AdaByron 2017 jajaja-va

Numéricos

1. Euclid's algorithm, etc. (C++)

Grafos

1. Dense Dijkstra's (C++)
2. Kruskal's (C++)

Mix

1. Longest Increasing Subsequence (C++)
2. Dates (C++)
3. Knuth-Morris-Pratt (C++)
4. Búsqueda binaria y lineal (C++)
5. QuickSort (C++)
6. FloydWharshall
7. Otras constantes
8. Ficheros leer

Problemas Ada-Byron 2016

1. Double Decker (C++)
2. Cucuruchos (C++)
3. Palmeras en la Nieve (C++)
4. Máquina Calculadora (C++)
5. Primera línea de playa (C++)
6. Duckindromo (C++)
7. El conteo de la Rosa (C++)
8. Alimentando a los pollitos (C++)
9. Teclas del piano (C++)

Otros

1. BackTracking – The KnapSack (C++)
2. Longest Increasing Subsequence (C++)
3. Longest Common Increasing Subsequence (C++)
4. Maximum submatrix (C++)
5. Partitions Integer (C++)
6. Hojas de un árbol balanceado (C++)
7. Arboles, Grafos, Recorridos (C++)

Euclid's algorithm, etc. (C++)

```
#include <iostream>
using namespace std;
int gcd(int a, int b)
{
   while (b > 0)
   {
      int temp = b;
   b = a % b;
   a = temp;
  }
  return a;
}

int lcm(int a, int b)
  return a*(b/gcd(a,b));

int main()
{
  int numero;
  while(1)
  {
    cin>>numero;
    if(numero!=0)
    {
      int array[numero];
      for(int p=0; p<numero;p++)
      {
        cin>>array[p];
      }
      int final=array[0];
      for(int i=1;i<numero;i++)
      {
        final=lcm(final,array[i]);
      }
      cout<<final<<"\n"; }
    else
      break;   }
  return 0;  }
```

Linear systems, matrix inverse (Stanford) (C++)

```
// Gauss-Jordan eliminacion con pivotacion parcial.
//   (1) resolver sistemas de ecuaciones lineales (AX=B)
//   (2) Inversas de madrices (AX=I)
//   (3) Determinantes de matrices O(|N|^3)
// INPUT:    a[][] = an nxn matrix b[][] = an nxm matrix
// OUTPUT:   x[][] = an nxm matrix (stored in b[][]) //          returns
determinant of a[][] const double EPSILON = 1e-7;
typedef vector<double> VD; typedef vector<VD> VVD;
// Gauss-Jordan elimination with partial pivoting
double GaussJordan (VVD &a, VVD &b){   double det = 1;   int i,j,k;
int n = a.size();   int m = b[0].size();   for (k=0;k<n;k++){     j=k;
    for (i=k+1;i<n;i++) if (fabs(a[i][k])>fabs(a[j][k])) j = i;     if
(fabs(a[j][k])<EPSILON){ cerr << "Matrix is singular." << endl; exit(1)
for (i=0;i<n;i++) swap (a[j][i],a[k][i]);     for (i=0;i<m;i++) swap
(b[j][i],b[k][i]);     if (j!=k) det *= -1;
       double s = a[k][k];     for (j=0;j<n;j++) a[k][j] /= s;     for
(j=0;j<m;j++) b[k][j] /= s;     det *= s;     for (i=0;i<n;i++) if (i != k){
double t = a[i][k];       for (j=0;j<n;j++) a[i][j] -= t*a[k][j];       for
(j=0;j<m;j++) b[i][j] -= t*b[k][j];
            }   }   return det; }
```

RREF, matrix rank (C++)

```
// Reducir por el metodo de  Gauss-Jordan eliminacion
// con pivotado. Devuelve el rango de la matriz. // : O(n^3)
// INPUT:    a[][] = an nxn matrix
// OUTPUT:rref[][] = an nxm matrix (stored in a[][]),returns rank of a[][]
const double EPSILON = 1e-7;
typedef vector<double> VD; typedef vector<VD> VVD; // returns rank
int rref (VVD &a){   int i,j,r,c;   int n = a.size();   int m = a[0].size();   for
(r=c=0;c<m;c++){     j=r;     for (i=r+1;i<n;i++) if
(fabs(a[i][c])>fabs(a[j][c])) j = i;     if (fabs(a[j][c])<EPSILON) continue;
for (i=0;i<m;i++) swap (a[j][i],a[r][i]);
       double s = a[r][c];     for (j=0;j<m;j++) a[r][j] /= s;     for
(i=0;i<n;i++) if (i != r){       double t = a[i][c];
     for (j=0;j<m;j++) a[i][j] -= t*a[r][j];
           }     r++;
         }   return r; }
```

Dense Dijkstra's (C++)

```
void Dijkstra (const VVT &w, VT &dist, VI &prev, int start){   int n = w.size();
VI found (n);   prev = VI(n, -1);   dist = VT(n, 1000000000);   dist[start] = 0;
while (start != -1){     found[start] = true;     int best = -1;     for (int k = 0; k <
n; k++) if (!found[k]){       if (dist[k] > dist[start] + w[start][k]){ dist[k] =
dist[start] + w[start][k]; prev[k] = start;       }
      if (best == -1 || dist[k] < dist[best]) best = k;
   }     start = best;
         } }
```

Kruskal's (C++)

```
/*
El algoritmo de Kruskal calcula el tamaño minimo de un bosque es decir
union de arboles cada uno conectado a una componente, posibilitando una
matriz de adyacencia
dada una matriz con peso en los nodos donde -1 es si no existe el vertice.
Devuelve el minimo peso del bosque calculando los vertices guardados en
T, usa un arbol disjunto para
amortizar la efectividad en un tiempo contante siendo la complejidad
O(E*log(E))
*/ typedef int TYPE;
struct edge {  int u, v;
   TYPE d; };
struct edgeCmp {
   int operator()(const edge& a, const edge& b) { return a.d > b.d; }
}; int find(vector <int>& C, int x) { return (C[x] == x) ? x : C[x] = find(C, C[x]);
TYPE Kruskal(vector <vector <TYPE> >& w)
{     int n = w.size();
    TYPE weight = 0;          vector <int> C(n), R(n);
    for(int i=0; i<n; i++) { C[i] = i; R[i] = 0; }
        vector <edge> T;
    priority_queue <edge, vector <edge>, edgeCmp> E;
        for(int i=0; i<n; i++)          for(int j=i+1; j<n; j++)     if(w[i][j] >= 0)
{                edge e;
              e.u = i; e.v = j; e.d = w[i][j];
              E.push(e);
          }
```

```
    while(T.size() < n-1 && !E.empty())
    {
        edge cur = E.top(); E.pop();

        int uc = find(C, cur.u), vc = find(C, cur.v);          if(uc != vc)
        {
            T.push_back(cur); weight += cur.d;
            if(R[uc] > R[vc]) C[vc] = uc;              else if(R[vc] > R[uc])
C[uc] = vc;              else { C[vc] = uc; R[uc]++; }
        }
            }
return
weight;
        }
```

Longest Increasing Subsequence (C++)

```
//Dada una lista de numberos de longitud n, extrae a que es la mayor
subsecuencia de aumento O(nlogn)
//   INPUT: a vector of integers // Posible solucion
X= XMJYAUZ Y=MZJAWXU|| LCS= MJAU
//   OUTPUT: a vector containing the longest increasing subsequence
int dp[1001][1001];
  int lcs(const string &s, const string &t)
  {
  int m = s.size(), n = t.size();
  if (m == 0 || n == 0) return 0;
  for (int i=0; i<=m; ++i)
  dp[i][0] = 0;
  for (int j=1; j<=n; ++j)
  dp[0][j] = 0;
  for (int i=0; i<m; ++i)
  for (int j=0; j<n; ++j)
  if (s[i] == t[j])
  dp[i+1][j+1] = dp[i][j]+1;
  else
  dp[i+1][j+1] = max(dp[i+1][j], dp[i][j+1]);
  return dp[m][n];
        }
```

Dates (C++) Convertir de Georgiano a Juliano

```
// Cambio de fechas los meses son expresados como enteros de 1 a 12 y
los dias de 1 a 31.
string dayOfWeek[] = {"Mo", "Tu", "We", "Th", "Fr", "Sa", "Su"}; // converts
Gregorian date to integer (Julian day number)
int DateToInt (int m, int d, int y){
   return 1461 * (y + 4800 + (m - 14) / 12) / 4 + 367 * (m - 2 - (m - 14) / 12 *
12) / 12 -    3 * ((y + 4900 + (m - 14) / 12) / 100) / 4 +    d - 32075; }
// converts integer (Julian day number) to Gregorian date: month/day/year
void IntToDate (int jd, int &m, int &d, int &y){   int x, n, i, j;    x = jd + 68569;
n = 4 * x / 146097;   x -= (146097 * n + 3) / 4;   i = (4000 * (x + 1)) / 1461001;
x -= 1461 * i / 4 - 31;   j = 80 * x / 2447;   d = x - 2447 * j / 80;   x = j / 11;   m
= j + 2 - 12 * x;   y = 100 * (n - 49) + i + x; }
// converts integer (Julian day number) to day of week
        string IntToDay (int jd){   return dayOfWeek[jd % 7]; }
```

Knuth-Morris-Pratt (C++) (KMP)

```
// Busca un string w en s de una determinada longitud, devuelve el indice del
primer encuentro, k si no lo encuentra y es O(k)
void build(string& w, vector <int>& t)
{   t = vector <int>(w.length());   int i = 2, j = 0;   t[0] = -1; t[1] = 0;
  while(i < w.length()) {    if(w[i-1] == w[j]) { t[i] = j+1; i++; j++; }    else if(j > 0) j =
t[j];    else { t[i] = 0; i++; }
} }
int KMP(string& s, string& w)
{   int m = 0, i = 0;   vector <int> t;   build(w, t);
  while(m+i < s.length()) {    if(w[i] == s[m+i]) {      i++;
    if(i == w.length()) return m;
          } else {      m += i-t[i];
        if(i > 0) i = t[i];    }  }
        return s.length(); }
```

```cpp
//Método mio
void kmp(const string &needle, const string &haystack) {
  int m = needle.size();
  vector<int> border(m + 1);
  border[0] = -1;
  for (int i = 0; i < m; ++i) {
    border[i+1] = border[i];
    while (border[i+1] > -1 and needle[border[i+1]] != needle[i]) {
      border[i+1] = border[border[i+1]];
    }
    border[i+1]++;
  }
  int n = haystack.size();
  int seen = 0;
  for (int i = 0; i < n; ++i){
    while (seen > -1 and needle[seen] != haystack[i]) {
      seen = border[seen];
    }
    if (++seen == m) {
      printf("%d\n", i - m + 1);
    seen = border[m];
    }
  }
}
```

Búsqueda Lineal – Búsqueda Binaria

```cpp
#include <iostream>
#include <stdlib.h>
using namespace std;

int lineal_search(int *array, int searched, int arraySize)
{
    for (int i = 0; i< arraySize; i++) {
        if (searched == array[i]) {
        }
   }
    return 0;
}
int binary_search(int *array, int searched, int arraySize)
{
    int first = 0, middle, last = arraySize - 1;
    while (first <= last) {
        middle = (first + last) / 2;

        if (searched == array[middle]) {
                return array[middle];
        } else {
            if (array[middle] > searched) {
                last = middle - 1;
            } else {
                first = middle + 1;
            }
        }
    }
    return -1;
}
int main()
{
    int arraySize, searched;
    cin >> arraySize;
    int array[arraySize];
    cin >> searched;
    lineal_search(array, searched, arraySize);
    binary_search(array, searched, arraySize);
            return 0; }
```

QuickSort

```
#include <stdio.h>
#include <stdlib.h>
int values[] = { 40, 10, 100, 90, 20, 25 };

int compare (const void * a, const void * b)
{
  return ( *(int*)a - *(int*)b );
}

int main ()
{
  int n;
  qsort (values, 6, sizeof(int), compare);
  for (n=0; n<6; n++)
     printf ("%d ",values[n]);
  return 0;
}
```

FloydWarshall

```
#include<stdio.h>
#define V 4
#define INF 99999
void printSolution(int dist[][V]);
void floydWarshall (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];
         }
      }
   } //Solucion en dist
}


int main()
{
   int graph[V][V] = { {0,   5,  INF, 10},
                       {INF, 0,   3, INF},
                       {INF, INF, 0,   1},
                       {INF, INF, INF, 0}
                     };
   floydWarshall(graph);
   return 0;
        }
```

Otras constantes

PI: 4 * atan(1)

Ficheros leer

```
//Cantidad de Enteros, Char
        if(scanf("%ld",&casos)!=
        EOF)
//Linea de fichero
        while(getline(cin, str))
        o if (!cin.eof())
```

Problemas AdaByron 2016 – Resueltos – Double Decker

```
#include <bits/stdc++.h>
using namespace std;
int main() {
	int testcase;
	scanf("%d", &testcase);
	while (testcase--) {
			long long N, M;
			scanf("%lld %lld", &N, &M);
			long long S = N + M, ret;
			ret = (S*(S+1)/2) + N + 1;
			printf("%lld\n", ret);
	}
	return 0; }
```

Problemas AdaByron 2016 – Resueltos – Cucuruchos

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

int main() {
   int casos;
   scanf("%d", &casos);
   while (casos--) {
      int C, V, A[16] = {};
      char s[16], mm[3] = "CV";
      scanf("%d %d", &C, &V);
      for (int i = C; i < C+V; i++)
         A[i] = 1;
      int f = 0;
      do {
         for (int i = 0; i < C+V; i++)
            s[i] = mm[A[i]];
         s[C+V] = '\0';
         if (f)    putchar(' ');
         printf("%s", s), f = 1;
      } while (next_permutation(A, A+C+V));
      puts("");
   }
   return 0; }
```

Problemas AdaByron 2016 – Resueltos – Palmeras en la nieve

```
#include <iostream>
using namespace std;
int main() {
   int casos;
   cin >> casos;
   while (casos--) {
      int nieve, arboles;
      cin >> nieve >> arboles;
      int p[arboles];
      int i = 0, j = 0, l = 0, enpie = 0;

         while (arboles--) {
         cin >> p[j];
         if (p[j] >= nieve && ++enpie > 5)
            while (enpie > 5) {
               if (p[i] >= nieve)
                  enpie--;
               i++;
            }
         l = max(l, j-i+1);
         j++;
      }
      cout << l << endl;
   }
   return 0;
      }
```

Problemas AdaByron 2016 – Resueltos – Máquina Calculadora

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

int main() {
    int st, ed;
    while (scanf("%d %d", &st, &ed) == 2) {
        int used[32767] = {}, u, x;
        queue<int> Q;
        Q.push(st), used[st] = 1;
        while (!Q.empty()) {
            u = Q.front();
            Q.pop();
            if (used[ed])      break;
            x = (u + 1)%10000;
            if (used[x] == 0) {
                used[x] = used[u] + 1;
                Q.push(x);
            }
            x = (u * 2 + 10000)%10000;
            if (used[x] == 0) {
                used[x] = used[u] + 1;
                Q.push(x);
            }
            x = (u / 3);
            if (used[x] == 0) {
                used[x] = used[u] + 1;
                Q.push(x);
            }
        }
        printf("%d\n", used[ed] - 1);
    }
    return 0;
        }
```

Problemas AdaByron 2016 – Resueltos – Primera línea de Playa

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

int main() {
    int N;
    while (scanf("%d", &N) == 1 && N) {
        vector<pair<int, int>> A;
        int L, R;
        for (int i = 0; i < N; i++) {
            scanf("%d %d", &L, &R);
            A.push_back(make_pair(L, R));
        }
        sort(A.begin(), A.end());

        int ret = 0;
        int PQ = INT_MAX;
        for (int i = 0; i < N; i++) {
            int line_x = A[i].first;
            if (PQ != INT_MAX && PQ <= line_x) {
                ret++;
                PQ = INT_MAX;
            }
            PQ = min(PQ, A[i].second);
        }
        if (PQ != INT_MAX)
            ret++;
        printf("%d\n", ret);
    }
    return 0;
        }
```

Problemas AdaByron 2016 – Resueltos – Duckindromo

```
#include <bits/stdc++.h>
int memo[1000][1000];
string s;

int patindrome(int i, int j) {
    if (i > j) return 0;
    if (memo[i][j] != -1) return memo[i][j];
    if (i == j) return memo[i][j] = 1;

    if (s[i] == s[j])
        return memo[i][j] = 2 + patindrome(i+1, j-1);
    else
        return memo[i][j] = max(patindrome(i+1, j), patindrome(i, j-1));
}
int main() {
    while (cin >> s) {
        for (int i = 0; i < s.length(); i++)
            for (int j = 0; j < s.length(); j++)
                memo[i][j] = -1;
        patindrome(0, s.length()-1);

        stack<char> b;
        int i = 0, j = s.length()-1;
        while (i <= j) {
            if (s[i] == s[j]) {
                cout << s[i];
                if (i != j) b.push(s[i]);
                i++, j--;
            } else if (memo[i][j] == memo[i+1][j]) i++; // eliminar patito
    izquierdo primero
            else j--;
        }
        while (!b.empty()) cout << b.top(), b.pop();
        cout << endl;
    }

return 0;}
```

Problemas AdaByron 2016 – Resueltos –El Conteo de la Rosa

```
#include <bits/stdc++.h>
using namespace std;
int digits(int i) { // i > 0
    int cnt = 0;
    while (i > 0) {
        cnt++;
        i /= 10;
    }
    return cnt;
}
int ndigits(int n) { // number of digits of the numbers [1...n]
    if (n < 1) return 0;
    int d = digits(n);
    if (d == 1) return n;
    int p = 10, s = 1;
    for (int i = 1; i <= d-2; i++) {
        s += p * (i+1);
        p *= 10;
    }
    return 9*s + d * (n-p+1);
}
int a(int x, int y) { // number of digits of the numbers [x...y], 1 <= x <= y
    return ndigits(y) - ndigits(x-1);
}
int main() {
    int p1, p2;
    while (cin >> p1 >> p2 && !(p1 == 0 && p2 == 0)) {
        int i = p1, j = p2, mid = a(p1, p2) / 2, m;
        while (i <= j) {
            m = (i + j) / 2;
            int d1 = a(p1, m);
            if (d1 == mid) break;
            if (d1 < mid) i = m + 1;
            else j = m - 1;
        }
        if (a(p1, m) > a(m+1, p2)) m--;
        cout << m << endl;   }

    return 0; }
```

Problemas AdaByron 2016 – Resueltos –Alimentando a los pollitos

```
#include <bits/stdc++.h>
using namespace std;
int G[128][128];
int N, M, Q;
const char dir[] = "NESW";
const int dx[] = {-1, 0, 1, 0};
const int dy[] = {0, 1, 0, -1};
void draw(int sx, int sy, char sd[], int sv) {
        int d = 0;
        for (int i = 0; i < 4; i++) {
                if (sd[0] == dir[i])
                        d = i;
        }
        G[sx][sy]++;
        for (int i = 1; ; i++) {
                for (int j = 0; j < 2; j++) {
                        for (int k = 0; k < i; k++) {
                                sx += dx[d], sy += dy[d];
                        if (sx > N || sy > M || sx <= 0 || sy <= 0)
                                        return ;
                                G[sx][sy]++;
                                sv--;
                                if (sv == 0)            return ;
                        }
                        d = (d + 1)%4;
                }
        }
}
int main() {
        int testcase;
        scanf("%d", &testcase);
        while (testcase--) {
                scanf("%d %d %d", &N, &M, &Q);
                memset(G, 0, sizeof(G));
                for (int i = 0; i < Q; i++) {
                        char s[128];
                        int x, y, v;
                        scanf("%d %d %s %d", &x, &y, s, &v);
                        draw(x, y, s, v);
                }
                for (int i = 1; i <= N; i++) {
                        for (int j = 1; j <= M; j++)
                        printf("%d%c", G[i][j], j == M ? '\n' : ' ');
                }
                puts("---");
        }
        return 0;
}
```

Problemas AdaByron 2016 – Resueltos –Teorema del punto

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

long long gcd(long long x, long long y) {
   long long t;
   while (x%y)
      t = x, x = y, y = t%y;
   return y;
}
long long lcm(long long x, long long y)
   return x / gcd(x, y) * y;
int main() {
   int n;
   while (scanf("%d", &n) == 1 && n) {
      int A[128] = {}, used[128] = {};
      for (int i = 1; i <= n; i++)
         scanf("%d", &A[i]);
      long long ret = 1;
      for (int i = 1; i <= n; i++) {
         if (used[i])    continue;
         int u = i, cc = 0;
         while (!used[u])
            used[u] = 1, cc++, u = A[u];
         ret = lcm(ret, cc); }
      printf("%lld\n", ret);  }
   return 0; }
```

Problemas AdaByron 2016 – Resueltos – Teclas del piano

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

const char s[][16] = {
    "Dob", "Do", "Do#", "Reb", "Re", "Re#", "Mib", "Mi", "Fab",
    "Mi#", "Fa", "Fa#", "Solb", "Sol", "Sol#", "Lab", "La",
    "La#", "Sib", "Si", "Si#"};
const int w[] = { -1, 0, 1, 1, 2, 3, 3, 4, 4, 5, 5, 6, 6, 7, 8, 8, 9,
    10, 10, 11, 12};
int main() {
    int m = sizeof(s) / sizeof(s[0]);
    map<string, int> R;
    for (int i = 1; i <= 7; i++) {
    int base = i * 12;
    for (int j = 0; j < m; j++) {
    char buf[16];
    sprintf(buf, "%s%d", s[j], i);
    R[buf] = base + w[j];
                } }
    int n;
    while (scanf("%d", &n) == 1 && n) {
                map<int, int> ret;
                for (int i = 0; i < n; i++) {
                        char buf[16];
                        scanf("%s", buf);
                        ret[R[buf]]++;
                }
                int prev = -1;
    for (auto &e : ret) {
    if (prev == -1) {
    printf("%d", e.second), prev = e.first+1;
    } else {
                for (; prev < e.first; prev++)
                printf(" 0");
    printf(" %d", e.second), prev = e.first+1;
                                } }
                puts("");
    }
    return 0;}
```

BackTracking – The KnapSack

```
#include<iostream>
using namespace std;
int f[1000]={0};
int n=0, m=0;
int main(void)
{
cin >> n >> m;
for (int i=1;i<=n;i++)
{
int price=0, value=0;
cin >> price >> value;
for (int j=m;j>=price;j--)
if (f[j-price]+value>f[j])
f[j]=f[j-price]+value;
}
cout << f[m] << endl;
return 0;
}
```

Longest Increasing Subsequence

```
// 0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15 a longest increasing
subsequence is 0, 2, 6, 9, 11, 15
#include<iostream>
using namespace std;
int n=0;
int a[100]={0}, f[100]={0}, x[100]={0};
int main(void)
{
cin >> n;
for (int i=1;i<=n;i++)
{
cin >> a[i];
x[i]=INT_MAX;
}
f[0]=0;
int ans=0;
for(int i=1;i<=n;i++)
{
```

```
    int l=0, r=i;
    while (l+1<r)
{
int m=(l+r)/2;
if (x[m]<a[i]) l=m; else r=m;
// change to x[m]<=a[i] for non-decreasing case
}
f[i]=l+1;
x[l+1]=a[i];
if (f[i]>ans) ans=f[i];
}
cout << ans << endl;
return 0;
        }
```

Longest Common Increasing Subsequence

```
// 2 3 1 6 5 4 6 AND 1 3 5 6 the LCIS is 3 5 6.
#include<iostream>
using namespace std;
int a[100]={0};
int b[100]={0};
int f[100]={0};
int n=0, m=0;
int main(void)
{
cin >> n;
for (int i=1;i<=n;i++) cin >> a[i];
cin >> m;
for (int i=1;i<=m;i++) cin >> b[i];
for (int i=1;i<=n;i++)
{
int k=0;
for (int j=1;j<=m;j++) {
if (a[i]>b[j] && f[j]>k) k=f[j];
else if (a[i]==b[j] && k+1>f[j]) f[j]=k+1;
} }
int ans=0;
for (int i=1;i<=m;i++)
if (f[i]>ans) ans=f[i];
cout << ans << endl;
return 0;}
```

Maximum submatrix

```
#include<iostream>
using namespace std;
int a[150][150]={0};
int c[200]={0};
int maxarray(int n) {
int b=0, sum=-100000000;
for (int i=1;i<=n;i++)
{
if (b>0) b+=c[i];
else b=c[i];
if (b>sum) sum=b;
}
return sum;
}
int maxmatrix(int n)
{
int sum=-100000000, max=0;
for (int i=1;i<=n;i++)
{
for (int j=1;j<=n;j++)
c[j]=0;
for (int j=i;j<=n;j++)
{
for (int k=1;k<=n;k++)
c[k]+=a[j][k];
max=maxarray(n);
if (max>sum) sum=max;
} }
return sum;
}
int main(void) {
int n=0;
cin >> n;
for (int i=1;i<=n;i++)
for (int j=1;j<=n;j++)
cin >> a[i][j];
cout << maxmatrix(n);
return 0;}
```

Partitions Integer

```
//4 - 3 + 1 - 2 + 2 -
2 + 1 + 1 - 1 + 1 +
1 + 1
#define MAXN 100 // largest n or m
long int_coefficient(n,k) // compute f(n,k)
int n,m; {
int i,j;
long f[[MAXN][MAXN];
f [1][1] = 1;
for (i=0;i<=n;i++) f[i][0] = 0;
for (i=1; i<=n; i++)
for (j=1; j<i; j++)
if (i-j <= 0)
f[i][j] = f[i][k-1];
else
f[i][j] = f[i-j][k]+f[i][k-1];
return f[n][k];
        }
```

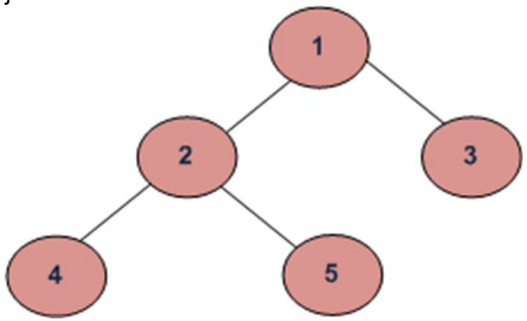
Hojas de un árbol balanceado

```
#include <iostream>
using namespace std;
int main() {
    int n=1;
    long long int sum=0;
    while(n!=0)
    {
        cin >> n;
        int arr[n];
        arr[0]=n;
        if(n!=0) {
        for(int i = 1 ; i <= n ; i++)
        {
            cin >> arr[i];
            if((2 * i) > n)
                sum += arr[i]; }
  cout<<sum<<"\n";
  sum=0; } }
return 0; }
```

Árbol recorrido PreOrden – InOrden - PostOrden

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
    int data;
    struct node* left;
    struct node* right;
};

struct node* newNode(int data)
{
    struct node* node = (struct node*)
    malloc(sizeof(struct node));
    node->data = data;
    node->left = NULL;
    node->right = NULL;

    return(node);
}

void printPostorder(struct node* node)
{
    if (node == NULL)
      return;
    printPostorder(node->left);
    printPostorder(node->right);
    printf("%d ", node->data);
}
void printInorder(struct node* node)
{
    if (node == NULL)
        return;
    printInorder(node->left);
    printf("%d ", node->data);
    printInorder(node->right);
}
```

```
void printPreorder(struct node* node)
{
    if (node == NULL)
        return;
    printf("%d ", node->data);
    printPreorder(node->left);
    printPreorder(node->right);
}
int main()
{
    struct node *root  = newNode(1);
    root->left        = newNode(2);
    root->right       = newNode(3);
    root->left->left   = newNode(4);
    root->left->right  = newNode(5);
    printf("\nPreorder traversal of binary tree is \n");
    printPreorder(root);
    printf("\nInorder traversal of binary tree is \n");
    printInorder(root);
    printf("\nPostorder traversal of binary tree is \n");
    printPostorder(root);

    getchar();
    return 0;
}
```

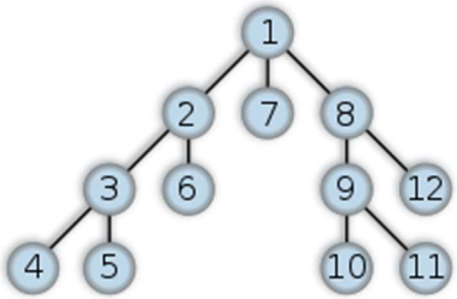



```
Preorder traversal//BFS:  1 2 4 5 3
Inorder traversal:   4 2 5 1 3
Postorder traversal: 4 5 2 3 1
```

Grafo recorrido DFS y BFS

```
#include <bits/stdc++.h>
#define oo 1005

using namespace std;

int N, A, K, D[oo]; ///N: Cant. de Nodo A: Cant. de Aristas K: Cant. de Intrucciones
bool Mk[oo];
vector<int> V[oo];
queue<int> Q;
char S;             ///Caracter Separador

void DFS ()
{
   int nodo = 0, newNod = 0, ady = 0;

   Q.push(0);

   while(!Q.empty())
   {
      nodo = Q.front();
      Q.pop();

      vector<int>::iterator i;

      for(i = V[nodo].begin(); i != V[nodo].end(); i++)
      {
         ady = *i;
         if(Mk[ady])
            continue;
         Mk[ady] = true;
         D[ady] = D[nodo] + 1;
         Q.push(ady);
      }
   }
}

int main ()
{
   freopen("DFS.in", "r", stdin);
```

```
    freopen("DFS.out", "w", stdout);

    int C_1, C_2;

    cin >> N >> A >> K;

    for(int i = 0; i < A; i++)
    {
        cin >> C_1 >> C_2;

        V[C_1].push_back(C_2);
        V[C_2].push_back(C_1);
    }

    D[0] = 0;
    Mk[0] = true;

    DFS();

    for(int i = 0; i < K; i++)
    {
        cin >> C_1;
        cout << D[C_1] << "\n";
    }

    return 0;
}
```

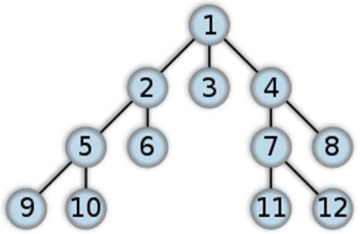


BFS

Matriz Recorrido DFS

```
#include <bits/stdc++.h>
#define oo 1005

using namespace std;

struct two
{
    int f, c;

    two(int a = 0, int b = 0)
    {
        f = a;
        c = b;
    }
};

const int Mf [] = {1, -1, 0, 0};
const int Mc [] = {0, 0, 1, -1};

int N, M, CA[oo][oo];
bool Mk[oo][oo];
queue<two> Q;

bool isPossible (int f, int c)  ///Saber si es posible el movimiento hacia esa casilla
{
    if(f < 0 || f > N - 1 || c < 0 || c > M - 1 || Mk[f][c])
        return false;
    return true;
}

void DFS ()
{
    int F, C;

    while(!Q.empty())
    {
        F = Q.front().f;
        C = Q.front().c;
```

```
        Q.pop();

        for(int i = 0; i < 4; i++)
        {
            int nf = F + Mf[i];
            int nc = C + Mc[i];

            if(isPossible(nf, nc))
            {
                CA[nf][nc] = CA[F][C] + 1;
                Mk[nf][nc] = true;
                Q.push(two (nf, nc));
            } }  } }

int main ()
{
    freopen("DFS.in", "r", stdin);
    freopen("DFS.out", "w", stdout);

    int X = 0;
    two _s, _e; ///Punto de Inicio

    cin >> N >> M;


    for(int i = 0; i < N; i++)
    {
        for(int j = 0; j < M; j++)
        {
            scanf("%s", &X); ///Leer como caracter pero asignar a numero
            if(X == 83) ///Inicio Letra - S
            {
                _s.f = i;
                _s.c = j;
                continue;
            }
            if(X == 69) ///Final Letra - E
            {
                _e.f = i;
                _e.c = j;
                continue;
            }
            if(X == 1)  ///Rocas
            {
                Mk[i][j] = true;
                continue;
            } }  }
    Q.push(two (_s.f, _s.c));
    CA[0][0] = 0;
    Mk[0][0] = true;
    DFS();
    printf("%d\n", CA[_e.f][_e.c]);
    return 0;}
```



DFS

```
#include <bits/stdc++.h>
using namespace std;
void permutations() {
        int C, V, A[16] = {};
        char s[16], mm[3] = "AB";
        scanf("%d %d", &C, &V);
        for (int i = C; i < C+V; i++)
            A[i] = 1;
        int f = 0;
        do {
            for (int i = 0; i < C+V; i++)
                s[i] = mm[A[i]];
            s[C+V] = '\0';
            if (f)     putchar(' ');
            printf("%s", s), f = 1;
        } while (next_permutation(A, A+C+V));
        puts("");
    return 0; }
```