      lefthalf[i] = array[pos_ini + i];
  for (int j = 0; j < size_right; j++)
      righthalf[j] = array[pos_mid + 1+ j];
  int index_right_half  = 0;
  int index_left_half = 0;
  int index = pos_ini;
  while (index_left_half < size_left && index_right_half <
      size_right) {
    if(lefthalf[index_left_half] <= righthalf[index_right_half]){
      array[index] = lefthalf[index_left_half];
      index_left_half = index_left_half +1;
    }else{
      array[index] = righthalf[index_right_half];
      index_right_half  = index_right_half+1;
    }
    index = index +1;
  }
  //Copy the remaining elements if there is any
  while( index_left_half < size_left){
    array[index] =  lefthalf[index_left_half];
    index_left_half = index_left_half +1;
    index = index +1;
  }
  while( index_right_half < size_right){
    array[index] =  righthalf[index_right_half];
    index_right_half = index_right_half +1;
    index = index +1;
  }
}

int main(){
  int array[] = {-10, 37, 98 , 0 ,12, 192, 5};
  int length_array = sizeof(array)/ sizeof(array[0]);
  show(array, length_array);
  sort(array, 0, length_array  -1);
  show(array, length_array);
}
```

8.7 MergeSort MergeSortPY

```python
def merge_sort(array):
    ##Stop when the len of the array is less or equal than one
    if len(array)>1:
        #Calc the mid of the array
        mid = len(array) // 2 # // mean integer division

        #Create two arrays left and right
        lefthalf = array[:mid]
        print(lefthalf)

        righthalf = array[mid:]
        print(righthalf)

        #Divide the subarrays left and right
        merge_sort(lefthalf)
        merge_sort(righthalf)

        ##I send the array as a argument to change the same array
            and not another
        merge(lefthalf, righthalf, array)

def merge(lefthalf, righthalf, array):
    index_array_left=0
    index_array_right=0
    k=0
    while index_array_left < len(lefthalf) and \
          index_array_right < len(righthalf):

        if lefthalf[index_array_left] < \
            righthalf[index_array_right]:

            #assign the less to the new array
            array[k]=lefthalf[index_array_left]

            """As the less was an element in the lefthalf we dont
            need to compare this again so we increase the index
```

```
            of the left array"""
            index_array_left=index_array_left+1

        else:
            array[k]=righthalf[index_array_right]

            """As the less was an element in the righthalf we dont
            need to compare this again so we increase the index
            of the left array"""
            index_array_right=index_array_right+1

        #It is necesary increase the pos of the original array
        k=k+1

    ##add the remaining elements
    while index_array_left < len(lefthalf):
        array[k]=lefthalf[index_array_left]
        index_array_left=index_array_left+1
        k=k+1

    while index_array_right < len(righthalf):
        array[k]=righthalf[index_array_right]
        index_array_right=index_array_right+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()
```

8.8 StandardSort

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef vector < ll > vl;
int main(){
    vl data = {234234LL, 2322LL,1LL, -1LL, 3454LL};
    sort(data.begin(), data.end());
    for (int i=0; i< data.size(); i++)
        printf("%lld ", data[i]);
    return 0;
}
```

9 Strings

9.1 Palindrome

```
#include <iostream>
#include <string>

using namespace std;
inline bool evaluate(string word, int i, int j){
  if (i >= j) return true;
  else if (word[i] != word[j]) return false;
  return evaluate(word, i+1, j-1);
}
inline bool is_palindrome(string word){
  int length = word.length();
  if (length == 1) return true;
  return evaluate(word, 0, length-1);
}

int main(){
  string word = "anamariaairamana";
  string word2 = "Thisssiss";
  cout << word << " ";
  cout << is_palindrome(word) << endl;
  cout << word2 << " ";
  cout << is_palindrome(word2) << endl;
  return  0;
}
```

9.2 FunctionsOverChart

```
#include <bits/stdc++.h>
using namespace std;
int main(){
  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;
  return 0;
}
```

9.3 Split

```
#include <bits/stdc++.h>
using namespace std;
/*
 * Split by space
 */
int main(){
    string line;
    while(getline(cin, line)){
        stringstream ss;
        ss.str(line);
        string item;
        while (getline(ss, item,' ')) {
            cout << item << endl;
        }
    }
    return 0;
}
```

9.4 Regex

```
#include <iostream>
#include <iterator>
#include <regex>
#include <string>
using namespace std;
int main(){
    string  s = "123daniel , jajaja, lol, 234234534, I am from
        Earth";
```

```
    regex tel("\\d{8},\\sI");
    auto words_begin = sregex_iterator(s.begin(), s.end(), tel);
    auto words_end = sregex_iterator();
    cout << "Found " << distance(words_begin, words_end)<< " words
        \n";
    const int N = 6;
    for (sregex_iterator i = words_begin; i != words_end; ++i) {
        smatch match = *i;
        string match_str = match.str();
        if (match_str.size() > N) {
            cout << "  " << match_str << '\n';
        }
    }
    return 0;
}
```

9.5 KMP

```
#include <bits/stdc++.h>
using namespace std;
bool kmp(const string &needle, const string &haystack){
    int m = needle.size();
    vector<int> border(m);
    border[0] = 0;
    for (int i = 1; i < m; ++i) {
        border[i] = border[i - 1];
        while (border[i] > 0 and needle[i] != needle[border[i]])
        border[i] = border[border[i] - 1];
        if (needle[i] == needle[border[i]]) border[i]++;
    }
    int n = haystack.size();
    int seen = 0;
    for (int i = 0; i < n; ++i){
        while (seen > 0 and haystack[i] != needle[seen])
        seen = border[seen - 1];
        if (haystack[i] == needle[seen]) seen++;
        if (seen == m) return true; // Ocurre entre [i - m + 1, i]
    }
    return false;
}

int main(){
    string a ="hola";
    string b ="thauautholueeo";
    cout << kmp(a, b);
}
```

9.6 LCI

```
#include <bits/stdc++.h>
using namespace std;
//Compute the largest increasing subsequence
int lis( int arr[], int n ){
        int *lis, i, j, max = 0;
        lis = (int*) malloc ( sizeof( int ) * n );
        for (i = 0; i < n; i++ )
                lis[i] = 1;
        for (i = 1; i < n; i++ )
                for (j = 0; j < i; j++ )
                        if ( arr[i] > arr[j] && lis[i] < lis[j] +
                            1)
                                lis[i] = lis[j] + 1;
        for (i = 0; i < n; i++ )
                if (max < lis[i])
                        max = lis[i];
        free(lis);
        return max;
}
int main(){
        int arr[] = { 10, 22, 9, 33, 21, 50, 41, 60 };
        int n = sizeof(arr)/sizeof(arr[0]);
        printf("Length of lis is %d\n", lis( arr, n ) );
  //sol = 10, 22, 33, 50, 60
        return 0;
}
```

9.7 LCS

```
#include <bits/stdc++.h>
#define endl '\n'
using namespace std;
const int M_MAX = 20; // M ximo size del String 1
const int N_MAX = 20; // M ximo size del String 2
int m, n; // Size de Strings 1 y 2
string X; // String 1
string Y; // String 2
int memo[M_MAX + 1][N_MAX + 1];
int lcs (int m, int n) {
  for (int i = 0; i <= m; i++) {
    for (int j = 0; j <= n; j++) {
      if (i == 0 || j == 0) memo[i][j] = 0;
      else if (X[i - 1] == Y[j - 1]) memo[i][j] = memo[i - 1][j -
          1] + 1;
      else memo[i][j] = max(memo[i - 1][j], memo[i][j - 1]);
    }
  }
  return memo[m][n];
}
int main(){
  X = "XMJYAUZ";
  Y = "MZJAWXU";
  cout << lcs(X.size(), Y.size()) <<endl;
  //Sol = MJAU
  return 0;
}
```

10 Math

10.1 Matrix GaussianElimination

```
// C++ program to demostrate working of Guassian Elimination
// method
#include<bits/stdc++.h>
using namespace std;

#define N 3      // Number of unknowns

// function to reduce matrix to r.e.f. Returns a value to
// indicate whether matrix is singular or not
int forwardElim(double mat[N][N+1]);

// function to calculate the values of the unknowns
void backSub(double mat[N][N+1]);

// function to get matrix content
void gaussianElimination(double mat[N][N+1])
```

```
{
        /* reduction into r.e.f. */
        int singular_flag = forwardElim(mat);

        /* if matrix is singular */
        if (singular_flag != -1)
        {
                printf("Singular Matrix.\n");

                /* if the RHS of equation corresponding to
                zero row is 0, * system has infinitely
                many solutions, else inconsistent*/
                if (mat[singular_flag][N])
                        printf("Inconsistent System.");
                else
                        printf("May have infinitely many "
                                "solutions.");

                return;
        }

        /* get solution to system and print it using
        backward substitution */
        backSub(mat);
}

// function for elemntary operation of swapping two rows
void swap_row(double mat[N][N+1], int i, int j)
{
        //printf("Swapped rows %d and %d\n", i, j);

        for (int k=0; k<=N; k++)
        {
                double temp = mat[i][k];
                mat[i][k] = mat[j][k];
                mat[j][k] = temp;
        }
}

// function to print matrix content at any stage
void print(double mat[N][N+1])
{
        for (int i=0; i<N; i++, printf("\n"))
                for (int j=0; j<=N; j++)
                        printf("%lf ", mat[i][j]);

        printf("\n");
}

// function to reduce matrix to r.e.f.
int forwardElim(double mat[N][N+1])
{
        for (int k=0; k<N; k++)
        {
                // Initialize maximum value and index for pivot
                int i_max = k;
                int v_max = mat[i_max][k];

                /* find greater amplitude for pivot if any */
                for (int i = k+1; i < N; i++)
                        if (abs(mat[i][k]) > v_max)
                                v_max = mat[i][k], i_max = i;

                /* if a prinicipal diagonal element is zero,
                * it denotes that matrix is singular, and
                * will lead to a division-by-zero later. */
                if (!mat[k][i_max])
                        return k; // Matrix is singular

                /* Swap the greatest value row with current row */
                if (i_max != k)
                        swap_row(mat, k, i_max);

                for (int i=k+1; i<N; i++)
                {
                        /* factor f to set current row kth elemnt
                           to 0,
                        * and subsequently remaining kth column to
                           0 */
                        double f = mat[i][k]/mat[k][k];

                        /* subtract fth multiple of corresponding
                           kth
                        row element*/
                        for (int j=k+1; j<=N; j++)
                                mat[i][j] -= mat[k][j]*f;

                        /* filling lower triangular matrix with
                           zeros*/
                        mat[i][k] = 0;
                }

                //print(mat);    //for matrix state
        }
        //print(mat);            //for matrix state
        return -1;
}

// function to calculate the values of the unknowns
void backSub(double mat[N][N+1])
{
        double x[N]; // An array to store solution

        /* Start calculating from last equation up to the
        first */
        for (int i = N-1; i >= 0; i--)
        {
                /* start with the RHS of the equation */
                x[i] = mat[i][N];

                /* Initialize j to i+1 since matrix is upper
                triangular*/
                for (int j=i+1; j<N; j++)
                {
                        /* subtract all the lhs values
                        * except the coefficient of the variable
                        * whose value is being calculated */
                        x[i] -= mat[i][j]*x[j];
                }

                /* divide the RHS by the coefficient of the
                unknown being calculated */
                x[i] = x[i]/mat[i][i];
        }

        printf("\nSolution for the system:\n");
        for (int i=0; i<N; i++)
                printf("%lf\n", x[i]);
}

// Driver program
int main()
{
        /* input matrix */
        double mat[N][N+1] = {{3.0, 2.0,-4.0, 3.0},
                                                {2.0, 3.0, 3.0,
                                                    15.0},
                                                {5.0, -3, 1.0,
```

```
                                                    14.0}
                                                };

        gaussianElimination(mat);

        return 0;
}
```

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

10.3 NumberTheory GCD$_LCM$

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

10.4 NumberTheory DivisorsCPP

```
#include <bits/stdc++.h>
using namespace std;
typedef set<int> si;
/* Get the divisors of a number */
si divisores(int n) {
  si d;
  int r = sqrt(n);
  for(int 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 (set<int>::iterator it=divi.begin(); it!=divi.end(); ++it)
    printf("%d ", *it);
  printf("\n");
}
```

10.5 NumberTheory DivisorsPYTHON

```
import math
"""Get the divisors of a number"""
def listDivisors(n):
    divisors = set()
    lim = int(math.sqrt(n))
    for i in range(1, lim + 1):
        if n % i == 0:
            divisors.add(i)
            divisors.add(n // i)
    return divisors
def main():
    d = listDivisors(100)
    print(len(d))
    print(d)
main()
```

10.6 NumberTheory Josephus

```
#include <bits/stdc++.h>
#define show(x) cout << #x << " = "<< x << endl;
using namespace std;
//https://www.youtube.com/watch?v=uCsD3ZGzMgE
int jose(int n, int k) {
  if (n == 1) return 0;
  if (n < k) return (jose(n-1,k)+k)%n;
  int np = n - n/k;
  return k*((jose(np,k)+np-n%k%np)%np) / (k-1);
}

int maxBit(int x){
  for (int i =31; i>=0; i--){
    if(x&(1LL<<i)){
      return i;
    }
  }
  return 0;
}
//always start with soldier 1
int sol(int numSoldiers){
```

```
  int maxr = maxBit(numSoldiers) +1;
  int it = (numSoldiers << 1) -  (1LL<<maxr) +1;
  return it; //soldier that survives
}

int main(){
  int n = 10;
  int res = sol(n);
  show(res);
  return 0;
}
```

10.7 Pow 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;
}
```

10.8 NumberSystems ChangeBases

```
# coding=utf-8
""" CHANGE THE BASE OF A NUMBER
    ob -> origin base
    od -> destiny base
"""

chars = "0123456789ABCDEFGHIJKLMN OPQRSTUVWXYZ"

def changeBase(number, ob,od):
    if ob == 10:
        return tob(number, od)
    return tob(to10(number,ob),od)

""" FROM ANY BASE TO BASE 10
    b -> base of the number n
    pos -> location of a sub-number in n
"""
def to10(n, b, pos =0):
   if n == 0: return 0
   return (n % 10)* (b  ** pos) + to10(n / 10, b, pos+1)

"""FROM TEN BASE TO ANOTHER BASE"""
def tob(n, b):
    if n == 0: return ""
    return   tob(n // b, b) + chars[n % b]

def main():
    print ( tob(7,2))
    print ( tob(252,16))
    print ( tob(234,15))
    print ( to10(1000,2))
    print ( changeBase(111,2,10))
main()
```

10.9 NumberSystems ChangeBases

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

int to10(int n , int b, int pos){
  if (n ==0 ) return 0;
  return ((n % 10)* pow(b,pos)) + to10(n / 10, b, pos+1);
}

string tob(int n, int b){
  if (n == 0) return "";
  return  tob(n / b, b) + chars[n % b];
}
/*
 * ob -> origin base
 * db -> destiny base
 */
string changeBase(int num, int ob, int db){
  if (ob == 10) return tob(num, db);
  return tob(to10(num, ob, 0), db);
}

int main(){
  cout << changeBase(8757,2,16) <<endl;
}
```

11 Sequences

11.1 MatrixFibo

```
#include <iostream>
using namespace std;
typedef long long ll;
ll  *f;

int fib(ll n){
    if (n == 0)  return 0;
    if (n == 1 || n == 2) return (f[n] = 1);
    if (f[n])  return f[n];
    int k = (n & 1)? (n+1)/2 : n/2;
    if (n&1){
      f[n] = (ll) fib(k) * fib(k) + fib(k-1) * fib(k-1) ;
    }else{
      f[n] = (2*fib(k-1) + fib(k))*fib(k);
    }
    return f[n];
}

int main(){
    ll n = 10;
    f = new ll[n];
    cout<< fib(n);
    return 0;
```

```
}
```

12 probability

12.1 ComposedProbability

```
#include <iostream>
#include <vector>
#define forn(i,n) for (int i=0; i<n; i++)
#define s(a) ((int) a.size())

using namespace std;
typedef double d;
typedef vector < d > vd;

/*
 * N --> Number of faces
 * toss -> number of toss
 */
void prob(int N, int t){
    vd  P = {1.0};
    /*
     * RANDOM VARIABLES
     * X,Y
     * Z = X + Y
     *
     * f(Z) = f(X) CONVOLUCION f(Y)
     *
     */
  forn(_, t){
    vd  Pr(s(P) +N -1, 0);
    forn (j, s(P)){
      forn (k, N){
        Pr[j + k ] += P[j] / N;
      }
    }
    P.swap(Pr);
  }
  forn(i, s(P)){
    cout << "P(" << (i+t) <<"):"  << P[i] << endl;
  }
}

int main(){
  prob(6, 1);
  cout << endl;
  prob(6, 2);
  cout << endl;
  prob(6, 3);
}
```

13 Geometry

13.1 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;

}
```

13.2 LineIntersect2

```
#include <bits/stdc++.h>
#define mp make_pair
#define f first
#define s second
using namespace std;
#define show(x) cout << #x << " = " << x << endl;
typedef long double ld;
struct point {
    ld x;
    ld y;
    point (){}
    point (int _x, int _y){
      x = _x;
      y = _y;
    }
};
typedef vector < point > vp;
struct line {
  point o, d;
  line(){}
  line (point _o, point _d){
    o=_o;
    d=_d;
  }
};

pair < bool, point> getLineIntersection(line l1, line l2){
    point p0 =l1.o, p1=l1.d, p2=l2.o, p3=l2.d;
    point AB( p1.x - p0.x, p1.y -p0.y);
    point DC( p3.x - p2.x, p3.y - p2.y);

    ld s, t;
    point i;

    int dx = p0.x - p2.x;
    int dy = p0.y - p2.y;
    s = (-AB.y * dx + AB.x * dy) / (-DC.x * AB.y + AB.x * DC.y);
    t = ( DC.x * dy - DC.y * dx) / (-DC.x * AB.y + AB.x * DC.y);

    if (s >= 0 && s <= 1 && t >= 0 && t <= 1){
        // Collision detected
        i.x = p0.x + (t * AB.x);
        i.y = p0.y + (t * AB.y);
        return mp(true, i);
    }
    return mp(false, i); // No collision
}

int main(){
    line l1(point(0,1),point(2,3));
    line l2(point(3,0),point(0,3));
    pair<bool, point> i = getLineIntersection(l1,l2);
    // intersect x=1, y=2
    if (i.f){
      printf("The lines does collide in: \n");
      show(i.s.x);
      show(i.s.y);
    }else {
      printf("There is no collision.\n");
    }
    return 0;
}
```

13.3 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;
}
```

13.4 EulerFormule

```
#include <stdio.h>
using namespace std;
typedef long long ll;
bool is_a_poligon(ll V, ll E, ll F){
   return V - E + F == 2;
}
int main(){
    printf("%s\n",
            is_a_poligon(3, 34,5) ? "true" : "false");
    //This is a cube
    printf("%s\n",
            is_a_poligon(8,12,6) ? "true" : "false");
    return 0;
}
```

13.5 Line2Point

```
#include<bits/stdc++.h>
#define f first
#define s second
#define mp make_pair
#define magnitude(x) (sqrt(x.f*x.f + x.s*x.s))
#define show(x) cout << #x << " = " << x << endl;
using namespace std;

typedef long double ld;
typedef pair< ld, ld>  point;
```

```
struct line {
  point o, d;
  line(point _o, point _d){
    o = _o;
    d = _d;
  }
};
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;
}
/*
 *Find the  minimun distance from a point to a line
 * just having  two points 'AB' of the line and the point C
 */
ld distance(line l, point C){
    //A, B points in the line
    point A = l.o, B=l.d;
    point AB = diff(A,B); //base
    point AC = diff(A,C);
    ld area = crossProduct(AB, AC);
    ld distance1 = area / magnitude(AB);
    ld distance2 = area / magnitude(AC);
    return min(distance1, distance2);
}
int main(){
    point  A,B,C;
    A = mp(2,4);
    B = mp(5,0);
    C = mp(6,4);
    cout << distance(line(A,B),C);
    return 0;
}
```

13.6 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;
}
```

13.7 ConvexHull

```
#include <bits/stdc++.h>
using namespace std;
struct Point{
    int x, y;
};
Point p0;
Point nextToTop(stack<Point> &S){
    Point p = S.top();
    S.pop();
    Point res = S.top();
    S.push(p);
    return res;
}
int swap(Point &p1, Point &p2){
    Point temp = p1;
    p1 = p2;
    p2 = temp;
}

int distSq(Point p1, Point p2){
    return (p1.x - p2.x)*(p1.x - p2.x) +
          (p1.y - p2.y)*(p1.y - p2.y);
}

int orientation(Point p, Point q, Point r){
    int val = (q.y - p.y) * (r.x - q.x) -
              (q.x - p.x) * (r.y - q.y);
    if (val == 0) return 0;  // colinear
    return (val > 0)? 1: 2; // clock or counterclock wise
}

int compare(const void *vp1, const void *vp2){
   Point *p1 = (Point *)vp1;
   Point *p2 = (Point *)vp2;
   int o = orientation(p0, *p1, *p2);
   if (o == 0)
     return (distSq(p0, *p2) >= distSq(p0, *p1))? -1 : 1;
   return (o == 2)? -1: 1;
}

void convexHull(Point points[], int n){
   int ymin = points[0].y, min = 0;
   for (int i = 1; i < n; i++){
     int y = points[i].y;
     if ((y < ymin) || (ymin == y && points[i].x < points[min].x))
        ymin = points[i].y, min = i;
   }
   swap(points[0], points[min]);
   p0 = points[0];
   qsort(&points[1], n-1, sizeof(Point), compare);
   int m = 1;
   for (int i=1; i<n; i++){
       while (i < n-1 && orientation(p0, points[i], points[i+1])
           == 0){
          i++;
       }
       points[m] = points[i];
```

```
        m++;
    }
    if (m < 3) return;
    stack<Point> S;
    S.push(points[0]);
    S.push(points[1]);
    S.push(points[2]);

    for (int i = 3; i < m; i++){
       while (orientation(nextToTop(S), S.top(), points[i]) != 2)
          S.pop();
       S.push(points[i]);
    }
    while (!S.empty()){
        Point p = S.top();
        cout << "(" << p.x << ", " << p.y <<")" << endl;
        S.pop();
    }
}
int main(){
    Point points[] = {{0, 3}, {1, 1}, {2, 2}, {4, 4},
                      {0, 0}, {1, 2}, {3, 1}, {3, 3}};
    int n = sizeof(points)/sizeof(points[0]);
    convexHull(points, n);
    return 0;
}
```

13.8 CircleCenter

```
#include <bits/stdc++.h>
using namespace std;
const double PI = acos(-1);
#define show(x) cout << #x << " = " << x << endl;
struct point {
    double x;
    double y;
    point (){}
    point (double _x, double _y){
        x = _x;
        y = _y;
    }
};
inline point getCenter(point p1, point p2, point p3){
    point 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(){
  point p1(1,1), p2(2,4), p3(5,3);
  point res = getCenter(p1, p2, p3);
  show(res.x)
  show(res.y)
  return 0;
}
```

14 Arrays

14.1 PermutationPYTHON

```
def permutation(array, start = 0):
    if (start == len(array)):
        print(array)
        return
    for i in range(start, len(array)):
        array[start], array[i] = array[i], array[start]
        permutation(array, start + 1)
        array[start], array[i] = array[i], array[start]
if __name__ == "__main__":
    permutation(['d','a','n'])
```

14.2 Operations

```
from itertools import permutations, combinations

def pers(array):
    ps = permutations(array)
    for p in ps: print (p)

def combs(array, r=2):
    cmb = combinations(array, r)
    for c in cmb: print (c)

if __name__ == "__main__":
    pers([1,2,3,4,5])
    combs([1,2,3,4,5])
```

14.3 MapFunctions

```
"""
    Apply different function over an array
"""
def square(num): return num ** 2
def cube(num): return num ** 3
def is_pair(num): return num % 2
functions = [square, cube, is_pair]
array = [1, 7, -2, 4 , 5,10, 0]
for elemn in array:
    value = map(lambda x: x(elemn), functions)
    print (elemn , end=" :")
    [print (x, end=" ") for x in value if x != None]
    print()
```

14.4 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 get_max_sum(ll data[8], int size){
  ll max_so_far= data[0];
  ll max_ending_here = data[0];
  forn(i, 1, size){
    max_ending_here = max(data[i], \
    data[i] + max_ending_here);
    max_so_far = max(max_so_far, max_ending_here);
  }
  return max_so_far;
}
int main(){
  int size = 8;
  ll data[8] = {-1,2,4,-3,5,2,-5,2};
  ll res = get_max_sum(data, size);
  printf("The max sum that can be done with \n \
    Contiguous elements is: %lld \n", res);
  return 0;
}
```

14.5 Combination

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

def get_combinations(array, r, n):
    combination(array, [0] * r, 0, n-1,  0, r)

if __name__ == "__main__":
    array = [0,1,2,3,4,5]
    get_combinations(array, r=3, n=len(array))
```

14.6 PermutationCPP

```
#include <bits/stdc++.h>
using namespace std;
typedef vector <int > vi;
inline void show(vi &data, int &size){
  for (int i=0; i<size; i++)
      printf("%d \t", data[i]);
  printf("\n");
}

inline void permutation(vi data, int size){
    sort(data.begin(), data.end());
    do {
        show(data, size);
    }while(next_permutation(data.begin(), data.end()));
    show(data, size);
}

int main(){
    int size = 3 ;
    int data[] = {1,4,-1};
    vi vals(begin(data), end(data));
    permutation(vals, size);
    return 0;
}
```

15 Graphs

15.1 Traverse 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;
}
```

15.2 Traverse 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;
        // visit neighbors
        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;
}
```

15.3 BestPath 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;
}
```

15.4 BestPath Dijkstra

```
#include <bits/stdc++.h>
#define numVertices 9
inline  int showSol(int dist[], int n){
  printf("numVerticesertex\tDistance from Source\n");
  for (int i = 0; i < numVertices; i++)
    printf("%d\t%d\n", i, dist[i]);
}
int minDis(int dist[], bool is_set[]){
  int min = INT_MAX, min_index;
  for (int v = 0; v < numVertices; v++){
    if (is_set[v] == false && dist[v] <= min){
      min = dist[v], min_index = v;
    }
  }
  return min_index;
}

inline  void dijkstra(int graph[numVertices][numVertices], int src
    ){
     int dist[numVertices];
     bool is_set[numVertices];
     for (int i = 0; i < numVertices; i++){
        dist[i] = INT_MAX, is_set[i] = false;
     }
     dist[src] = 0;
     for (int count = 0; count < numVertices-1; count++){
       int u = minDis(dist, is_set);
       is_set[u] = true;
       for (int v = 0; v < numVertices; 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];
       }
     }
     showSol(dist, numVertices);
}
int main(){
   int graph[numVertices][numVertices] =
      {{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;
}
```

15.5 BestPath BellmanFord

```
#include <cstdio>
#include <vector>
#define f first
#define s second
#define pb push_back
#define MAX 2e9
using namespace std;

typedef vector<int> vi;
typedef pair<int, int> pii;
typedef vector<pii> vpii;
typedef vector<vpii> vvpii;

void init(vi &distances, int src) {
  for(int i=0; i<distances.size(); i++)
    distances[i] = MAX;
  distances[src] = 0;
}
/*
 *Given a graph and a source vertex src in graph,
 *find shortest paths from src to all vertices in
 *the given graph. The    graph may contain negative weight edges.
*/
void bellmanFord(vvpii &graph, vi &dist) {
  for(int i=0; i<graph.size() - 1; i++) {
    for(int u = 0; u < graph.size(); u++) {
      for(pii v : graph[u]) {
        dist[v.f] = min(dist[v.f], v.s + dist[u]);
      }
    }
  }
}

int main() {
    vvpii adj(5);
    vi d(5);
    int src = 0;
    init(d, src);
    adj[0].pb({1, 6}); adj[0].pb({3, 7});
    adj[1].pb({2, 5}); adj[1].pb({3, 8});
    adj[1].pb({4, -4}); adj[2].pb({1, -2});
    adj[3].pb({2, -3}); adj[3].pb({4, 9});
    adj[4].pb({0, 2}); adj[4].pb({2, 7});
    bellmanFord(adj, d);
    printf("from node= %d\n", src);
    for(int i=0; i<d.size(); i++) {
        printf("to %d = %d \n",i, d[i]);
    }
    printf("\n");
    return 0;
}
```

15.6 BestPath 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 print_sol(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];
        }
      }
    }
    print_sol(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;
}
```