Desafios de Programação
Estrutura de Dados

Wladimir Araújo Tavares [1]

[1]Universidade Federal do Ceará - Campus de Quixadá

10 de julho de 2017

STL Vector

Função	Descrição
$resize(n, val)$	redimensiona o vetor para ter n elementos inicializados com o valor val
$push\_back(val)$	adiciona o elemento no final do vetor
$pop\_back()$	remove o último elemento do vetor
$insert(it, val)$	insere o elemento val antes da posição especificada por it
sort(first,last)	ordena os elementos no intervalo [first,last) em ordem crescente

Exemplo: vector

```
#include <vector>
#include <algorithm>
#include <iostream>
#define all(v) v.begin(),v.end()
 using namespace std;
 typedef vector <int> vi;
 int main(){
   vi v1;
   v1.resize(3, 0);//Aloca três posições e inicializa com zero
   v1[0] = 5; v1[1] = 4; v1[2]=6;
   v1.push_back(9);
   vi::iterator it = v1.begin();
   v1.insert(it,7);// insere no começo
   for(int i = 0; i < v1.size(); i++) cout << v1[i] << endl;
   int maior = *max_element(all(v1));
   cout << "maior:␣" << maior << endl;
   vi v2(v1.begin(), v1.begin()+3);
   for(int i = 0; i < v2.size(); i++) cout << v2[i] << endl;
   vi v3(v1); // copy v1
   cout << "sort␣vector" << endl;
   sort(all(v3));
   reverse(all(v3));
   for(int i = 0; i < v3.size(); i++) cout << v3[i] << endl;
 }
```

STL queue

Função	Descrição
push(*val*)	insere o elemento val na fila
back()	retorna uma referência para o elemento mais novo na fila
pop()	remove o elemento mais antigo na fila
front()	retorna uma referência para o elemento mais antigo na fila

Exemplo: queue

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

int main()
{
  queue <int> fila;
  fila.push(2); fila.push(3);
  fila = queue <int> ();
  fila.push(2); fila.push(3); fila.push(4);
  cout << fila.size() << endl;
  cout << fila.back() << endl;
  while( !fila.empty() ){
    cout << fila.front() << "␣";
    fila.pop();
  }
}
```

STL stack

Função	Descrição
push(*val*)	insere o elemento val no topo da pilha
top()	retorna uma referência para o elemento no topo da pilha
pop()	remove o elemento do topo da pilha

Exemplo: stack

```
#include <iostream>
#include <algorithm>
#include <vector>
#include <queue>
#include <stack>
using namespace std;
int main()
{
  stack <int> p;
  p.push(4);
  p.push(5);
  p.push(6);
  while( !p.empty() )
  {
    cout << p.top() << "␣";
    p.pop();
  }


}
```

STL bitset

```
#include <bits/stdc++.h>
using namespace std;
#define M 32
int main()
{
    // default constructor initializes with all bits 0
    bitset<M> bset1;
    // bset2 is initialized with bits of 20
    bitset<M> bset2(20);
    // bset3 is initialized with bits of specified binary string
    bitset<M> bset3(string("1100"));
    // cout prints exact bits representation of bitset
    cout << bset1 << endl;  // 00000000000000000000000000000000
    cout << bset2 << endl;  // 00000000000000000000000000010100
    cout << bset3 << endl;  // 00000000000000000000000000001100
    cout << endl;
    // declaring set8 with capacity of 8 bits
    bitset<8> set8;     // 00000000
    // setting first bit (or 6th index)
    set8[1] = 1;      set8[4] = 1;
    cout << set8 << endl; //  00010010
    // count function returns number of set bits in bitset
    int numberof1 = set8.count();
```

STL bitset

```
// size function returns total number of bits in bitset
// so there difference will give us number of unset(0)
// bits in bitset
int numberof0 = set8.size() - numberof1;
cout << set8 << "␣has␣" << numberof1 << "␣ones␣and␣"
     << numberof0 << "␣zeros\n";
// test function return 1 if bit is set else returns 0
cout << "bool␣representation␣of␣" << set8 << "␣:␣";
for (int i = 0; i < set8.size(); i++)
    cout << set8.test(i) << "␣";
cout << endl;
// any function returns true, if atleast 1 bit
// is set
if (!set8.any())
    cout << "set8␣has␣no␣bit␣set.\n";
if (!bset1.any())
    cout << "bset1␣has␣no␣bit␣set.\n";
// none function returns true, if none of the bit
// is set
if (!bset1.none())
    cout << "bset1␣has␣all␣bit␣set\n";
// bset.set() sets all bits
cout << set8.set() << endl;
//  bset.set(pos, b) makes bset[pos] = b
```

STL bitset

```
// bset.set(pos) makes bset[pos] = 1  i.e. default
    // is 1
    cout << set8.set(4) << endl;

    // reset function makes all bits 0
    cout << set8.reset(2) << endl;
    cout << set8.reset() << endl;

    // flip function flips all bits i.e.  1 <-> 0
    // and  0 <-> 1
    cout << set8.flip(2) << endl;
    cout << set8.flip() << endl;

    // Converting decimal number to binary by using bitset
    int num = 100;
    cout  << "\nDecimal␣number:␣" << num
         << "␣␣Binary␣equivalent:␣" << bitset<8>(num);

    return 0;
}
```

BitMask

```
#include <vector>
#include <stdio.h>
#include <iostream>
#define INT_SIZE (8*sizeof(int))
#define high(x)         ((x)/INT_SIZE )
#define low(x)          ((x)%INT_SIZE)
 using namespace std;
 class BitMask{
  public:
  vector <int> bit;
  vector <unsigned> mask;
  BitMask(int N){
   bit.assign(N/INT_SIZE+1, 0);
   mask.assign(INT_SIZE, 0);
   mask[0] = 1;
   for(int i=1;i<INT_SIZE;i++)
     mask[i] = (1<<i);
  }
  void set(int x){ bit[high(x)] |= mask[low(x)];}
  void reset(int x){ bit[high(x)] &= ~(mask[low(x)]);}
  bool test(int x){ return (bit[high(x)] & mask[low(x)]) != 0 ;}
 };
```

STL set

Função	Descrição
insert(val)	adiciona o elemento val, mas não permite elementos duplicados
erase(val)	remove o elemento val
erase(position)	remove o elemento apontado pelo iterator position
erase(first,last)	remove todos os elementos entre first e last
count(val)	o número de vezes que val aparece no set
find(val)	se existe o elemento val a função devolve seu iterador; caso contrário devolve end()

Exemplo Set

```
#include <iostream>
#include <set>
 using namespace std;
 int main(){
  set<int> myset;
  set<int>::iterator itlow,itup,it;
  for(int i = 1; i <= 10; i++) myset.insert(i);
  myset.insert(8); // nao insere
  itlow=myset.lower_bound (3);
  itup=myset.upper_bound (6);
  myset.erase(itlow,itup); // 1 2 7 8 9 10
  it = myset.find(7); // Complexidade logaritmica
  myset.erase(it); // 1 2 8 9 10
  cout << "size␣of␣set:" << myset.size() << endl;
  if( myset.count(7) == 0) cout << "7␣is␣not␣a␣element␣of␣myset"
  for (it=myset.begin(); it!=myset.end(); ++it) //Percorre em ord
     cout << '␣' << *it;
 }
```

STL map

Função	Descrição
insert(val)	adiciona o elemento val
erase(val)	remove o elemento val
erase(position)	remove o elemento apontado pelo iterator position
erase(first,last)	remove todos os elementos entre first e last
find(val)	se existe o elemento val então a função devolve seu it; caso contrário devolve map::end()

Exemplo map

```
#include <iostream>
#include <algorithm>
#include <vector>
#include <map>
using namespace std;
int main()
{
  char poema [] = "Sou␣chama␣sem␣luz␣jardim␣\
␣␣sem␣luar␣luar␣sem␣amor";
  map <char, int> mapa;
  map<char,int >::iterator it;
  for(int i = 0; poema[i] != '\0'; i++)
  {
    char c = poema[i];
    mapa[c]++;
  }

  for(it = mapa.begin(); it != mapa.end(); it++)
  {
    if( it->second > 0 )
    printf("mapa[%c]␣=␣%d\n", it->first, it->second );
  }
}
```

Exemplo unordered_map

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

int main()
{
  char poema [] = "Sou␣chama␣sem␣luz␣jardim␣\
␣␣sem␣luar␣luar␣sem␣amor";
  unordered_map <char, int> mapa;
  unordered_map <char,int >::iterator it;
  for(int i = 0; poema[i] != '\0'; i++){
    char c = poema[i];
    mapa[c]++;
  }
  for(it = mapa.begin(); it != mapa.end(); it++){
    if( it->second > 0 )
    printf("mapa[%c]␣=␣%d\n", it->first, it->second );
  }
}
```

Exercício 3

Determine se existe dois elementos distintos x,y em L tal que x+y=sum. Seja n ($1 \leq n \leq 10^6$) o tamanho do vetor. Qual é o método mais viável para esse problema.

- Para cada elemento x, faça uma busca binária pelo elemento sum-x.
- Insira cada elemento em uma tabela de dispersão. Antes de inserir, verifique se sum-x está presente na tabela de dispersão.

Leia:

`http://marathoncode.blogspot.com.br/2012/10/sum-problem.html`

Sum problem

```
bool twosum(vector<int> &v, int sum)
{
  int i, j;
  sort(v.begin(),v.end());
  i = 0;
  j = v.size()-1;

  while( i < j)
  {
    if( v[i] + v[j] == sum) return true;
    else if( v[i] + v[j] > sum){
      j--;
    }else{
      i++;
    }
  }
  return false;
}
```

Sum problem

```
bool twosum_with_map(vector<int> &v, int sum)
{
  map <int, int> mapa;
  map <int,int>::iterator it;
  for(int i = 0; i < (int) v.size(); i++)
  {
    it = mapa.find(sum-v[i]);
    if( it != mapa.end() ) return true;
    mapa[v[i]] = i;
  }
}
```

Sum problem

```
bool twosum_with_unordered_map(vector<int> &v, int sum)
{
  unordered_map <int, int> mapa;
  unordered_map <int,int>::iterator it;
  for(int i = 0; i < (int) v.size(); i++)
  {
    it = mapa.find(sum-v[i]);
    if( it != mapa.end() ) return true;
   mapa[v[i]] = i;
  }

}
```

Testes computacionais

```
twosum
Found
time                    =      5.18100
twosum\_with_map
Found
time                    =      0.00200
twosum\_with\_unordered\_map
Found
time                    =      0.00300
```

STL priority_queue

Função	Descrição	Complexidade
empty()	verifica se a fila de prioridade está vazia	O(1)
size()	devolve o número de elementos na estrutura	O(1)
push()	insere um novo elemento na fila de prioridade	O(lg n)
pop()	remove o elemento do topo da fila de prioridade	O(lg n)
top()	devolve o elemento do topo da fila de prioridade	O(1)

Exemplo 1: priority_queue

```cpp
#include <iostream>
#include <algorithm>
#include <queue>
using namespace std;
int main()
{
  //fila de prioridade mínima
  priority_queue <int, vector <int>, greater <int> > pq;
  pq.push(30); pq.push(20);
  pq.push(25); pq.push(40);
  while( !pq.empty() )
  {
    cout << pq.top() << "␣";
    pq.pop();
  }
  cout << endl;
  //20 25 30 40
}
```

Exemplo 2: priority_queue

```
#include <iostream>
#include <algorithm>
#include <queue>
using namespace std;
int main()
{
  //fila de prioridade mínima
  priority_queue <int, vector <int>, greater <int> > pq;
  pq.push(30); pq.push(20);
  pq.push(25); pq.push(40);
  while( !pq.empty() )
  {
    cout << pq.top() << "␣";
    pq.pop();
  }
  cout << endl;
  //20 25 30 40
}
```

Exemplo 3: priority_queue

```
#include <iostream>
#include <algorithm>
#include <queue>
using namespace std;
int main()
{
  //fila de prioridade máxima
  priority_queue <int, vector <int>, less <int> > pq;
  pq.push(30); pq.push(20);
  pq.push(25); pq.push(40);
  while( !pq.empty() )
  {
    cout << pq.top() << "␣";
    pq.pop();
  }
  cout << endl;
  //40 30 25 20
}
```

Exemplo 4: priority_queue

```
#include <iostream>
#include <algorithm>
#include <queue>
using namespace std;
typedef bool (*comp)(int,int);
bool compare(int a, int b)
{
   return (a<b);
}
int main()
{
 int v[] = {10,60,50,20};
 priority_queue <int> pq1(v,v+4); //default less<int>
 priority_queue <int, vector<int>, comp> pq2(compare);
 pq2.push(10); pq2.push(60);
 pq2.push(50); pq2.push(20);
 while( !pq2.empty() )
 {
   cout << pq2.top() << "␣";
   pq2.pop();
 }
 cout << endl;
}
```

Exemplo 5 priority_queue

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

class Human {

    public:
        string name;
        int age;
        Human(string name, int age);
};
Human::Human(string name, int age) : name(name), age(age) {}
bool operator<(Human a, Human b) {return (a.age < b.age);}
int main() {

    Human p1("Child",5);
    Human p2("Grandfather",70);
    priority_queue<Human> Q;
    Q.push(p1);
    Q.push(p2);
}
```

Exemplo 6 priority_queue

```
#include <iostream>
#include <algorithm>
#include <vector>
#include <queue>
#include <time.h>
#include <stdio.h>
using namespace std;
typedef vector<int> vi;
int main(){
  clock_t clk;
  double elapsed;
  default_random_engine generator;
  uniform_int_distribution<int> distribution(1, 1000000000);
  vi v1, v2;
  for(int i = 0; i < 10000000; i++){
    int x = distribution(generator);
    v1.push_back ( x );v2.push_back ( x );
  }
  clk = clock();
  sort(v1.begin(), v1.end(), greater <int>() );
  for(int i = 0; i < 10; i++) cout << v1[i] << "␣";
  cout << endl;
  elapsed = ((double) (clock() - clk)) / CLOCKS_PER_SEC;
  printf("time␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣=␣%10.5f\n", elapsed);
```

Exemplo 6 priority_queue

```
  clk = clock();
  make_heap(v2.begin() , v2.end() );
  for(int i = 0; i < 10; i++){
    cout << v2.front() << "␣";
    std::pop_heap (v2.begin(),v2.end());
    v2.pop_back();
  }
  cout << endl;
  elapsed = ((double) (clock() - clk)) / CLOCKS_PER_SEC;
  printf("time␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣=␣%10.5f\n", elapsed);
  clk = clock();
  priority_queue <int> pq(v1.begin(),v1.end());
  for(int i = 0; i < 10; i++){
    cout << pq.top() << "␣";pq.pop();
  }
  cout << endl;
  elapsed = ((double) (clock() - clk)) / CLOCKS_PER_SEC;
  printf("time␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣=␣%10.5f\n", elapsed);
}
```

Exemplo 6 priority_queue

```
999999945 999999761 999999642 999999323 999999321 999999264
999999229 999999227 999999125 999999055
time                =    6.89800
999999945 999999761 999999642 999999323 999999321 999999264
999999229 999999227 999999125 999999055
time                =    1.16000
999999945 999999761 999999642 999999323 999999321 999999264
999999229 999999227 999999125 999999055
time                =    1.66500
```

Exemplo UnionFind

```
#include <iostream>
#include <vector>
#include <stdio.h>
 using namespace std;
 class UnionFind {
  private:
   vector <int> p, rank, setSize;
   int numSets;
  public:
   UnionFind(int N):numSets(N){
    rank.resize(N,0); p.resize(N,0);setSize.resize(N,1);
    for(int i=0;i<N;i++) p[i]=i;
   }
   int findSet(int i) {
     return (p[i]==i)? i : (p[i] = findSet(p[i]));
   }
   bool isSameSet(int i, int j){
     return findSet(i) == findSet(j);
   }
```

Exemplo UnionFind

```
void unionSet(int i, int j){
   if( !isSameSet(i,j)){
   numSets--;
   int x = findSet(i), y = findSet(j);
   if(rank[x]>rank[y]){ p[y]=x; setSize[x] += setSize[y]; }
   else{
    p[x]=y; setSize[y] += setSize[x];
    if(rank[x]==rank[y]) rank[y]++; }
   }
  }
  int numDisjointSets(){ return numSets; }
  int sizeOfSet(int i){ return setSize[findSet(i)];}
};
```

Exemplo UnionFind

```
int main(){
 UnionFind Set(10);
 Set.unionSet(5,6);
 Set.unionSet(6,7);
 cout << Set.numDisjointSets() << endl;
 Set.unionSet(0,1);
 cout << Set.numDisjointSets() << endl;
 Set.unionSet(5,8);
 cout << Set.numDisjointSets() << endl;
 cout << Set.sizeOfSet(5) << endl;
}
```

Table

i	1	2	3	4	5
f	1	1	2	2	3
	1	$1 \ldots 2$	$1 \ldots 3$	$1 \ldots 4$	$1 \ldots 5$
c	1	3	6	10	15

Table

```
#include <stdio.h>
#include <vector>

using namespace std;
class Table{
  private:
  vector <int> t;
  public:
  Table(int n){t.assign(n+1,0);}
  read(int b){ return t[b];}
  void update(int k, int v){
   for( ; k < (int)t.size(); k++) t[k] += v;
  }
  int range(int a, int b){return read(b)-read(a-1);}
};

int main(){
 int f[] = {2,4,5,5,6,6,6,7,7,8,9};
 Table t(10);
 for(int i = 0; i < 10; i++) t.update(i+1,f[i]);//11
 printf("fst(1,3)␣=␣%d\n", t.range(1,3) );
}
```

Sparse Table

```
#include <stdio.h>
#include <vector>
#include <math.h>

using namespace std;
class SparseTable{
  private:
  int n;
  vector <int> A;
  vector < vector <int> > lookup;
  void process();
  public:
  SparseTable(const vector <int> &_A);
  int query(int L, int R);
};
```

Sparse Table

```
SparseTable::SparseTable(const vector <int> &_A){
    A = _A;
    n = _A.size();
    lookup.resize( n );
    for(int i =0; i < n; i++)
      lookup[i].resize( (int)(ceil(log2(n))), 0 );
    process();
}

void SparseTable::process(){
 //inicialize lookup para intervalo com tamanho 1
 for(int i = 0; i < n; i++){
   lookup[i][0] = A[i];
 }
 //compute o valor de intervalos maiores
 //a partir de intervalos menores
 for(int j = 1; 1 << j <= n; j++){
  for(int i = 0; i+(1<<j)-1<n;i++){
    lookup[i][j] = lookup[i][j-1] + lookup[i+(1<<(j-1))][j-1];
  }
 }
}
```

Sparse Table

```
int SparseTable::query(int L, int R){
    if(L==R){
     return lookup[L][0];
    }else{
     int j = (int) log2(R-L+1);
     return lookup[L][j] + query(L+(1<<j),R);
    }
}

int main(){
 int f[] = {1,2,3,4,5,6,7};
 vector <int> A(f, f+7);
 SparseTable st(A);
 printf("query(%d,%d)␣=␣%d\n", 2,4, st.query(2,6) );
}
```

Binary Indexed Tree

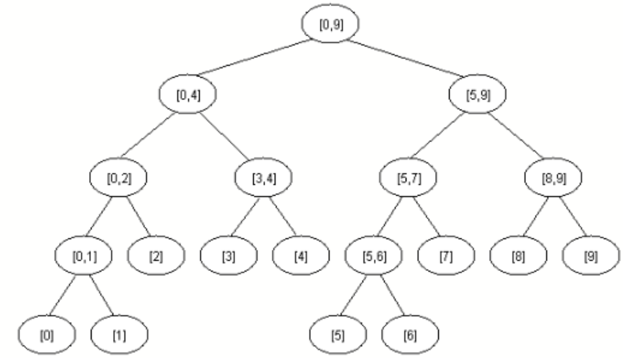
i	1	2	3	4	5	6	7	8	9
f	1	2	3	4	5	6	7	8	9
	1	$1 \ldots 2$	3	$1 \ldots 4$	5	$5 \ldots 6$	7	$1 \ldots 8$	9
st	1	3	3	10	5	11	7	36	9

- Para alterar o valor de $f[3]$ precisamos atualizar st[3],st[4],st[8].
- Precisamos somar (1 $<<$ posição do bit menos significativo).
  - $3 = (11)_2 + (01)_2 = (100)_2 = 4$
  - $4 = (100)_2 + (100)_2 = (1000)_2 = 8$
  - $8 = (1000)_2 + (1000)_2 = (10000)_2 = 16$
- Para calcular $f[1] + \ldots + f[6]$ precisamos somar st[6]+st[4]
- Precisamos decrementar (1 $<<$ posição do bit menos significativo).
  - $6 = (110)_2 - (10)_2 = (100)_2 = 4$
  - $4 = (100)_2$ - $(100)_2 = (000)_2 = 0$

Segment Tree

```
#include <stdio.h>
#include <vector>
using namespace std;
class FenwickTree{
 private:
 vector <int> ft;
 public:
 FenwickTree(int n){ ft.assign(n+1,0);}
 int read(int b){
  int sum = 0;
  for(; b; b -= (b & -b)) sum+= ft[b];
  return sum;
 }
 void update(int k ,int v){
  for( ; k < (int) ft.size(); k += (k & -k) ) ft[k] += v;
 }
 int range(int a, int b){
  if(a==0) return read(b);
  else return read(b) - read(a-1);
 }
};
```

Segment Tree



Segment Tree

```
class SegmentTree {
 private:
  vector <int> st,A;
  int n;
  int left(int p) { return p << 1;}
  int right(int p){ return (p << 1) + 1; }

  void build(int p, int L, int R);
  int rmq(int p, int L, int R, int i, int j);

  public:

  SegmentTree(const vector <int> &_A){
   A = _A;
   n = (int)A.size();
   //2*2^(floor((lg n))+1) = O(4n)
   st.assign(4*n,0);
   build(1, 0, n-1);
  }

  int rmq(int i, int j){ return rmq(1,0,n-1,i,j); }

  }
```

Segment Tree

```
void build(int p, int L, int R){
   if(L==R){
    st[p]=L;
   }else{
    build(left(p), L, (L+R)/2);
    build(right(p), (L+R)/2+1, R);
    int p1 = st[left(p)];
    int p2 = st[right(p)];
    st[p] = (A[p1]<=A[p2])?p1:p2;
   }
}
```

Segment Tree

```
int rmq(int p, int L, int R, int i, int j) {
  if (i > R || j < L) return -1;
  if (L >= i && R <= j) return st[p];
  int p1 = rmq(left(p), L , (L + R) / 2, i, j);
  int p2 = rmq(right(p), (L + R) / 2 + 1, R , i, j);

  if (p1 == -1) return p2;
  if (p2 == -1) return p1;
  return (A[p1] <= A[p2]) ? p1 : p2;
}
```

Segment Tree

```
void update(int p, int L, int R, int i)
{
  if( L == R) return ;
  if( i >= L && i <= R)
  {
    int mid = L+R/2;
    if( i <= mid ) update(left(p) , L, (L+R)/2, i);
    else update(right(p), (L+R)/2 + 1, R, i);
    int p1 = st[left(p)];
    int p2 = st[right(p)];
    st[p] = (A[p1]<=A[p2])?p1:p2;
  }
}
```

Segment Tree

```
class SegmentTree {
 private:
 //
 public:
  SegmentTree(const vector <int> &_A){
   A = _A; n = (int)A.size(); st.assign(4*n,0);
    build(1, 0, n-1);
  }
  int update(int k, int val){A[k] = val; update(1,0,n-1,k);};
  int rmq(int i, int j){ return A[rmq(1,0,n-1,i,j)]; }
};

int main(){
 int arr[] = {18,17,13,19,15,11,20};
 vector<int> A(arr,arr+7);
 SegmentTree st(A);
 printf(”RMQ(4,6)␣=␣%d\n”, st.rmq(4,6));
 printf(”RMQ(1,3)␣=␣%d\n”, st.rmq(1,3));
 st.update(2,10);
 printf(”RMQ(1,3)␣=␣%d\n”, st.rmq(1,3));
}
```

Grafo : Matriz Adjacência

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

class Graph{
 private:
  vector<vector<bool>> M;
 public:
  int N;
  Graph(int N): N(N){
   M.resize(N);
    for(int j=0;j<N;j++){
     M[j].resize(N,false);
    }
  }
  vector<bool>& operator [](int i) { return M[i]; }
  edge(int a, int b, bool directed = false){
   M[a][b]=1;
    if(!directed)M[b][a]=1;
  }
};
```

Grafo : Matriz Adjacência

```
void dfs(Graph & G, vector<int> &visited, int i){
 if(!visited[i]){
  cout << "visitando␣" << i << endl;
  visited[i] = true;
  for(int j = 0; j < G.N; j++){
   if(G[i][j]){
    dfs(G, visited, j);
   }
  }
 }
}

bool conexo(Graph & G){
  vector <int> visited;
  visited.assign(G.N, false);
  dfs(G, visited, 0);
  for(int i = 0; i < (int)visited.size(); i++)
   if(!visited[i]) return false;
  return true;
}
```

Grafo : Matriz Adjacência

```
int main(){
 Graph G(5);
 G.edge(0,1);
 G.edge(1,2);
 G.edge(3,4);
 cout << (conexo(G) ? "conexo" : "desconexo") << endl;
 G.edge(2,3);
 cout << (conexo(G) ? "conexo" : "desconexo") << endl;
}
```

Grafo : Matriz Lista de Adjacência

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

class Graph{
 public:
  int N;
  vector<vector<int> > adj;
  Graph(int N): N(N){
   adj.resize(N);
  }
  edge(int a, int b, bool directed = false){
   adj[a].push_back(b);
   if(!directed) adj[b].push_back(a);
  }
};
```

Grafo : Matriz Lista de Adjacência

```
void dfs(Graph & G, vector<int> &visited, int i){
 if(!visited[i]){
  cout << "visitando␣" << i << endl;
  visited[i] = true;
  for(int j = 0; j < G.adj[i].size(); j++){
   int u = G.adj[i][j];
   dfs(G, visited, u);
  }
 }
}

bool conexo(Graph & G){
  vector <int> visited;
  visited.assign(G.N, false);
  dfs(G, visited, 0);
  for(int i = 0; i < (int)visited.size(); i++)
   if(!visited[i]) return false;
  return true;
}
```