Índice general

Capítulo 1

Ad-hoc

1.1. Invertir Bytes

```
// Entrada: Un entero sin signo de 32 bits
// Salida : El mismo numero pero con los bytes invertidos como si fuera una
    cadena
int reverse_bytes(unsigned int i) {
    return ((i >> 24) ) |
         ((i >> 8) & 0xFF00) |
         ((i << 8) & 0xFF0000) |
         ((i << 24));
}
```

1.2. Año bisiesto

```
bool esBisiesto(int y){
    return ( ((y%4)==0 && (y%100)!=0) || (y%400)==0 );
}
```

1.3. Doomsday

```
//Entrada: Una fecha (ene = 1, ..., dic = 12) Salida: Dia de la semana
string vec[7] = {"DOM","LUN","MAR","MIE","JUE","VIE","SAB"};
string doomsday(int dia, int mes, int anio){
    int a, y, m, d;
    a = (14 - mes) / 12;
    y = anio - a;
    m = mes + 12 * a - 2;
    d = (dia + y + y/4 - y/100 + y/400 + (31*m)/12) % 7;
    return vec[d];
}
```

1.4. Hanoi

```
int hanoi(int n){
    if(n==1) return 1;
    else return 2 * hanoi(n-1) + 1;
}
```

Capítulo 2

Estructuras

2.1. Union - Find

```
#define MAX 100010

using namespace std;

int padre[ MAX ];
int rango[ MAX ];

void MakeSet( int n ){for(int i = 0 ; i <= n ; ++i )padre[ i ] = i, rango[ i ] =
    0;}
int Find(int x){return (padre[x]==x?x:padre[x]=Find(padre[x]));}
bool sameComponent(int x, int y){ return Find(x) == Find(y);}

void Union( int x , int y ){
    int xRoot = Find( x );
    int yRoot = Find( y );
    if( rango[ xRoot ] > rango[ yRoot ] )
        padre[ yRoot ] = xRoot;
    else{
        padre[ xRoot ] = yRoot;
        if( rango[ xRoot ] == rango[ yRoot ] )
            rango[ yRoot ]++;
    }
}
```

2.2. BIT

```
#include <iostream>
#include <cstring>

using namespace std;

#define MAXN 1000000

int T[MAXN + 1];
int A[MAXN];
int N;
int lowbit(int i) {
    return (i & -i);
}
```

```
int sum(int i){
    int value = 0;
    for(; i > 0; i-= lowbit(i))
        value+= T[i];
    return value;
}
int sum(int i, int j){
    return i > 1 ? sum(j) - sum(i-1) : sum(j);
}
void update(int i, int value){
    for(; i <= N ; i += lowbit(i))
        T[i] += value;
}
void build(){
    memset(T, 0, sizeof(T));
    for(int i=0; i < N; i++)
        update(i+1, A[i]);
}
int main(){
    cin >> N;
    for(int i = 0; i < N; ++i)
        cin >> A[i];

    build();
    cout << sum( 1, N ) << endl;
    return 0;
}
```

2.3. Segment Tree

```
#include <ctime>
#include <iostream>
#include <cmath>
#include <cstdio>
using namespace std;
int c[100000];
int tree[400001];
//si es segment tree de -,* o / solo sustituir el + en init query y update
void init(int node,int a,int b)
{
    if(a==b)
    {
        tree[node]=c[a];
        return ;
    }
    init(2*node+1,a,(a+b)/2);
    init(2*node+2,(a+b)/2+1,b);
    tree[node]=tree[2*node+1]+tree[2*node+2];
}
// consula para llamar query(0,0,n-1,desde,hasta)
int query(int node,int a,int b,int p,int q)
{
    //la consulta se hace en el rango desde p a q, a y b son los limites del
       rango
    if( q<a || b<p )return 0;
    if(p<=a && b<=q)
    {
```

```
27          return tree[node];
28      }
29      return query(2*node+1,a,(a+b)/2,p,q)+query(2*node+2,(a+b)/2+1,b,p,q);
30  }
31  //sustituir para llamar(0,0,n-1,posicion,valor)
32  void update(int node,int a,int b,int p,int val)
33  {
34      if(p<a || b<p)return;
35      if(a==b)
36      {
37          tree[node]=val;
38          return ;
39      }
40      update(2*node+1,a,(a+b)/2,p,val);
41      update(2*node+2,(a+b)/2+1,b,p,val);
42      tree[node]=tree[2*node+1]+tree[2*node+2];
43  }
44
45  int main()
46  {
47      int n,aux;
48      for(int i=0; i<n; i++)
49      {
50          scanf("%d",&aux);
51          c[i]=aux;
52      }
53      init(0,0,n-1);
54      //ejemplo de sustitucion tree[a]=val
55      int a,b;
56      int val;
57      scanf("%d %d",&a,&val);
58      a--;//solo si los subindices del problema van de 1...n
59      update(0,0,n-1,a,val);
60      //ejemplo de consulta x=SUM(a,a+1,....,b)
61      scanf("%d %d",&a,&b);
62      a--;//solo si los subindices del problema van de 1...n
63      b--;//solo si los subindices del problema van de 1...n
64      int x=query(0,0,n-1,a,b);
65      printf("%d\n",x);
66      printf("\n");
67      return 0;
68  }
```

Otra implementacion en java:

```
1   public class SegmentTree{
2
3       static final int NEUT = 2147483647;// Cambiar segun operacion
4
5       public static void main(String[] args){
6           int[] m = {4,0,7,1,3,5,8,8,3,2,2,1};
7           int[] rmq = new int[m.length*4];
8           rmq_init(1, 0, m.length, rmq, m);
9           System.out.println(rmq_query(1, 0, m.length, rmq, 2, 5));
10      }
11      public static int LEFT(int n){ return 2*n; }
12      public static int RIGHT(int n){ return 2*n+1; }
13      public static int oper(int a, int b){ return Math.min(a, b); }
14
```

```
15      public static void rmq_init(int n, int s, int e, int[] rmq, int[] m){
16          if(s+1 == e)
17              rmq[n] = m[s];
18          else{
19              rmq_init(LEFT(n), s, (s+e)/2, rmq, m);
20              rmq_init(RIGHT(n), (s+e)/2, e, rmq, m);
21              rmq[n] = oper(rmq[LEFT(n)], rmq[RIGHT(n)]);
22          }
23      }
24      public static void rmq_update(int n, int s, int e, int[] rmq, int[] m, int p
            , int v){
25          if(s+1 == e)
26              rmq[n] = m[s] = v;
27          else{
28              if(p < (s+e)/2)
29                  rmq_update(LEFT(n), s, (s+2)/2, rmq, m, p, v);
30              else
31                  rmq_update(RIGHT(n), (s+e)/2, e, rmq, m , p, v);
32              rmq[n] = oper(rmq[LEFT(n)], rmq[RIGHT(n)]);
33          }
34      }
35      public static int rmq_query(int n, int s, int e, int[] rmq, int a, int b){
36          if(a >= e || b <= s)
37              return NEUT;
38          else if (s >= a && e <= b)
39              return rmq[n];
40          else{
41              int l = rmq_query(LEFT(n), s, (s+e)/2, rmq, a, b);
42              int r = rmq_query(RIGHT(n), (s+e)/2, e, rmq, a, b);
43              return oper(l, r);
44          }
45      }
46  }
```

2.4. Segment Tree con Lazy Propagatión

```
1   #include <iostream>
2   #include <cstdio>
3   #include <algorithm>
4   #include <cstring>
5   using namespace std;
6
7   struct Node{
8       long long sum;  //suma de hijos
9       long long offset;//suma que no propaga de total nodo
10  }Tree[500000];
11
12  void Update(long long node,long long lo,long long hi,long long i,long long j,
        long long val){
13      if(lo>j || hi<i)
14          return;
15
16      if(lo>=i && hi<=j){
17          Tree[node].sum+=(hi-lo+1)*val;
18          Tree[node].offset+=val;
19      }else{
20          long long mid=(lo+hi)>>1;
```

```
21          Update(2*node,lo,mid,i,j,val);
22          Update(2*node+1,mid+1,hi,i,j,val);
23          Tree[node].sum=Tree[2*node].sum+Tree[2*node+1].sum+(hi-lo+1)*Tree[node].
                offset;
24      }
25  }
26
27  long long Query(long long node,long long lo,long long hi,long long i,long long j
        ,long long offst){
28      if(lo>j || hi<i)
29          return 0;
30
31      if(lo>=i && hi<=j){
32          return offst*(hi-lo+1)+Tree[node].sum;
33      }else{
34          long long mid=(lo+hi)>>1;
35          offst+=Tree[node].offset;
36          long long q1=Query(2*node,lo,mid,i,j,offst);
37          long long q2=Query(2*node+1,mid+1,hi,i,j,offst);
38          return q1+q2;
39      }
40  }
41  void Clear(long long N){
42      for(long long i=0;i<4*N;i++){
43          Tree[i].offset=Tree[i].sum=0;
44      }
45  }
46
47  int main(){
48      int N,ofst=0;
49      Clear(N);
50      cout<<Query(1,1,N,x,y,ofst)<<endl;
51      Update(1,1,N,x,y,val);
52      return 0;
53  }
```

2.5. Trie

```
1   #include <iostream>
2   #include <vector>
3   #include <algorithm>
4   #include <cstdio>
5   #include <fstream>
6
7   using namespace std;
8   struct Trie{
9       Trie* hijos[26];
10      bool esfin;
11      vector<int> idhijos;
12      Trie(){
13          esfin = false;
14          for(int i = 0; i < 26; ++i)
15              hijos[i] = NULL;
16      }
17  };
18  void insertar(Trie* v, char* c){
19      if(c[0] == '\0')
```

```
20          v->esfin = true;
21      else{
22          int k = c[0] - 'a';
23          if(v->hijos[k] == NULL){
24              v->hijos[k] = new Trie();
25              (v->idhijos).push_back(k);
26          }
27          insertar(v->hijos[k], c+1);
28      }
29  }
30  void ordenarids(Trie *t){
31      sort((t->idhijos).begin(), (t->idhijos).end());
32      int tam = (t->idhijos).size();
33      for(int i = 0; i < tam; i++){
34          int k = t->idhijos[i];
35          ordenarids(t->hijos[k]);
36      }
37  }
38  bool haysol = false;
39  int tmp, caso;
40  char vacio[25]="";
41  string cad;
42  void mostrar(Trie* v, string p, char* pr){
43      if(pr[0] == '\0'){
44          if(v->esfin){
45              haysol = true;
46              puts(p.c_str());
47          }
48          int tam = (v->idhijos).size();
49
50          for(int i = 0; i < tam; ++i){
51              int k = v->idhijos[i];
52              mostrar(v->hijos[k], p + ((char)('a'+k)), vacio);
53          }
54      }
55      else{
56          int k = pr[0] - 'a';
57          if(v->hijos[k] != NULL){
58              bool aux = (v->hijos[k])->esfin;
59              (v->hijos[k])->esfin = false;
60
61              mostrar(v->hijos[k], p, pr+1);
62
63              (v->hijos[k])->esfin = aux;
64
65          }
66      }
67  }
68  int main(){
69      //freopen("entrada.in","r",stdin);
70      Trie* t;
71      t = new Trie();
72      scanf("%d", &tmp);
73      char cadena[25];
74
75      while(tmp--){
76          scanf("%s", cadena);
77          insertar(t, cadena);
```

```
78      }
79      ordenarids(t);;
80      cin >> tmp;
81      for(caso = 1; caso <= tmp; caso++){
82          scanf(" %s", cadena);
83          cad = cadena;
84          haysol = false;
85          printf("Case #%d:\n", caso);
86          mostrar(t, cad, cadena);
87          if(!haysol)
88              puts("No match.");
89      }
90      return 0;
91  }
```

Capítulo 3

Complete Search

3.1. El problema de las ocho reinas

```
int nrosol, f, c;
int sol[10];
bool sePuede(int fila, int col){
    if(col != c &&fila == f)return false;
    for(int i = col - 1 ; i >= 1; i--)
        if(sol[i] == fila || (abs(sol[i] - fila) == abs(i - col)))
            return false;
    return true;
}
void solve(int col){
    if(col == 9){
        if(sol[c] == f){
            printf(" %2d      %d", nrosol++, sol[1]);
            for(int i = 2; i <= 8; i++)
                printf(" %d", sol[i]);
            printf("\n");
        }
        return ;
    }
    for(int i = 1; i <= 8; i++)
        if(sePuede(i, col)){
            sol[col] = i;
            solve(col + 1);
        }
}
int main(){
    int T; cin >> T;
    while(T--){
        printf( "SOLN       COLUMN\n #      1 2 3 4 5 6 7 8\n\n" );
        cin >> f >> c;
        scanf(" %d %d", &f, &c);
        nrosol = 1;
        memset(sol, 0, sizeof(sol));
        sol[c] = f;
        solve(1);
    }
    return 0;
}
```

Capítulo 4

Divide y venceras

4.1. Busqueda Binaria (Recursivo)

```
template <class T>
int binsearch(const T array[], int len, T what)
{
    if (len == 0) return -1;
    int mid = len / 2;
    if (array[mid] == what) return mid;
    if (array[mid] < what) {
        int result = binsearch(array+mid+1, len-(mid+1), what);
        if (result == -1) return -1;
        else return result + mid+1;

    }
    if (array[mid] > what)
        return binsearch(array, mid, what);

}

#include <iostream>
int main()
{
    int array[] = {2, 3, 5, 6, 8};
    int result1 = binsearch(array, sizeof(array)/sizeof(int), 4),
        result2 = binsearch(array, sizeof(array)/sizeof(int), 8);
    if (result1 == -1) std::cout << "4 not found!" << std::endl;
    else std::cout << "4 found at " << result1 << std::endl;
    if (result2 == -1) std::cout << "8 not found!" << std::endl;
    else std::cout << "8 found at " << result2 << std::endl;

    return 0;

}
```

4.2. Busqueda Binaria (Iterativo)

```
template <class T>
int binSearch(const T arr[], int len, T what) {
    int low = 0;
    int high = len - 1;
    while (low <= high) {
```

```
6           int mid = (low + high) / 2;
7           if (arr[mid] > what)
8               high = mid - 1;
9           else if (arr[mid] < what)
10              low = mid + 1;
11          else
12              return mid;
13
14      }
15      return -1; //indica que no esta
16  }
```

4.3. MergeSort

Primera implementacion:

```
1   //Entrada: p = indice del primer elemento a ordenar
2   //         r = indice del ultimo elemento a ordenar
3   //Salida: El vector "a" ordenado
4   int a[LIM], L[LIM/2 + 4], R[LIM/2 + 4];
5   void fusionar(long p,long q,long r) {
6       long i,j,k,ind1,ind2;
7       ind1 = ind2 = 1;
8       for(i = p; i <= q; i++)
9           L[ind1++] = a[i];
10      L[ind1] = inf;
11      for(i = q+1; i <= r; i++)
12          R[ind2++] = a[i];
13      R[ind2] = inf;
14      i = j = 1;
15      for(k = p; k <= r; k++)
16          if(L[i] > R[j]) {
17              a[k] = R[j];
18              j++;
19          } else {
20              a[k] = L[i];
21              i++;
22          }
23  }
24  void mergeSort(long p,long r) {
25      if(p < r) {
26          long q = (p+r)/2;
27          mergeSort(p,q);
28          mergeSort(q+1,r);
29          fusionar(p,q,r);
30      }
31  }
```

Segunda implementación:

```
1   //Entrada: Un array
2   //Salida: El vector "array" ordenado
3   //        La cantidad optima de intercambios
4   int array[MAXN];
5   int temp[MAXN];
6   int intercambios = 0;
7   int d = 0;
8   void mergesort(int inicio, int length) {
```

```
9       if(length<=1)
10          return;
11      mergesort(inicio, length/2);
12      mergesort(inicio+length/2, length-length/2);
13      int i = inicio;
14      int j = inicio+length/2;
15      int n = 0;
16      while(i<inicio+length/2 && j<inicio+length)
17          if(array[i]<array[j]) {
18              temp[n++] = array[i++];
19              intercambios+=d; //para contar los intercambios
20          } else {
21              temp[n++] = array[j++];
22              d++;
23          }
24      while(i<inicio+length/2) {
25          temp[n++] = array[i++];
26          intercambios+=d;
27      }
28      while(j<inicio+length)
29          temp[n++] = array[j++];
30      memcpy(array+inicio, temp, length*sizeof(int));
31  }
```

4.4. Meet in the Middle

Si generamos los subconjuntos de una lista de N elementos; sabemos que existen 2N subconjuntos, pero esta complejidad es demasiado cuando N es grande, digamos (N>20).

Problema.- Dado una secuencia de N (1<= N <=34) números, determinar cuántos subconjuntos de esta secuencia tiene una suma entre A y B.

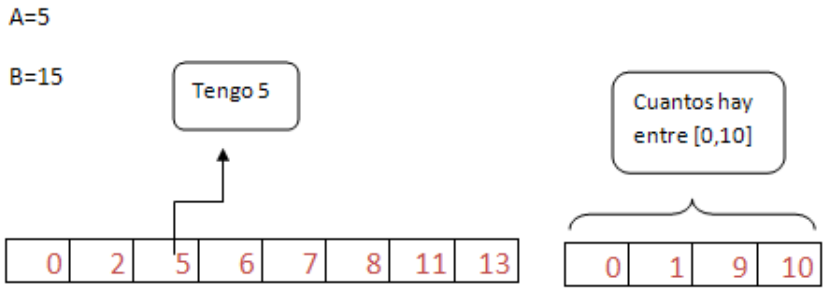
Esta técnica consiste en *dividir en dos listas por la mitad*, Luego generamos las sumatorias de los subconjuntos de cada lista y *los ordenamos*. Luego combinar ambos resultados.

Considerare la sumatoria de conjunto vacio como 0.

Ejemplo:

L= { 2 , 5 , 6 , 1, 9};
A=5, B=15
Los dividimos en los listas por la mitad.
X= { 2 , 5, 6}
Y= { 1 , 9}
Generamos los subconjuntos de cada lista.
X' = { 0, 2, 5 , {2,5}, 6,{2,6}, {5,6} ,{2,5,6} }
Y'= { 0, 1, 9, {1,9} }
Sumatorias de cada subconjunto de manera ordenada.
X'= {0,2,5,6,7,8,11,13}
Y'= {0,1,9,10}
Combinar resultados
Tenemos que recorrer la lista X' y ver cuánto nos falta para llegar al intervalo [A, B].

- Si el primer elemento de X' tenemos 0 entonces en la lista Y' tenemos que buscar cuantos elementos existen entre 5 y 15.
- Si el segundo elemento de X' tenemos 2 entonces en la lista Y' buscamos cuantos existen entre 3 y 13.



x

Si bien se dan cuenta es muy importante tener ambas listas de una manera ordenada para poder buscar cuantos elementos existen en la segunda matriz entre ese intervalo.

El algoritmo sigue buscando hasta terminar con el último elemento de la primera lista.

```
1  #include <vector>
2  #include <algorithm>
3  #include <iostream>
4
5  #define all(v) (v).begin(),(v).end()
6  #define rall(v) (v).rbegin(),(v).rend()
7  #define sz size()
8
9  using namespace std;
10 vector<int>A,B;
11 void subsets(vector<int> v, vector<int>&ans){
12     ans.clear();
13     int n=v.sz;
14     for(int mask=0;mask<(1<<n);mask++){
15         int sum=0;
16         for(int j=0;j<v.sz;j++)if(mask&(1<<j))sum+=v[j];
17         ans.push_back(sum);
18     }
19     sort(all(ans));
20 }
21
22 int main(){
23
24     int N,X,Y,n,nx;
25     cin>>N>>X>>Y;
26     vector<int>v(N);
27     for(int i=0;i<N;i++)cin>>v[i];
28     n=N/2;
29     subsets(vector<int>(v.begin(),v.begin()+n),A);
30     subsets(vector<int>(v.begin()+n,v.end()),B);
31     long long ans=0;
32     for(int i=0;i<A.sz;i++){
33         int val=A[i];
34         int frs=X-val;
35         int scd=Y-val;
36         int x1=lower_bound(all(B),frs)-B.begin();
37         int y1=upper_bound(all(B),scd)-B.begin();
38         if(y1>=x1)ans+=(y1-x1);
39     }
40     cout<<ans<<endl;
41     return 0;
42 }
```

Capítulo 5

Programación Dinamica

5.1. El problema de la mochila y Subset Sum

Subset Sum:

$$OPT(j,W) = max \begin{cases} OPT(j-1,W) & (w_j > W) \\ w_j + OPT(j-1,W-w_j) & (w_j \leq W) \end{cases}$$

0-1 Mochila

$$OPT(j,W) = max \begin{cases} OPT(j-1,W) & (w_j > W) \\ \mathbf{v_j} + OPT(j-1,W-w_j) & (w_j \leq W) \end{cases}$$

```
/*
Nota:
Si los pesos son muy grandes o el valor de llenar la mochila es muy grande
se puede utilizar un map en vez de una matriz para la memoizacion
*/
#include <cstdio>
#include <algorithm>
#include <cstring>

using namespace std;

long long int lib;

bool sel[35][1005], sol[35];//para reconstruccion (*)
int cant ; // * cantidad de elementos tomados

long long int dp[35][1005], v[35], p[35];//dp[OBJETOS + 1][LIBRE + 1]
long long int mochila(int i, int libre) { // Llamar mochila(N-1, libre)
  if(i < 0)return 0LL;
  long long int &ans = dp[i][libre];
  if(ans != -1) return ans;
  ans = mochila(i-1, libre);// no lo tomo
  if(p[i] <= libre) { // si cabe en el espacio disponible
    ans = max(ans ,v[i] + mochila(i-1, libre - p[i])); // pruebo si lo tomaria
    if(ans ==  v[i] + mochila(i-1, libre - p[i]))//si se tomo el objeto (*)
      sel[i][libre] = true;// marco como tomado (*)
  }
  return ans;
}

//* Marca en el vector sol[ELEMEMTOS] los elementos que fueron tomados
void reconstruir(int i, int libre) {
```

```
33   if(i < 0) return;
34   if(sel[i][libre]) { //si fue seleccionado
35     sol[i] = true;//marco
36     cant++;
37     reconstruir(i-1, libre - p[i]);
38   } else
39     reconstruir(i-1, libre);
40 }
41
42 int main() {
43   int n;
44   scanf("%d %lld", &n, &lib);
45   for(int i = 0; i < n; i++)
46     scanf("%lld %lld", &p[i], &v[i]);
47   memset(dp, -1, sizeof(dp));
48   memset(sel, false, sizeof(sel));//para la reconstruccion (*)
49   printf("%lld\n", mochila(n-1, lib));
50   // imprimir los elementos
51   memset(sol, false, sizeof(sol));
52   cant = 0;
53   reconstruir(n-1, lib);
54   printf("%d\n", cant);
55   for(int i = 0; i < n; i++)
56     if(sol[i])
57       printf("%lld %lld\n", p[i], v[i]);
58   return 0;
59 }
```

5.2. El problema de la mochila(iterativo)

```
1  /*
2  Nota:
3  -Tomar en cuenta que para calcular la fila i solo se necesita la fila i-1
       entonces,
4  si el problema utiliza demasiada memoria o no entra se puede hacer el mismo
       procedimiento
5  con solo 2 vectores llenando la fila actual con la fila ya calculada
6  y luego actual.swap(anterior), para todas la filas
7
8  - La version recursiva corre mas rapido por que no es necesario llenar toda la
       matriz dp
9  */
10 #include <cstdio>
11 #include <algorithm>
12 #include <cstring>
13
14 using namespace std;
15
16 long long int d[35], w, lib;
17 //Para recosntruccion (*)
18 bool sel[35][1005]; //seleccionados (*)
19 bool sol[35];//el elemento fue seleccionado (*)
20 int cant ; // * cantidad de elementos tomados (*)
21
22 long long int dp[35][1005], v[35], p[35];//dp[OBJETOS + 1][LIBRE + 1]
23
24 long long int mochila(int n, int libre) { //Llamar con mochila(N, libre)
```

```
25    memset(sel, false, sizeof(sel));
26    memset(dp, 0, sizeof(dp));
27    for(int i = 1; i <= n; i++)
28      for(int j = 1; j<= libre; j++)
29        if(p[i-1] <= j) {
30          dp[i][j] = max(dp[i-1][j-p[i-1]] + v[i-1], dp[i-1][j]);
31          if(dp[i][j] == dp[i-1][j-p[i-1]] + v[i-1])// se selecciono el elemento ?
                (*)
32            sel[i][j] = 1; // marcar (*)
33        } else
34          dp[i][j] = dp[i-1][j];
35
36    return dp[n][libre];
37  }
38  void reconstruir(int n, int libre) {
39    int j = libre;
40    cant = 0;
41    memset(sol, false, sizeof(sol));
42    for(int i = n; i >= 1; i--)
43      if(sel[i][j] == 1) {
44        sol[i-1] = true;
45        cant++;
46        j = j - p[i-1];
47      }
48  }
49
50  int main() {
51    int n;
52
53    scanf("%d %lld", &n, &lib);//N objetos y LIV espacio de la mochila
54    for(int i = 0; i < n; i++)
55      scanf("%lld %lld", &p[i], &v[i]);
56
57    memset(dp, -1, sizeof(dp));
58    memset(sel, false, sizeof(sel));//para reconstruccion (*)
59    printf("%lld\n", mochila(n, lib));
60
61    reconstruir(n, lib);
62    printf("%d\n", cant);
63    for(int i = 0; i < n; i++)
64      if(sol[i])
65        printf("%lld %lld\n", p[i], v[i]);
66    return 0;
67  }
```

5.3. Cambio de modenas

```
1   #define clr(a)  memset(a,0,sizeof(a))
2   using namespace std;
3   long long int dp[30001];
4   int money[]={1,5,10,25,50};
5   void coinchange(int sum){
6       clr(dp);
7       dp[0]=1;
8       for(int i=0;i<5;i++){
9           for(int j=money[i];j<=sum;j++){
10              dp[j]+=dp[j-money[i]];
```

```
        }
    }
}
```

5.4. LIS ($O(n^2)$)

```
#include <cstdio>
#include <vector>
#include <algorithm>

using namespace std;

vector<long long int> v;
vector<long long int> ancho;
vector<long long int> dp;
int t, n;
long long int lis(int i) {
  if(i == 0)return ancho[0];//(*) si no es pesado 1
  long long int ans = dp[i];
  if(ans != -1LL) return ans;
  ans = ancho[i];//(*) 1
  for(int j = 0; j < i; j++)
    if(v[j] < v[i]) // si no es estrictamente creciente "<="
      ans = max(ans, ancho[i] + lis(j)); // 1 + lis(j)
  return dp[i] = ans;
}
int solve() {
  (vector<long long int>(n, -1LL)).swap(dp);
  long long int ans = 0;
  for(int i = n-1; i >= 0; i--)
    ans = max(ans, lis(i));
  return ans;
}

int main() {
    scanf("%d", &n);
    (vector<long long int>(n)).swap(v);
    (vector<long long int>(n)).swap(ancho);
    for(int i = 0; i < n; i++)
      scanf("%lld", &v[i]);
    for(int i = 0; i < n; i++)//si es pesado
      scanf("%lld", &ancho[i]);//
    printf("lis = %d\n", solve());
  return 0;
}
```

5.5. LIS ($O(n\,log(n))$)

```
#include <algorithm>
#include <cstdio>
#include <stack>
using namespace std;

```

```
#define MAX_N 100000

void imprimir(int end, int a[], int p[]) {
  int x = end;
  stack<int> s;
  for (; p[x] >= 0; x = p[x]) s.push(a[x]);
  printf("%d\n", a[x]);
  for (; !s.empty(); s.pop()) printf("%d\n", s.top());
}

int main() {
  int A[MAX_N], L[MAX_N], L_id[MAX_N], P[MAX_N];
  int n, tmp;
  scanf("%d", &n);
  for(int i = 0; i < n; i++)
    scanf("%d", &A[i]);

  int lis = 0, lis_end = 0;
  for (int i = 0; i < n; i++) {
    int pos = lower_bound(L, L + lis, A[i]) - L;
    L[pos] = A[i];
    L_id[pos] = i;//(*) marcamos que posicion es la que estamso guardando
    P[i] = pos ? L_id[pos - 1] : -1;//(*) El padre del i esimo elemento
    if (pos + 1 > lis) {
      lis = pos + 1;
      lis_end = i;
    }
  }
  printf("%d\n", lis);
  imprimir(lis_end, A, P);
  return 0;
}
```

5.6. Suma maxima en rango

```
//Algoritmo de kadane
//Entrada: Un vector de numero enteros
//Salida : La suma maxima de alguno de rangos en O(n)
long long int maxrangesum(vector<int> &v) {
  long long int sum = 0, ans = -(1LL<<60);//la maxima suma podria ser negativa
  for(int i = 0; i < v.size(); i++) {
    sum += v[i];
    ans = max(ans, sum);
    if(sum < 0) sum = 0;
  }
  return ans;
}
```

5.7. Suma maxima en rango 2D $O(n^4)$

```
#include <cstdio>
#include <algorithm>
#define MAX 103
using namespace std;
```

```
int v[MAX][MAX] = {0};
int main() {
    int n, tmp;
    scanf("%d", &n);
    for(int i = 1; i <= n; i++)
        for(int j = 1; j <= n; j++) {
            scanf("%d", &v[i][j]);
            v[i][j] += (v[i-1][j] + v[i][j-1] - v[i-1][j-1]);
        }
    int ans = -127*100*100;
    for(int i = 1; i <= n; i++)for(int j = 1; j <= n; j++)
    for(int k = i; k <= n; k++)for(int l = j; l <= n; l++)
        ans = max(ans, v[k][l] - v[i-1][l] - v[k][j-1] + v[i-1][j-1]);
    printf("%d\n", ans);
    return 0;
}
```

5.8. Suma maxima en rango 2D $O(n^3)$

```
#include <algorithm>
#include <cstdio>
using namespace std;

int n, A[101][101] = {0}, maxSubRect, subRect;

int main() {  // O(n^3) 1D DP + greedy (Kadane) solucion
  scanf("%d", &n);            // La dimension de la matriz
  for (int i = 1; i <= n; i++)
    for (int j = 1; j <= n; j++) {
      scanf("%d", &A[i][j]);
      A[i][j] += A[i][j - 1]; // vamos acumulando de fila en fila
    }

  maxSubRect = -127*100*100; // El valor mas bajo para la respuesta
  for (int l = 1; l <= n; l++)
    for (int r = l; r <= n; r++) {
      subRect = 0;
      for (int row = 1; row <= n; row++) {
        // Max 1D Range Sum en las columnas de la fila i
        subRect += A[row][r] - A[row][l - 1];

        // El algoritmo de Kadane en las filas
        if (subRect < 0) subRect = 0;     // golosamente, reiniciamos si sum < 0
        maxSubRect = max(maxSubRect, subRect);
      }
    }
  printf("%d\n", maxSubRect);
  return 0;
}
```

5.9. Maxima submatriz de ceros

```
#include <vector>
#include <iostream>
```

```
#include <stack>
using namespace std;
int main(){
    int n, m;
    cin >> n >> m;
    vector < vector<char> > a (n, vector<char> (m));
    for (int i=0; i<n; ++i)
        for (int j=0; j<m; ++j)
            cin >> a[i][j];

    int ans = 0;
    vector<int> d (m, -1);
    vector<int> dl (m), dr (m);
    stack<int> st;
    for (int i=0; i<n; ++i) {
        for (int j=0; j<m; ++j)
            if (a[i][j] == 1)
                d[j] = i;
        while (!st.empty()) st.pop();
        for (int j=0; j<m; ++j) {
            while (!st.empty() && d[st.top()] <= d[j]) st.pop();
            dl[j] = st.empty() ? -1 : st.top();
            st.push (j);

        }
        while (!st.empty()) st.pop();
        for (int j=m-1; j>=0; --j) {
            while (!st.empty() && d[st.top()] <= d[j]) st.pop();
            dr[j] = st.empty() ? m : st.top();
            st.push (j);
        }
        for (int j=0; j<m; ++j)
            ans = max (ans, (i - d[j]) * (dr[j] - dl[j] - 1));
    }
    cout << ans;
    return 0;
}
```

5.10. Submatriz de suma maxima

```
void subMatriz_con_sumaMaxima(int m[n][n]){
    for(int i = 1; i <= n; i++){
        vector<int> vc(n + 1,0);//para aplicar DP magico
        for(int j = i; j <= n; j++){
            mn_temp = n * n;
            for(int k = 1; k <= n; k++)
                vc[k] += m[j][k];

            dp[0] = 0;
            for(int i = 1; i <= n;i++){
                dp[i] = max(vc[i], dp[i-1] + vc[i]);
                if(res < dp[i])
                    res = dp[i];
            }
        }
    }

```

```
18      printf("%d\n",res);
19  }
```

5.11. Distancia de edición (Algoritmo de Levenshtein)

```
1   #include <string>
2   #include <vector>
3   #include <algorithm>
4   #include <iostream>
5
6   using namespace std;
7   /*Encuentra la distancia de edicion entre 2 cadenas*/
8   /*
9   El numero minimos de cambios para transformar la cadena s1 en la cadena s2, con
        los siguientes cambios:
10
11  - Elimina una letra de una de las cadenas
12  - Inserta una letra en una de las cadenas
13  - Reemplaza una letra en una de las cadenas
14  */
15  int levenshtein(const string &s1, const string &s2)
16  {
17      int N1 = s1.size();
18      int N2 = s2.size();
19      int i, j;
20      vector<int> T(N2+1);
21
22      for ( i = 0; i <= N2; i++  )
23          T[i] = i;
24
25      for ( i = 0; i < N1; i++  ) {
26          T[0] = i+1;
27          int corner = i;
28          for ( j = 0; j < N2; j++  ) {
29              int upper = T[j+1];
30              if ( s1[i] == s2[j]  )
31                  T[j+1] = corner;
32              else
33                  T[j+1] = min(T[j], min(upper, corner)) + 1;
34              corner = upper;
35
36          }
37
38      }
39      return T[N2];
40  }
41  int main(){
42      int casos;
43      cin >> casos;
44      string a, b;
45      while(casos--){
46          cin >> a >> b;
47          if(a.size() > b.size())
48              swap(a,b);
49          cout << levenshtein(a,b) << endl;
50      }
51      return 0;
```

```
52 }
```

5.12. Distancia de edición (Recursivo)

```
1  #include <cstdio>
2  #include <cstring>
3
4  #define MAXS 2005
5  #define max(a,b) a>b?a:b
6  #define min(a,b) a<b?a:b
7  using namespace std;
8
9  char a[MAXS], b[MAXS];
10 int ta, tb;
11
12 int casos;
13 int dp[MAXS][MAXS];
14
15 int solve(int ta, int tb){
16     if(ta < 0 && tb < 0)
17         return 0;
18     if(ta < 0)
19         return tb + 1;
20     if(tb < 0)
21         return ta + 1;
22
23     int &ans = dp[ta][tb];
24     if(ans == -1){
25         ans = 1<<30;
26
27         if(a[ta] == b[tb] )
28             ans = min(ans, solve(ta - 1, tb - 1));
29         else
30             ans = min(ans, 1 + solve(ta - 1, tb - 1));// con un solo cambio
                   convierto en lo mismo
31         ans = min(ans, 1 + solve(ta - 1, tb    ));
32         ans = min(ans, 1 + solve(ta    , tb - 1));
33     }
34     return ans;
35 }
36 int main(){
37     //freopen("entrada.in", "r", stdin);
38     scanf("%d", &casos);
39     while(casos--){
40         scanf("%s %s", a, b);
41         ta = strlen(a);
42         tb = strlen(b);
43         memset(dp, -1, sizeof(dp));
44         printf("%d\n",solve(ta-1, tb - 1));
45     }
46     return 0;
47 }
```

Capítulo 6

Grafos

6.1. BFS

```
#include <queue>
#include <vector>
#include <cstring>

#define MAXN 1000

using namespace std;

vector<int> grafo[MAXN];
int d[MAXN];
queue<int> cola;

void BFS(int ini){
    memset(d,-1, sizeof d);
    cola.push(ini);
    d[ini] = 0;
    while(!cola.empty()){
        int act = cola.front();cola.pop();
        for(int i = 0; i < grafo[act].size(); i++){
            int ady = grafo[act][i];
            if(d[ady] == -1){
                d[ady] = d[act] + 1;
                cola.push(ady);
            }
        }
    }
}
```

6.2. DFS

```
#define MAX 1001
#define pain 64
int n;
int mat[MAX][MAX];
bool visitado[MAX][MAX];
int di[]={1,1, 1,-1,-1,-1,0, 0};
int dj[]={1,0,-1, 1, 0,-1,1,-1};

void ff_dfs(int i, int j) {
```

```
    visitado[i][j] = true;
    for(int k=0; k<8; k++) {
        int a = di[k] + i;
        int b = dj[k] + j;
        if(a >= 0 && b >= 0 && a < n && b < n &&!visitado[a][b]) {
            if(mat[a][b]==pain) ff_dfs(a,b);
            else visitado[a][b]=true;
        }
    }
}
```

6.3. Topological Sort

Grafo Aciclico Dirigido (DAG)

Las tareas no son independientes y la ejecución de una tarea sólo es posible si otras tareas que ya se han ejecutado. Solo hay un orden

```
using namespace std;
vector<int>graph[110];
bool visitado[110];
vector<int> sol;
int n,m;
void dfs(int node) {
    visitado[node] = true;
    for (int i = 0; i < graph[node].size(); i++) {
        if (!visitado[graph[node][i]])
            dfs(graph[node][i]);
    }
    sol.push_back(node);
}
void lim(){
    for(int i=0;i<=110;i++)
        graph[i].clear();

    clr(visitado);
    sol.clear();
}
int main() {
    //llenado
        for (int i = 1; i <= n; i++)
            if(!visitado[i])dfs(i);
        reverse(sol.begin(), sol.end());
        //printsol
    return 0;
}
```

6.4. Dijkstra

```
#include <stdio.h>
#include <vector>
#include <queue>
using namespace std;
#define MAX 10005
#define Node pair< int , int >
#define INF 1<<30
```

```
8  struct cmp {
9      bool operator() ( const Node &a , const Node &b ) {
10         return a.second > b.second;
11     }
12 };
13 vector< Node > ady[ MAX ];
14 int distancia[ MAX ];
15 bool visitado[ MAX ];
16 priority_queue< Node , vector<Node> , cmp > Q;
17 int V;
18 int previo[ MAX ];
19
20
21 void init(){
22     for( int i = 0 ; i <= V ; ++i ){
23         distancia[ i ] = INF;
24         visitado[ i ] = false;
25         previo[ i ] = -1;
26     }
27 }
28 void relajacion( int actual , int adyacente , int peso ){
29     if( distancia[ actual ] + peso < distancia[ adyacente ] ){
30         distancia[ adyacente ] = distancia[ actual ] + peso;
31         previo[ adyacente ] = actual;
32         Q.push( Node( adyacente , distancia[ adyacente ] ) );
33     }
34 }
35 void print( int destino ){
36     if( previo[ destino ] != -1 )
37         print( previo[ destino ] );
38     printf("%d " , destino );
39 }
40
41 void dijkstra( int inicial ){
42     init();
43     Q.push( Node( inicial , 0 ) );
44     distancia[ inicial ] = 0;
45     int actual , adyacente , peso;
46     while( !Q.empty() ){
47         actual = Q.top().first;
48         Q.pop();
49         if( visitado[ actual ] ) continue;
50         visitado[ actual ] = true;
51
52         for( int i = 0 ; i < ady[ actual ].size() ; ++i ){
53             adyacente = ady[ actual ][ i ].first;
54             peso = ady[ actual ][ i ].second;
55             if( !visitado[ adyacente ] ){
56                 relajacion( actual , adyacente , peso );
57             }
58         }
59     }
60
61
62     printf( "Distancias mas cortas iniciando en vertice %d\n" , inicial );
63     for( int i = 1 ; i <= V ; ++i ){
64         printf("Vertice %d , distancia mas corta = %d\n" , i , distancia[ i ] );
65     }
```

```
66
67      puts("\n**************Impresion de camino mas corto**************");
68      printf("Ingrese vertice destino: ");
69      int destino;
70      scanf("%d" , &destino );
71      print( destino );
72      printf("\n");
73  }
74
75
76  int main(){
77      int E , origen, destino , peso , inicial;
78      scanf("%d %d" , &V , &E );
79      while( E-- ){
80          scanf("%d %d %d" , &origen , &destino , &peso );
81          ady[ origen ].push_back( Node( destino , peso ) ); //consideremos grafo
                dirigido
82          //ady[ destino ].push_back( Node( origen , peso ) ); //grafo no dirigido
83      }
84      printf("Ingrese el vertice inicial: ");
85      scanf("%d" , &inicial );
86      dijkstra( inicial );
87      return 0;
88  }
```

6.5. BellmandFord

Si un grafo contiene un ciclo de coste total negativo entonces este grafo no tiene solución

```
1   #include <cstdio>
2   #include <cstdlib>
3   #include <iostream>
4   #include <vector>
5   #define f first
6   #define s second
7   #define pb push_back
8   #define mp make_pair
9   using namespace std;
10
11  #define M 1000000000
12  typedef pair<pair<int,int>,int> P;
13  int N;
14  vector <P> v;
15  int bellmandford(int a, int b){
16      vector<int>d(N,M);
17      d[a]=0;
18      for(int i=0;i<N;i++){
19          for(int j=0;j<v.size();j++){
20              if(d[v[j].f.f]<M && d[v[j].f.f]+v[j].s < d[v[j].f.s]){
21                  d[v[j].f.s] = d[v[j].f.f]+v[j].s;
22              }
23          }
24      }
25      for(int j=0;j<v.size();j++){
26          if(d[v[j].f.f] < M && d[v[j].f.f]+v[j].s < d[v[j].f.s]){
27              return -M; /// ciclo negativo
28          }
```

```
    }
    return d[b];
}
int main(){
    N=5;
    v.pb(mp(mp(0,1),-1));
    v.pb(mp(mp(0,2),0));
    v.pb(mp(mp(1,3),3));
    v.pb(mp(mp(2,1),2));
    v.pb(mp(mp(3,2),-6));
    v.pb(mp(mp(3,4),-3));
    printf("%d\n",bellmandford(0,4));
    return 0;
}
```

6.6. Floyd Warshall

```
int dp[100][100]; //grafo matriz de ady

void floyd_warshall(){
   for(int k=1;k<=n;k++)
      for(int i=1;i<=n;i++)
         for(int j=1;j<=n;j++)
            dp[i][j] = min(dp[i][j],dp[i][k] + dp[k][j]);
}
```

6.7. Componentes Fuertemente Conexas

```
#include <cstring>
#include <string>
#include <cstdio>
#include <cstdlib>
#include <vector>
#include <algorithm>
#include <iostream>

#define clr(a)  memset(a,0,sizeof(a))
#define pb push_back
using namespace std;
vector<int>grafo[N];
vector<int>gaux;
bool visitado[N];
void dfs(int u){
    visitado[u]=true;
    for(int i=0;i<grafo[u].size();i++){
        if(!visitado[grafo[u][i]]) dfs(grafo[u][i]);
    }
}
int main(){
    //r(input);
    int t;
    int x,y;
    int n,m;
    sc("%d",&t);
```

```
    while(t--){
        sc("%d %d",&n,&m);
        clr(visitado);
        for(int i=0;i<=n;i++)   grafo[i].clear();
            gaux.clear();
        while(m--){
            sc("%d %d",&x,&y);
            grafo[x].pb(y);
        }
        int M=0;
        for(int i=1;i<=n;i++){
            if(!visitado[i]){
                dfs(i),gaux.pb(i);
            }
        }
        clr(visitado);
        M=0;
        for(int i=gaux.size()-1;i>=0;i--){
            if(!visitado[gaux[i]]){
                dfs(gaux[i]);
                M++;
            }
        }
        cout<<M<<endl;
    }
}
```

6.8. Componentes Fuertemente Conexas (Kosaraju)

```
#include <stack>
#include <queue>
#include <vector>
#include <cstring>

#include <map>
#include <cstdio>
#include <iostream>

using namespace std;

/*
Algoritmo de Kosaraju para
Componentes Fuertemente Conexas
*/

#define MAX_V 1000

int V,num_scc;
vector< vector<int> > G;
vector< vector<int> > GT;
bool visited[MAX_V];
stack<int> S;
queue<int> Q;

void dfs(int v){
    visited[v] = true;

```

```
29      for(int i=G[v].size()-1;i>=0;--i)
30          if(!visited[G[v][i]])
31              dfs(G[v][i]);
32
33      S.push(v);
34  }
35
36  void bfs(int v){
37      Q.push(v);
38      visited[v] = true;
39
40      int aux;
41
42      while(!Q.empty()){
43          aux = Q.front();
44          Q.pop();
45
46          for(int i=GT[aux].size()-1;i>=0;i--){
47              if(!visited[GT[aux][i]]){
48                  Q.push(GT[aux][i]);
49                  visited[GT[aux][i]] = true;
50              }
51          }
52      }
53  }
54  void SCC(){
55      memset(visited,false,sizeof(visited));
56
57      for(int i=0;i<V;++i) if(!visited[i]) dfs(i);
58
59      num_scc = 0;
60      int aux;
61
62      memset(visited,false,sizeof(visited));
63
64      for(int i=0;i<V;++i){
65          aux = S.top();
66          S.pop();
67
68          if(!visited[aux]){
69              bfs(aux);
70              ++num_scc;
71          }
72      }
73  }
74
75  int main(){
76      int E,u,v;
77      string s;
78      map<string, int> num;
79
80      while(true){
81          scanf("%d %d",&V,&E);
82          if(V==0) break;
83
84          getline(cin,s);
85          num.clear();
86
```

```
87          for(int i=0;i<V;++i){
88              getline(cin,s);
89              num[s] = i;
90          }
91
92          G.clear(); G.resize(V);
93          GT.clear(); GT.resize(V);
94
95          for(int i=0;i<E;++i){
96              getline(cin,s);
97              u = num[s];
98              getline(cin,s);
99              v = num[s];
100
101             G[u].push_back(v);
102             GT[v].push_back(u);
103         }
104
105         SCC();
106
107         printf(" %d\n",num_scc);
108     }
109
110     return 0;
111 }
```

6.9. Componentes Fuertemente Conexas (Tarjan $O(n + v)$)

```
1  #include<iostream>
2  #include<vector>
3  #include<cstdio>
4
5  using namespace std;
6
7  typedef long long lli;
8  typedef pair<int, int> pii;
9  #define LENGTH(a) ((int)a.length())
10 #define SIZE(a) ((int)a.size())
11 int const MAX = 100;
12 int const INF = 1000000;
13
14 class Grafo
15 {
16 public:
17     vector<vector<int> > L;
18     vector<int> num, low;
19     vector<bool> visitado;
20     vector<int> componente;
21     int nComponentes, numCnt;
22
23     Grafo(int n)
24     {
25         init(n);
26     }
27
28     void init(int n)
29     {
```

```
30          L = vector<vector<int> >(n);
31          num = vector<int>(n, -1);
32          low = vector<int>(n, 0);
33          visitado = vector<bool>(n, false);
34          componente.clear();
35          nComponentes = numCnt = 0;
36      }
37
38      void add(int a, int b)
39      {
40          L[a].push_back(b);
41      }
42
43      void ejecutarTarjan()
44      {
45          for (int i = 0; i < SIZE(L); ++i)
46              if (num[i] == -1)
47                  tarjan(i);
48      }
49
50      void tarjan(int u)
51      {
52          low[u] = num[u] = numCnt++;
53          visitado[u] = true;
54          componente.push_back(u);
55          for (int i = 0; i < SIZE(L[u]); ++i)
56          {
57              int v = L[u][i];
58              if (num[v] == -1)
59                  tarjan(v);
60              if (visitado[v])
61                  low[u] = min(low[u], low[v]);
62          }
63          if (num[u] == low[u])
64          {
65              nComponentes++;
66              int v;
67              do
68              {
69                  v = componente[SIZE(componente) - 1];
70                  visitado[v] = false;
71                  componente.erase(--componente.end());
72              }
73              while (v != u);
74          }
75      }
76  };
77
78  int main(){
79  int nodos;
80  cin>>nodos;
81  Grafo g(nodos);
82  //leer lados
83  g.ejecutarTarjan();
84  //g.nComponentes contiene el numero de SCC
85  return 0;
86  }
```

Otra implementacion:

```java
import java.util.*;

public class SCCTarjan {

  int time;
  List<Integer>[] graph;
  int[] lowlink;
  boolean[] used;
  List<Integer> stack;
  List<List<Integer>> components;

  public List<List<Integer>> scc(List<Integer>[] graph) {
    int n = graph.length;
    this.graph = graph;
    lowlink = new int[n];
    used = new boolean[n];
    stack = new ArrayList<>();
    components = new ArrayList<>();

    for (int u = 0; u < n; u++)
      if (!used[u])
        dfs(u);

    return components;
  }

  void dfs(int u) {
    lowlink[u] = time++;
    used[u] = true;
    stack.add(u);
    boolean isComponentRoot = true;

    for (int v : graph[u]) {
      if (!used[v])
        dfs(v);
      if (lowlink[u] > lowlink[v]) {
        lowlink[u] = lowlink[v];
        isComponentRoot = false;
      }
    }

    if (isComponentRoot) {
      List<Integer> component = new ArrayList<>();
      while (true) {
        int k = stack.remove(stack.size() - 1);
        component.add(k);
        lowlink[k] = Integer.MAX_VALUE;
        if (k == u)
          break;
      }
      components.add(component);
    }
  }

  public static void main(String[] args) {
    List<Integer>[] g = new List[1000];
    for (int i = 0; i < g.length; i++) {
      g[i] = new ArrayList<>();
```

```
    }

    List<List<Integer>> components = new SCCTarjan().scc(g);
    System.out.println(components);
  }
}
```

6.10. Minimum Spanning Tree (Kruskall)

```
#define lim 100000
using namespace std;
typedef vector<pair<int,pair<int,int> > > V;
int N,lider[lim];
V v;
void init(){
    Sort(v);
    for(int i=0;i<N;i++)lider[i]=i;
}
int find(int n){
    if(n==lider[n])return n;
    else return lider[n]=find(lider[n]);
}
int kruskall(){
    int a,b,sum=0;
    init();
    for(int i=0;(int)i<v.sz;i++){
        a=find(v[i].s.f);
        b=find(v[i].s.s);
        if(a!=b){
            lider[b]=a;
            sum+=v[i].f;
        }
    }
    return sum;
}
//v.pb(mp(c,mp(a,b))); peso(c),arista(a,b)
```

6.11. Minimum Cut

```
#include <iostream>
#include <limits.h>
#include <string.h>
#include <queue>
using namespace std;

#define V 100

int 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;
```

```
16      parent[s] = -1;
17      while (!q.empty())
18      {
19          int u = q.front();
20          q.pop();
21          for (int v=0; v<V; v++)
22          {
23              if (visited[v]==false && rGraph[u][v] > 0)
24              {
25                  q.push(v);
26                  parent[v] = u;
27                  visited[v] = true;
28              }
29          }
30      }
31      return (visited[t] == true);
32  }
33  void dfs(int rGraph[V][V], int s, bool visited[])
34  {
35      visited[s] = true;
36      for (int i = 0; i < V; i++)
37         if (rGraph[s][i] && !visited[i])
38             dfs(rGraph, i, visited);
39  }
40
41  void minCut(int graph[V][V], int s, int t)
42  {
43      int u, v;
44      int rGraph[V][V];
45      for (u = 0; u < V; u++)
46          for (v = 0; v < V; v++)
47               rGraph[u][v] = graph[u][v];
48
49      int parent[V];
50      while (bfs(rGraph, s, t, parent))
51      {
52          int path_flow = INT_MAX;
53          for (v=t; v!=s; v=parent[v])
54          {
55              u = parent[v];
56              path_flow = min(path_flow, rGraph[u][v]);
57          }
58
59          for (v=t; v != s; v=parent[v])
60          {
61              u = parent[v];
62              rGraph[u][v] -= path_flow;
63              rGraph[v][u] += path_flow;
64          }
65      }
66      bool visited[V];
67      memset(visited, false, sizeof(visited));
68      dfs(rGraph, s, visited);
69      int costoCorte=0;
70      for (int i = 0; i < V; i++)
71        for (int j = 0; j < V; j++)
72           if (visited[i] && !visited[j] && graph[i][j]){
73                //cout << i << " - " << j << endl;
```

```
74                costoCorte+=graph[i][j];
75            }
76             cout<<costoCorte<<endl;
77        return;
78    }
79
80    int main()
81    {
82        int graph[V][V];
83        minCut(graph, 0, 5);
84        return 0;
85    }
```

6.12. MaxFlow

```
1   #include <algorithm>
2   #include <cstring>
3   #include <cstdio>
4   #include <limits>
5   #include <queue>
6
7   using namespace std;
8
9   #define N 1000 //maximo de nodos
10  //halla el flujo maximo desde el nodo s
11  //hasta el nodo t
12  //cap[][] almacena flujos parciales
13  int cap[N][N], padre[N], n, s, t;
14
15  bool bfs() {
16      queue<int> q;
17      q.push(s);
18      memset(padre, -1, sizeof padre);
19      padre[s] = s;
20      while (q.size()) {
21          int u = q.front();
22          q.pop();
23          if (u == t)
24              return true;
25          for (int v = 0; v < n; ++v)
26              if (padre[v] == -1 && cap[u][v])
27                  padre[v] = u, q.push(v);
28      }
29      return false;
30  }
31
32  int maxFlow() {
33      int mf = 0, f, v;
34      while (bfs()) {
35
36          v = t;
37          f = numeric_limits<int>::max();
38          while (padre[v] != v)
39              f = min(f, cap[padre[v]][v]), v = padre[v];
40
41          v = t;
42          mf += f;
```

```
43          while (padre[v] != v)
44              cap[padre[v]][v] -= f, cap[v][padre[v]] += f, v = padre[v];
45      }
46      return mf;
47  }
48
49
50  #define capacidad(i, j, c) cap[i][j] += c, cap[j][i] += c
51  int main() {
52      int c;
53          printf("ingrese el numero de nodos\n");
54          scanf("%d", &n);
55          printf("ingrese el inicio destino y nro nodos");
56          memset(cap, 0, sizeof cap);
57          scanf("%d %d %d", &s, &t, &c);
58          --s, --t;
59          printf("ingrese los lados x y & capacidad");
60          for (int i = 0, x, y, z; i < c; ++i)
61              scanf("%d %d %d", &x, &y, &z), capacidad(x - 1, y - 1, z);
62
63
64          printf("el flujo maximo entre s y t es %d\n", maxFlow());
65
66      return 0;
67  }
```

6.13. Lowest common ancestor (LCA)

```
1   #include <cctype>
2   #include <cstdio>
3   #include <iostream>
4   #define NN 100048
5   using namespace std;
6
7   int P[NN][17], L[NN];
8   long long W[NN];
9
10  int query(int p, int q)
11  {
12      int log, i;
13      if (L[p] < L[q]) p ^= q ^= p ^= q;
14      for (log = 1; 1 << log <= L[p]; log++);
15      log--;
16      for (i = log; i >= 0; i--)
17          if (L[p] - (1 << i) >= L[q])
18              p = P[p][i];
19      if (p == q) return p;
20      for (i = log; i >= 0; i--)
21          if (P[p][i] != -1 && P[p][i] != P[q][i])
22              p = P[p][i], q = P[q][i];
23
24      return P[p][0];
25  }
26  int main(void)
27  {
28      int a, b, N, q;
29      cin>>N;//nodos
```

```
    long long w;
    P[0][0] = -1;
    L[0] = W[0] = 0;
    int lados;
    cin>>lados;
    int ii;//(ii<=>desde), (a<=> hasta), (w<=>peso)
    for(int i = 0; i < lados; ++i)
    {
        cin>>ii>>a>>w;
        for(int j = 0; (1 << j) < N; ++j) P[ii][j] = -1;
        P[ii][0] = a;
        L[ii] = L[a] + 1;
        W[ii] = W[a] + w;
    }
    for(int j = 1; (1 << j) < N; ++j)
        for(int i = 0; i < N; ++i)
            if(P[i][j - 1] != -1)
                P[i][j] = P[P[i][j - 1]][j - 1];
    cin>>q;
    while(q--)
    {
        cin>>a>>b;
        printf("%lld\n", W[a] + W[b] - (W[query(a, b)] << 1));
    }
    return 0;
}
```

6.14. Puntos Criticos

```
import java.util.*;

public class Bridges {

  static int time;
  static List<Integer>[] graph;
  static boolean[] used;
  static int[] tin;
  static int[] lowlink;
  static List<String> bridges;

  static void dfs(int u, int p) {
    used[u] = true;
    lowlink[u] = tin[u] = time++;
    for (int v : graph[u]) {
      if (v == p) {
        continue;
      }
      if (used[v]) {
        // lowlink[u] = Math.min(lowlink[u], lowlink[v]);
        lowlink[u] = Math.min(lowlink[u], tin[v]);
      } else {
        dfs(v, u);
        lowlink[u] = Math.min(lowlink[u], lowlink[v]);
        if (lowlink[v] > tin[u] && p != -1) {
          bridges.add("(" + u + "," + v + ")");
        }
      }
```

```
    }
  }
  public static void main(String[] args) {
    time = 0;
    int n = 6;
    graph = new List[n];
    for (int i = 0; i < n; i++) {
      graph[i] = new ArrayList<>();
    }
    used = new boolean[n];
    tin = new int[n];
    lowlink = new int[n];
    bridges = new ArrayList<>();

    dfs(0, -1);

    System.out.println(bridges);
  }
}
```

6.15. Maximum Bipartite Matching

```
import java.util.*;

public class MaxMatching2 {

  static boolean encontrarCamino(List<Integer>[] g, int u1, int[] matching,
      boolean[] vis) {
    vis[u1] = true;
    for (int v : g[u1]) {
      int u2 = matching[v];
      if (u2 == -1 || !vis[u2] && encontrarCamino(g, u2, matching, vis)) {
        matching[v] = u1;
        return true;
      }
    }
    return false;
  }

  public static int maxEmparejamiento(List<Integer>[] g, int n2) {
    int n1 = g.length;
    int[] matching = new int[n2];
    Arrays.fill(matching, -1);
    int matches = 0;
    for (int u = 0; u < n1; u++) {
      if (encontrarCamino(g, u, matching, new boolean[n1]))
        ++matches;
    }
    return matches;
  }

  public static void main(String[] args) {
    int n1 = 2;
    int n2 = 3;
    List<Integer>[] g = new List[n1];
    for (int i = 0; i < n1; i++) {
      g[i] = new ArrayList<Integer>();
```

```
    }
    //grafo dirigido
    g[0].add(2);
    g[0].add(0);
    g[1].add(2);
    System.out.println(maxEmparejamiento(g, n2));
  }
}
```

6.16. Puntos de Articulación

```
import java.util.*;

public class ArticulationPoints {

  static int time;
  static List<Integer>[] grafo;
  static boolean[] vis;
  static int[] tin;
  static int[] dfsLow;
  static List<Integer> puntosArticulacion;

  static void dfs(int u, int p) {
    vis[u] = true;
    dfsLow[u] = tin[u] = time++;
    int child = 0;
    for (int v : grafo[u]) {
      if (v == p) {
        continue;
      }
      if (vis[v]) {
        dfsLow[u] = Math.min(dfsLow[u], tin[v]);
      } else {
        dfs(v, u);
        dfsLow[u] = Math.min(dfsLow[u], dfsLow[v]);
        if (dfsLow[v] >= tin[u] && p != -1) {
          puntosArticulacion.add(u);
        }
        ++child;
      }
    }
    if (p == -1 && child > 1) {
      puntosArticulacion.add(u);
    }
  }

  public static void main(String[] args) {
    time = 0;
    int n = 5;
    grafo = new List[n];
    for (int i = 0; i < n; i++) {
      grafo[i] = new ArrayList<>();
    }

    vis = new boolean[n];
    tin = new int[n];
    dfsLow = new int[n];
```

```
47      puntosArticulacion = new ArrayList<>();
48
49      dfs(0, -1);
50
51      //System.out.println(puntosArticulacion); si puntosArticulacion == null
            entonces es un grafo biconexo
52    }
53  }
```

Capítulo 7

Matemáticas

7.1. GCD

```
1  //Entrada: Dos enteros a y b
2  //Salida: El maximo comun divisor entre a y b
3
4  #include <algorithm>
5  int gcd (int a, int b){
6      __gcd(a,b);
7  }
8
9  int gcd (int a, int b){
10     return b==0?a:gcd(b, a % b);
11 }
12
13 int gcd (int a, int b){
14     int c = a % b;
15     while(c != 0){
16         a = b;
17         b = c;
18         c = a % b;
19     }
20     return b;
21 }
```

7.2. LCM

```
1  //Entrada: Dos enteros a y b
2  //Salida: El minimo comun multiplo entre a y b
3  int lcm (int a, int b) {
4      return a / gcd (a, b) * b;
5  }
```

7.3. Exponenciación rapida

```
1  long long int pow(int b, int e){
2      if(!e) return 1;
3
4      long long int ans = pow(b, e>>1);
5      ans *= ans;
```

```
6
7       if(e&1)
8           ans *= b;
9       return ans;
10  }
11
12  unsigned long long int modpow(long long int b, int e, int m){
13      if(!e) return 1;
14
15      unsigned long long int ans = modpow(b, e>>1, m);
16      ans = (ans * ans) % m;// cuidado con desborde en la multiplicacion
17
18      if(e&1)
19          ans = (ans * b) % m;
20      return ans;
21  }
22  //podemos utilizar una idea similar para evitar el desborde en la multiplicacion
23  int multi(long long a, long long b, long long mod){
24      if(b == 0) return 0;
25      long long int ans = (multi( a, b / 2, mod ) * 2) % mod;
26      if( b % 2 == 1 ) ans = (ans + a) % mod;
27      return ans;
28  }
```

7.4. Criba de Erathostenes

```
1   #include <vector>
2   #include <bitset>
3
4   #define MAXPRIME 3162300 // raiz de maximo numero que manejaremos
5
6   using namespace std;
7
8   bitset<MAXPRIME> criba;
9   vector<long long int> primos;
10
11  void llena_criba()
12  {
13      criba.set();// marcamos todo como false
14      for (int i = 2; i < MAXPRIME; i++)
15          if (criba.test( i ))
16          {
17              primos.push_back( i );
18              for (int j= i + i; j<MAXPRIME; j += i)
19                  criba.reset( j );
20          }
21  }
```

7.5. Triangulo de Pascal

```
1   #define MAXN 100 // largest n or m
2
3   int n,m;
4   long binomial_coefficient(n,m){
```

```
    int i,j;
    long bc[MAXN][MAXN];
    for (i=0; i<=n; i++) bc[i][0] = bc[i][i]  = 1;
    for (i=1; i<=n; i++)
        for (j=1; j<i; j++)
            bc[i][j] = bc[i-1][j-1] + bc[i-1][j];
    return bc[n][m];

}
```

7.6. Combinaciones(Para numeros muy grandes)

```
unsigned long long C(int n, int k) {
    k = min(k,n-k);
    unsigned long long res = 1;
    for(int i = 0; i < k; i++)
    {
        res *= (n-i);
        res /= (i+1);
    }
    return res;
}
```

7.7. Polinomios

```
class Polinomio {
    double[] coef;

    // a0 + a1x + a2x^2 + ...
    public Polinomio(double[] coef) {
        this.coef = coef;
    }

    public double integrar(double a, double b) {
        Polinomio integ = integral();
        return integ.evaluar(b) - integ.evaluar(a);
    }

    public double evaluar(double x) {
        double result = 0;
        for (int i = coef.length - 1; i >= 0; i--)
            result = coef[i] + x*result;
        return result;
    }

    public Polinomio integral() {
        double[] newCoef = new double[coef.length + 1];
        newCoef[0] = 0;
        for (int i = 1; i < newCoef.length; i++)
            newCoef[i] = coef[i-1]/i;
        return new Polinomio(newCoef);
    }

    public Polinomio cuadrado() {
```

```
30          double[] newCoef = new double[coef.length * 2 - 1];
31          for (int i = 0; i < newCoef.length; i++)
32              for (int j = 0; j <= i; j++)
33                  if (j < coef.length && i - j < coef.length)
34                      newCoef[i] += coef[j] * coef[i-j];
35
36          return new Polinomio(newCoef);
37      }
38
39      public Polinomio multiplicar(double c) {
40          double[] newCoef = new double[coef.length];
41          for (int i = 0; i < newCoef.length; i++)
42              newCoef[i] = coef[i] * c;
43          return new Polinomio(newCoef);
44      }
45  }
```

7.8. Fibonacci ($O(log(n))$)

```
1  //Entrada: Un entero n
2  //Salida: El n-esimo fibonacci en tiempo O(log(n))
3  long long int fibo(int n) {
4      long long int a,b,x,y,tmp;
5      a = x = tmp = 0;
6      b = y = 1;
7      while(n!=0)
8          if(n%2 == 0) {
9              tmp = a*a + b*b;
10             b = b*a + a*b + b*b;
11             a = tmp;
12             n = n/2;
13         } else {
14             tmp = a*x + b*y;
15             y = b*x + a*y + b*y;
16             x = tmp;
17             n = n-1;
18         }
19     // x = fibo(n-1)
20     return y;// y = fibo(n)
21 }
```

7.9. Multiplicación entero por cadena

```
1  //Entrada: Una cadena y un numero entero
2  //Salida: La multiplicacion de la cadena con el numero
3  #include <iostream>
4  #include <string.h>
5
6  using namespace std;
7
8  unsigned int c,i,n;
9  char res[1000];
10
11 void multi(int n){
```

```
12      for(c = i = 0; res[i]; i++){
13          c = (res[ i ] - '0') * n + c;
14          res[ i ] = c % 10 + '0';
15          c /= 10;
16      }
17      while( c ){
18          res[ i++ ] = '0' + c % 10;
19          c /= 10;
20      }
21      res[i] = 0;
22  }
23
24  int main(){
25      string cad;
26      int m;
27
28      cin >> cad;
29      cad = string(cad.rbegin(), cad.rend());
30      strcpy(res, cad.c_str());
31
32      multi(m);
33      string r(res);
34      string sal(r.rbegin(), r.rend());
35
36      cout << sal << endl;
37      return 0;
38  }
```

7.10. Multiplicación de numeros grandes (Karatsuba)

```
1  //Entrada: 2 BigInteger's x e y
2  //Salida: La multplicacion de x e y
3  import java.math.BigInteger;
4  class Karatsuba {
5      private final static BigInteger ZERO = new BigInteger("0");
6      public static BigInteger karatsuba(BigInteger x, BigInteger y) {
7          int N = Math.max(x.bitLength(), y.bitLength());
8          if (N <= 2000) return x.multiply(y);
9          N = (N / 2) + (N % 2);
10         BigInteger b = x.shiftRight(N);
11         BigInteger a = x.subtract(b.shiftLeft(N));
12         BigInteger d = y.shiftRight(N);
13         BigInteger c = y.subtract(d.shiftLeft(N));
14         BigInteger ac = karatsuba(a, c);
15         BigInteger bd = karatsuba(b, d);
16         BigInteger abcd = karatsuba(a.add(b), c.add(d));
17         return  ac.add(abcd.subtract(ac).subtract(bd).shiftLeft(N)).add(bd.
               shiftLeft(2*N));
18     }
19 }
```

7.11. Integracion de Simpson

```
1  double a[12];
```

```
2
3  int n;
4
5  double f(double x){
6      double ans = 0;
7      for(int i = 0; i <= n; i++)
8          ans += (a[i] * pow(x,i));
9      return ans*ans*PI;
10 }
11 double S(double a, double b, double c, double h){
12     return (h / 3.0) * ( f(a) + 4.0 * f(c) + f(b));
13 }
14 double SIMP(double a, double b, double E ){
15     double h = (b-a)/2.0,
16            c = (b+a)/2.0;
17
18     double a1 = a,  b1 = c,
19            a2 = c,  b2 = b;
20
21     double c1 = (b1+a1)/2.0,
22            c2 = (b2+a2)/2.0;
23
24     double L = (S(a1,b1, c1,h) + S(a2,b2,c2,h)) / 2.0;
25     if(0.1*abs(L-S(a,b,c,h)) < E)
26         return L;
27     else
28         return SIMP(a1,b1, E/2.0) + SIMP(a2,b2, E/2.0);
29 }
```

7.12. Inverso modular

Si queremos hallar $a/b(mod\,p)$ donde p es $primo$, es lo mismo que:
$(a * b^{-1})(mod\,p)$
donde b^{-1} es el inverso modular de b .
Para hallar el inverso modular de un numero $x$modulo p:
$x^{-1}(mod\,p) = modpow(x, p-2, p)$
Entonces, volviendo al problema:
$a/b(mod\,p) = (a * modpow(b, p-2, p)$

7.13. Fraccion

```
1
2  ////////////////////////////////////////////////////////////////////
3  //Fraction.
4  ////////////////////////////////////////////////////////////////////
5  using namespace std;
6
7  typedef long long Num;
8
9  Num gcd(Num a, Num b) {
10     Num temp;
11     if (a < b) {
12         temp = a;
13         a = b;
14         b = temp;
15     }
```

```
16      do {
17          temp = a % b;
18          a = b;
19          b = temp;
20      } while (b != 0);
21      return a;
22  }
23
24  struct Fraction {
25      Num up;
26      Num down;
27
28      Fraction(Num _up = 1, Num _down = 1) {
29          up = _up;
30          down = _down;
31      }
32
33      void reduce() {
34          if (up != 0) {
35              Num div = up > 0? gcd(up, down): gcd(-up, down);
36              up /= div;
37              down /= div;
38          }
39          else {
40              down = 1;
41          }
42      }
43
44      Fraction operator + (const Fraction& add) const {
45          Fraction result(up * add.down + down * add.up, down * add.down);
46          result.reduce();
47          return result;
48      }
49
50      Fraction operator - (const Fraction& sub) const {
51          Fraction result(up * sub.down - down * sub.up, down * sub.down);
52          result.reduce();
53          return result;
54      }
55
56      Fraction operator * (const Fraction& mul) const {
57          Fraction result(up * mul.up, down * mul.down);
58          result.reduce();
59          return result;
60      }
61
62      Fraction operator / (const Fraction& div) const {
63          Fraction result(up * div.down, down * div.up);
64          result.reduce();
65          return result;
66      }
67
68      bool operator == (Num i) const {
69          return up % down == 0 && up / down == i;
70      }
71
72      bool operator != (Num i) const {
73          return !(*this == i);
```

```
74      }
75
76      void operator = (Num i) {
77          up = i;
78          down = 1;
79      }
80  };
81
82  //Test suite.
83  #include <iostream>
84
85  ostream& operator << (ostream& os, const Fraction& f) {
86      if (f.down != 1) {
87          os << f.up << "/" << f.down;
88      }
89      else {
90          os << f.up;
91      }
92      return os;
93  }
94
95  int main () {
96      typedef Fraction F;
97      Fraction f[] = {F(2, 3), F(5, 7), F(9, 6), F(3, 4), F(125, 63)};
98      Fraction result = (f[0] + f[1] * f[2]) / f[3];
99      cout << result << " ";
100     result = result - f[4];
101     cout << result << " " << (result == 1) << " " << (result != 0) << endl;
102     //Correct output: 146/63 1/3 0 1;
103     return 0;
104 }
```

7.14. Matriz

```
1   #define MOD 1000000007
2   #define MAXN 16
3
4   int size;
5
6   struct Matrix {
7     int X[MAXN][MAXN];
8     Matrix () {}
9     Matrix (int k) {
10      memset(X, 0, sizeof(X));
11
12      for(int i=0; i<size; i++)
13        X[i][i] = k;
14    }
15  };
16
17  Matrix operator *(Matrix &A, Matrix &B) {
18    Matrix M;
19
20    for(int i=0; i<size; i++) {
21      for(int j=0; j<size; j++) {
22        long long tmp = 0;
23        for(int k=0; k<size; k++)
```

```
        tmp += (long long)A.X[i][k] * B.X[k][j];
      M.X[i][j] = tmp % MOD;
    }
  }
  return M;
}
Matrix pow(Matrix x, long long n) {
  Matrix P(1);
  while(n) {
    if(n & 1) P = P * x;

    n >>= 1;
    x = x * x;
  }
  return P;
}
long long modpow(long long x, long long n) {
  long long P = 1;

  while(n) {
    if(n & 1) P = P * x % MOD;
    n >>= 1;
    x = x * x % MOD;
  }
  return P;
}
```

7.15. Gauss Jordan

```
class fraction {public:
   L n, d;
   fraction(L n,L d) {
      this->n=n;
      this->d=d;
      simplify();
   }
   void simplify(){
      L g=__gcd(abs(n),abs(d));
      n/=g;
      d/=g;
      if(n==0) d=1;
      if(n<1 && d<1) {
         n = abs(n);
         d = abs(d);
      } else if (n<1 || d<1) {
         n=abs(n);
         d=abs(d);
         n =-n;
      }
   }
   fraction inverse(){
      return fraction(d,n);
   }
   fraction() {
      n=0;
      d=1;
   }
```

```
29     fraction operator/(const fraction& f) {
30        fraction ret(n*f.d,d*f.n);
31        return ret;
32     }
33     fraction operator*(const fraction& f) {
34        fraction ret(n*f.n,d*f.d);
35        return ret;
36     }
37     fraction operator+(const fraction& f) {
38        fraction ret((n*f.d)+(f.n*d),d*f.d);
39        return ret;
40     }
41     fraction operator*(L x) {
42        fraction ret(x*n,d);
43        return ret;
44     }
45     fraction operator-() {
46        fraction ret(-n,d);
47        return ret;
48     }
49     fraction operator-(const fraction& f) {
50        fraction ret((n*f.d)-(f.n*d),d*f.d);
51        return ret;
52     }
53     bool operator>(const fraction& f) {
54        return abs(n*f.d)>abs(f.n*d);
55     }
56
57     fraction read(){
58        char c;
59        scanf("%lld",&n);
60        d = 1;
61        scanf("%c",&c);
62        if(c=='/')
63           scanf("%lld",&d);
64     }
65     void print(){
66        if(d==1) {
67           printf("%lld\n",n);
68        } else {
69           printf("%lld/%lld\n",n,d);
70        }
71     }
72  };
73
74  bool hasSol;
75  int rankA,rankAC;
76
77  const int MX = 59;
78  vector<fraction> mat[MX];
79  fraction x[MX];
80  int N,Y,X,nunk;
81
82  void gauss() {
83     for(int p=1;p<=nunk;p++) {
84        int maxR=p;
85        for(int y=p+1;y<=Y;y++)
86           if(mat[y][p] > mat[maxR][p])
```

```
87              maxR=y;
88          swap(mat[p],mat[maxR]);
89
90          if(mat[p][p].n == 0)
91              continue;
92
93          fraction t = mat[p][p].inverse();
94          for(int i=1;i <= X;i++)
95              mat[p][i] = mat[p][i] * t;
96          for(int y=p+1;y<=Y;y++){
97              t = -mat[y][p];
98              for(int j=1;j <= X;j++) {
99                  mat[y][j] = mat[y][j] + (t*mat[p][j]);
100             }
101         }
102     }
103 }
104
105 void rank() {
106     rankA=rankAC=Y;
107     bool allZeroes;
108     for(int y=1;y<=Y;y++) {
109         allZeroes=true;
110         for(int j=1;allZeroes and j <= nunk;j++)
111             if(mat[y][j].n != 0)
112                 allZeroes=false;
113
114         if(allZeroes) {
115             rankA--;
116             if(mat[y][nunk+1].n == 0)
117                 rankAC--;
118         }
119     }
120 }
121
122 void resuelva(){
123     L num,den;
124     char c;
125
126     scanf("%d %d",&nunk,&Y);
127     X = nunk + 1;
128     for(int y=1;y <= Y;y++)
129         for(int x=1;x <= X;x++)
130             mat[y][x].read();
131
132     hasSol=true;
133     gauss();
134     rank();
135
136     if(rankAC != rankA){
137         printf("No Solution.\n");
138     } else {
139         if(rankAC<nunk)
140             printf("Infinitely many solutions containing %d arbitrary constants.\n"
                    ,nunk-rankAC);
141         else{
142             for(int p=nunk;p>=1;--p) {
143                 fraction s(0,1);
```

```
144                 for(int k=p+1;k<=nunk;++k)
145                     s = s + (x[k]*mat[p][k]);
146                 x[p] = (mat[p][X]-s) / mat[p][p];
147             }
148             for(int y=1;y<=nunk;y++)
149                 printf("x[%d] = ",y),x[y].print();
150         }
151     }
152 }
153 int main() {
154     int nc;
155     bool first=true;
156 
157     for(int i=0;i<MX;i++)
158         mat[i].resize(MX);
159 
160     while(scanf("%d",&N) and N>0) {
161         if(first == false) printf("\n");first = false;
162         printf("Solution for Matrix System # %d\n",N);
163         resuelva();
164     }
165 }
```

Capítulo 8

Cadenas

8.1. Utilidades

```
#include <iostream>
#include <algorithm>

using namespace std;

bool is_palindrome(string &ss){
    return equal(ss.begin(), ss.end(), ss.rbegin());
}

int main(int argc, char const *argv[])
{

    return 0;
}
```

8.2. Boyer Moore

Encuentra todos los match de un patron en el texto.

```
#include <iostream>
#include <cstring>

using namespace std;

int M, N;

//M tamanio del patron
//N tamanio del texto

void preBM(char P[], int bmNext[])
{
    for(int i = 0; i <= 255; i++)
        bmNext[i] = M;
    for(int i = 0; i < M; i++)
        bmNext[P[i]] = M - 1 - i ;
}

void Boyer_Moore(char T[], char P[])
{

```

```
22      int i = M - 1;
23      int j = M - 1;
24      int bmNext[255];
25      preBM(P,bmNext);
26      while((i < N) && (j >= 0))
27      {
28          if(T[i] == P[j])
29          {
30              i--;
31              j--;
32          }
33          else
34          {
35              i += bmNext[T[i]];
36              j = M - 1;
37          }
38          if(j < 0)
39          {
40              cout<<"Match en: "<<(i + 1)<<endl;
41              i += M + 1;
42              j = M - 1;
43          }
44      }
45  }
46
47  int main()
48  {
49      char texto[100];
50      char patron[100];
51      cin>>texto;
52      cin>>patron;
53      M=strlen(patron);
54      N=strlen(texto);
55      Boyer_Moore(texto,patron);
56      return 0;
57  }
```

8.3. Knuth Morris Pratt

```
1   //Entrada: Una cadena S y el patron k
2   //Salida: El numero de ocurrencia del patron k en S
3   int KMP(string S, string K)
4   {
5       vector<int> T(K.size() + 1, -1);
6       for(int i = 1; i <= K.size(); i++)
7       {
8           int pos = T[i - 1];
9           while(pos != -1 && K[pos] != K[i - 1]) pos = T[pos];
10          T[i] = pos + 1;
11      }
12      vector<int> matches;
13      int sp = 0;
14      int kp = 0;
15      while(sp < S.size())
16      {
17          while(kp != -1 && (kp == K.size() || K[kp] != S[sp])) kp = T[kp];
18          kp++;
```

```
19          sp++;
20          if(kp == K.size()) matches.push_back(sp - K.size());
21      }
22      /* En el vector matches se guardan los Ãndices a cada una de las
           ocurrencias, por tanto el tamanio de dicho vector nos dirÃ¡ cuÃ¡ntas
           ocurrencias se han encontrado. */
23      return matches.size();
24  }
```

8.4. Iesima permutación

```
1   //Entrada: Una cadena cad(std::string), un long th
2   //Salida : La th-esima permutacion lexicografica de cad
3   string ipermutacion(string cad, long long int th){
4       sort(cad.begin(), cad.end());
5       string sol = "";
6       int pos;
7       for(int c = cad.size() - 1; c >= 0; c--){
8           pos = th / fact[c];
9           th %= fact[c];
10          sol += cad[pos];
11          cad.erase(cad.begin() + pos);
12      }
13      return sol;
14  }
```

8.5. Iesima permutación

```
1   //Entrada: Una cadena cad(std::string), un long th
2   //Salida : La th-esima permutacion lexicografica de cad
3   string ipermutacion(string cad, long long int th){
4       sort(cad.begin(), cad.end());
5       string sol = "";
6       int pos;
7       for(int c = cad.size() - 1; c >= 0; c--){
8           pos = th / fact[c];
9           th %= fact[c];
10          sol += cad[pos];
11          cad.erase(cad.begin() + pos);
12      }
13      return sol;
14  }
```

8.6. Algoritmo de Manacher

El algoritmo guarda el tamaño del palindrome mas grande en LP[pos] que tiene como centro las posición *pos* en la cadena original

```
1   #include <iostream>
2   #include <vector>
3   #include <fstream>
4   #include <string.h>
5   #define MAXN 20100
```

```
6
7  using namespace std;
8  int N,i,j,izq,der,posi,dist;
9  string cad,tmp;
10 char v[MAXN];
11 int pos[MAXN];
12
13 void todoslosPalindromos(vector<int> &LP, int par){
14     izq = der = -1;
15     for(posi = 0; posi < N; posi++){
16         dist = 0;
17         if(posi <= der)
18             dist = min(der - posi + par, LP[izq+der-(posi+par)]) + 1;
19         while(posi-dist>=0 && posi+dist-par<N
20                 && v[posi-dist]==v[posi+dist-par])
21             dist++;
22         LP[posi] = --dist;
23         if(posi+dist-par > der){
24             der = posi + dist - 1;
25             izq = posi - dist ;
26         }
27     }
28
29 }
30 int main() {
31     getline(fin,cad);
32     while(getline(fin,tmp))
33         cad +='\n'+tmp;
34     N = 0;
35     for(i = 0; i < cad.length(); i++)
36         if(isalpha(cad[i])){
37             v[N] = tolower(cad[i]);
38             pos[N] = i;
39             N++;
40         }
41     v[N] = '\0';
42     vector<int> LP(N,0),LI(N,0);
43     todoslosPalindromos(LP,1);
44     todoslosPalindromos(LI,0);
45     int SOL1 = 0,SOL2 = 0;
46     //primero en los de tamanio impar
47     for(int i = 0; i < N; i++)
48         if(LI[i]*2+1 > LI[SOL1]*2+1)
49             SOL1 = i;
50     //luego en los de tamanio par
51     for(int i = 0; i < N; i++)
52         if(LP[i]*2 > LP[SOL2]*2)
53             SOL2 = i;
54
55     if(LI[SOL1]*2+1 > LP[SOL2]*2){
56         izq = SOL1 - LI[SOL1];
57         der = SOL1 + LI[SOL1];
58     }else{
59         izq = SOL2 - LP[SOL2];
60         der = SOL2 + LP[SOL2]-1;
61     }
62
63     fout << der - izq + 1 <<   endl;
```

```
64
65      izq = pos[izq];
66      der = pos[der];
67      fout << cad.substr(izq,der-izq+1)<<endl;
68      return 0;
69  }
```

8.7. Longest Common Subsequence (LCS)

```
1   #include <iostream>
2   #include <string>
3   #include <algorithm>
4   using namespace std;
5
6   int longestCommonSubsequence(const string& a, const string& b) {
7     int A = a.size(), B = b.size();
8     int L[2][B + 1];
9     for (int i = 0; i <= 1; ++i) L[i][0] = 0;
10    for (int i = 0; i <= B; ++i) L[0][i] = 0;
11    for (int i = 1; i <= A; ++i) {
12      int this_i = i % 2, pre_i = this_i ? 0 : 1;
13      for (int j = 1; j <= B; ++j) {
14        if (a[i - 1] == b[j - 1]) L[this_i][j] = 1 + L[pre_i][j - 1];
15        else L[this_i][j] = max(L[pre_i][j], L[this_i][j - 1]);
16      }
17    }
18    return max(L[0][B], L[1][B]);
19  }
20
21  int main() {
22    string a, b;
23    cin >> a >> b;
24    cout << longestCommonSubsequence(a, b);
25  }
```

8.8. Suffix Array

```
1   #include<cstdio>
2   #include<cstdlib>
3   #include<iostream>
4   #include<sstream>
5   #include<cmath>
6   #include<string>
7   #include<cstring>
8   #include<cctype>
9   #include<algorithm>
10  #include<vector>
11  #include<bitset>
12  #include<queue>
13  #include<stack>
14  #include<utility>
15  #include<list>
16  #include<set>
17  #include<map>
```

```
18 using namespace std;
19 #define MAXN 1000005
20 int n,t;  //n es el tamanio de la cadena
21 int p[MAXN],r[MAXN],h[MAXN];
22 //p es el inverso del suffix array, no usa indices del suffix array ordenado
23 //h el el tamanio del lcp entre el i-esimo y el i+1-esimo elemento de suffix
       array ordenado
24 //r indices sufijo ordenado
25 string s;
26 void fix_index(int *b, int *e) {
27    int pkm1, pk, np, i, d, m;
28    pkm1 = p[*b + t];
29    m = e - b; d = 0;
30    np = b - r;
31    for(i = 0; i < m; i++) {
32       if (((pk = p[*b+t]) != pkm1) && !(np <= pkm1 && pk < np+m)) {
33          pkm1 = pk;
34          d = i;
35       }
36       p[*(b++)] = np + d;
37    }
38 }
39
40 bool comp(int i, int j) {
41    return p[i + t] < p[j + t];
42 }
43 void suff_arr() {
44    int i, j, bc[256];
45    t = 1;
46    for(i = 0; i < 256; i++) bc[i] = 0;  //alfabeto
47    for(i = 0; i < n; i++) ++bc[int(s[i])]; //counting sort inicial del alfabeto
48    for(i = 1; i < 256; i++) bc[i] += bc[i - 1];
49    for(i = 0; i < n; i++) r[--bc[int(s[i])]] = i;
50    for(i = n - 1; i >= 0; i--) p[i] = bc[int(s[i])];
51    for(t = 1; t < n; t *= 2) {
52       for(i = 0, j = 1; i < n; i = j++) {
53          while(j < n && p[r[j]] == p[r[i]]) ++j;
54          if (j - i > 1) {
55             sort(r + i, r + j, comp);
56             fix_index(r + i, r + j);
57          }
58       }
59    }
60 }
61
62 void lcp() {
63    int tam = 0, i, j;
64    for(i = 0; i < n; i++)if (p[i] > 0) {
65       j = r[p[i] - 1];
66       while(s[i + tam] == s[j + tam]) ++tam;
67       h[p[i] - 1] = tam;
68       if (tam > 0) --tam;
69    }
70    h[n - 1] = 0;
71 }
72
73 int main(){
74    s="mississippi$";
```

```
   n=s.size();
   suff_arr();
   lcp();
   for(int i=0;i<n;i++)cout<<r[i]<<" ";cout<<endl;
   for(int i=0;i<n;i++)cout<<h[i]<<" ";cout<<endl;
   return 0;
}
 //rotacion menor lexicografica
 /*
 int tam1=strlen(s);
      for(int i=tam1;i<2*tam1;i++)s[i]=s[i-tam1];
      n=2*tam1;
      suff_arr();
      char dev[tam1];
      for(int i=0;i<n;i++)
         if(r[i]<tam1){
            for(int j=r[i];j<r[i]+tam1;j++)
               dev[j-r[i]]=s[j];
            break;
         }
      for(int i=0;i<tam1;i++)
      printf("%c",dev[i]);

 */
/*
ACM 2009 File Recover   Solucion
Problema: Contar los substrings q se repiten al menos una vez.
Analisis: Notamos que si el lcp(i,i+1) con lcp(i+1,i+2)  aumenta
quiere decir que encontramos h[i+1]-h[i] palabras nuevas (prefijos)
SUM(max(h[i+1]-h[i],0))
*/
```

8.9. Suffix Array DC3

```
//from http://blog.csdn.net/acdreamers/article/details/10746023
#include <iostream>
#include <string.h>
#include <algorithm>
#include <stdio.h>

using namespace std;
const int N=250005;

struct State
{
    State *pre,*go[26];
    int step;
    void clear()
    {
        pre=0;
        step=0;
        memset(go,0,sizeof(go));
    }
}*root,*last;

State statePool[N*2],*cur;

```

```
24 void init()
25 {
26     cur=statePool;
27     root=last=cur++;
28     root->clear();
29 }
30
31 void Insert(int w)
32 {
33     State *p=last;
34     State *np=cur++;
35     np->clear();
36     np->step=p->step+1;
37     while(p&&!p->go[w])
38         p->go[w]=np,p=p->pre;
39     if(p==0)
40         np->pre=root;
41     else
42     {
43         State *q=p->go[w];
44         if(p->step+1==q->step)
45             np->pre=q;
46         else
47         {
48             State *nq=cur++;
49             nq->clear();
50             memcpy(nq->go,q->go,sizeof(q->go));
51             nq->step=p->step+1;
52             nq->pre=q->pre;
53             q->pre=nq;
54             np->pre=nq;
55             while(p&&p->go[w]==q)
56                 p->go[w]=nq, p=p->pre;
57         }
58     }
59     last=np;
60 }
61
62 char A[N],B[N];
63
64 int main()
65 {
66     int n,m;
67     scanf("%s%s",A,B);
68     n=strlen(A);
69     m=strlen(B);
70     init();
71     for(int i=0; i<n; i++)
72         Insert(A[i]-'a');
73     int ans=0,len=0;
74     State *p=root;
75     for(int i=0; i<m; i++)
76     {
77         int x=B[i]-'a';
78         if(p->go[x])
79         {
80             len++;
81             p=p->go[x];
```

```
        }
        else
        {
            while(p&&!p->go[x]) p=p->pre;
            if(!p) p=root,len=0;
            else   len=p->step+1,p=p->go[x];
        }
        ans=max(ans,len);
    }
    printf("%d\n",ans);
    return 0;
}
```

8.10. Trie

```
#include <iostream>
#include <vector>
using namespace std;
struct Node{
    char content;
    bool wordMarker;
    vector<Node*> children;
    Node() { content = ' '; wordMarker = false; }
    void setContent(char c) { content = c; }
    void setWordMarker() { wordMarker = true; }
    Node* findChild(char c);
    void appendChild(Node* child) { children.push_back(child); }
};


struct Trie {
    Node* root;
    Trie(){root=new Node();}
    void addWord(string s);
    bool searchWord(string s);
    void deleteWord(string s);
};

Node* Node::findChild(char c){
    for ( int i = 0; i < children.size(); i++ ){
        Node* tmp = children.at(i);
        if ( tmp->content == c )
        {
            return tmp;
        }
    }

    return NULL;
}


void Trie::addWord(string s){
    Node* current = root;

    if ( s.length() == 0 ){
        current->setWordMarker();
        return;
```

```
43      }
44
45      for ( int i = 0; i < s.length(); i++ ){
46          Node* child = current->findChild(s[i]);
47          if ( child != NULL ){
48              current = child;
49          }else{
50              Node* tmp = new Node();
51              tmp->setContent(s[i]);
52              current->appendChild(tmp);
53              current = tmp;
54          }
55          if ( i == s.length() - 1 )
56              current->setWordMarker();
57      }
58  }
59
60
61  bool Trie::searchWord(string s)
62  {
63      Node* current = root;
64
65      while ( current != NULL )
66      {
67          for ( int i = 0; i < s.length(); i++ )
68          {
69              Node* tmp = current->findChild(s[i]);
70              if ( tmp == NULL )
71                  return false;
72              current = tmp;
73          }
74
75          if ( current->wordMarker )
76              return true;
77          else
78              return false;
79      }
80
81      return false;
82  }
83
84
85  int main(){
86      Trie* trie = new Trie();
87      trie->addWord("algo");
88      trie->addWord("ala");
89      trie->addWord("abeja");
90      trie->addWord("abel");
91      trie->addWord("trie");
92
93      if ( trie->searchWord("abel") )
94          cout << "Encontrado" << endl;
95      else cout << "No encontrado" << endl;
96
97
98      delete trie;
99  }
```

Capítulo 9

Geometria Computacional

9.1. Intersección de rectangulos

```
#include <iostream>
#include <vector>
#include <fstream>

using namespace std;
#define MAXC 2501

struct Rect{
    int x1,y1, x2,y2;
    int color;
    int area;
    Rect(int _x1, int _y1, int _x2, int _y2){
        x1 = _x1;
        y1 = _y1;
        x2 = _x2;
        y2 = _y2;
        getArea();
    }
    int getArea(){
        if(x1>=x2 || y1>=y2)return area = 0;
        return area = (x2-x1)*(y2-y1);
    }
};

Rect interseccion(Rect t, Rect r){
    int x1,y1,x2,y2;
    x1 = max(t.x1,r.x1);
    y1 = max(t.y1,r.y1);
    x2 = min(t.x2,r.x2);
    y2 = min(t.y2,r.y2);
    Rect res(x1,y1,x2,y2);
    return res;
}
```

9.2. Distancia Punto - Segmento

```
struct point{
    double x,y;
};
```

```
4  inline double dist(const point &a, const point &b){
5      return sqrt((a.x-b.x)*(a.x-b.x) + (a.y-b.y)*(a.y-b.y));
6  }
7  inline double distsqr(const point &a, const point &b){
8      return (a.x-b.x)*(a.x-b.x) + (a.y-b.y)*(a.y-b.y);
9  }
10 double distance_point_to_segment(const point &a, const point &b, const point &
       pnt){
11     double u = ((pnt.x - a.x)*(b.x - a.x) + (pnt.y - a.y)*(b.y - a.y)) / distsqr
           (a, b);
12     point intersection;
13     intersection.x = a.x + u*(b.x - a.x);
14     intersection.y = a.y + u*(b.y - a.y);
15
16     if (u < 0.0 || u > 1.0)
17         return min(dist(a, pnt), dist(b, pnt));
18
19     return dist(pnt, intersection);
20 }
```

9.3. Distancia Punto - Recta

```
1  double distance_point_to_line(const point &a, const point &b, const point &pnt){
2      double u = ((pnt.x - a.x)*(b.x - a.x) + (pnt.y - a.y)*(b.y - a.y)) / distsqr
           (a, b);
3      point intersection;
4      intersection.x = a.x + u*(b.x - a.x);
5      intersection.y = a.y + u*(b.y - a.y);
6      return dist(pnt, intersection);
7  }
```

Capítulo 10

Utilitarios

10.1. Plantilla

```
#include <algorithm>
#include <iostream>
#include <iterator>
#include <cassert>
#include <sstream>
#include <fstream>
#include <cstdlib>
#include <cstring>
#include <utility>
#include <complex>
#include <string>
#include <cctype>
#include <cstdio>
#include <vector>
#include <bitset>
#include <stack>
#include <queue>
#include <cmath>
#include <deque>
#include <list>
#include <set>
#include <map>

#define ll long long
#define lld long long int
#define sc scanf
#define pf printf
#define pi 2*acos(0.0)
#define f first
#define s second
#define sz size()
#define mp make_pair
#define r(input) freopen("2966.in","r",stdin)
#define w(output) freopen("output.txt","w",stdout)
#define maxall(v) *max_element(v.begin(),v.end())
#define minall(v) *min_element(v.begin(),v.end())
#define Sort(v) sort(v.begin(),v.end())
#define un(v) Sort(v), v.erase(unique(v.begin(),v.end()),v.end())
#define clr(a)  memset(a,0,sizeof(a))
#define pb push_back
#define lim 100000
```

10.2. Lectura rapida

```cpp
#include <cstdio>

using namespace std;

char line[400000];//OJO maximo mas 3 para el '\n' y el '\0'
int now = 0;
inline int getInt(){
    int n;
    while(1)
        if(line[now]!=0){
            if(line[now]<'0' || line[now]>'9'){
                now++;
                continue;
            }
            n = 0;
            while(line[now]>='0' && line[now]<='9') {
                n = n*10 + line[now] - '0';
                now++;
            }
            return n;
        }
        else{
            gets(line);
            now = 0;
        }
    return n;
}
```

10.3. Espicificadores de formato para printf y scanf

Especificador	Tipo
%c	char
%s	cadena
%d, %i	int
%u	unsigned int
%ld	long int
%lu	unsigned long int
%lld	long long int
%llu	unsigned long long int
%e, %f, %g	float
%lf	double
%o	número octal
%x	número hexadecimal
%patron	patron

10.4. Contar bits en 1 en un numero

C++ :

```cpp
__builtin_popcount(unsigned int)
__builtin_popcountll(unsigned long long)
```

Java :

```
1 Integer.bitCount(int)
```

10.5. Busqueda binaria

C++'s Standard Template Library has four functions for binary search, depending on what information you want to get. They all need

```
#include <algorithm>
```

The **lower_bound()** function returns an iterator to the first position where a value could be inserted without violating the order; i.e. the first element equal to the element you want, or the place where it would be inserted.

```
int *ptr = std::lower_bound(array, array+len, what);
// a custom comparator can be given as fourth arg
```

The **upper_bound()** function returns an iterator to the last position where a value could be inserted without violating the order; i.e. one past the last element equal to the element you want, or the place where it would be inserted.

```
int *ptr = std::upper_bound(array, array+len, what);
// a custom comparator can be given as fourth arg
```

The **equal_range()** function returns a pair of the results of lower_bound() and upper_bound().

```
std::pair<int *, int *> bounds = std::equal_range(array, array+len, what);
// a custom comparator can be given as fourth arg
```

Note that the difference between the bounds is the number of elements equal to the element you want.

The **binary_search()** function returns true or false for whether an element equal to the one you want exists in the array. It does not give you any information as to where it is.

```
bool found = std::binary_search(array, array+len, what);
// a custom comparator can be given as fourth arg
```