Instituto Superior Técnico — LMAC/MEBiom

V1

Elementos de Programação

Dezembro 2020 Mini-teste 4 Duração: 30m

Considere a camada de abstração que disponibiliza o tipo de dados *grafo não-dirigido com arestas coloridas*, onde se assume que os nós de cada grafo são numerados (`0,1,...,n-1` onde `n` é o número de nós do grafo), e também as possíveis cores são numeradas (`0,1,...,c-1` onde `c` é o número de cores considerado para as arestas de cada grafo), com as seguintes operações:

- `grvazio(n,c)`: grafo com `n` nós e sem qualquer aresta (mas com `c` cores para colorir potenciais arestas);
- `jaresta(g,i,j,k)`: grafo que resulta de juntar ao grafo `g` uma aresta de cor `k` a ligar os nós `i` e `j`;
- `nos(g)`: número de nós do grafo `g`;
- `cores(g)`: número de cores potenciais das arestas do grafo `g`;
- `arestaQ(g,i,j,k)`: `True` se existe no grafo `g` uma aresta de cor `k` a ligar os nós `i` e `j`, `False` caso contrário.

Por exemplo, na definição da ilustração abaixo, à esquerda, constrói-se o grafo `gr` que se mostra ao centro, assumindo que às três cores azul, vermelho, verde correspondem os valores 0, 1, 2, respectivamente.

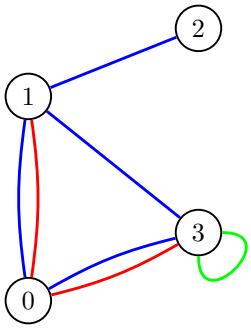
Defina em *Python*, sempre assegurando a independência relativamente à implementação do tipo de dados (disponibilizada na próxima página), as seguintes funções:

(**a**) `edgesfrom` que recebendo como argumento um grafo não-dirigido com arestas coloridas e um nó, calcula a lista de todos os pares `(j,k)` correspondentes a cada aresta de cor `k` que ligue o nó dado com o nó `j`; por exemplo, `edgesfrom(gr,3)` devolverá a lista `[(0,0),(0,1),(1,0),(3,2)]`;

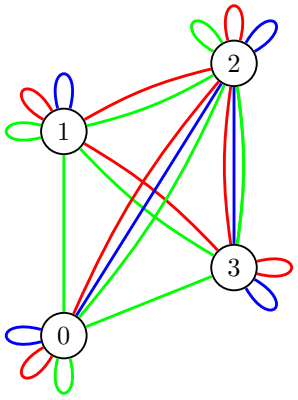
(**b**) `inverse` que recebendo como argumento um grafo não-dirigido com arestas coloridas calcula o grafo inverso, que tem os mesmos nós, mas que tem uma aresta de cor `k` a ligar os nós `i` e `j` se e só se tal aresta não existe no grafo dado; por exemplo, `inverse(gr)` corresponderá ao grafo abaixo, à direita;

(**c**) `coloronceQ` que recebendo como argumento um grafo não-dirigido com arestas coloridas e dois nós, devolve `True` se for possível no grafo dado ligar os nós dados usando arestas sem nunca repetir cores; por exemplo, `coloronceQ(gr,0,2)` deverá ser `True`, e `coloronceQ(gr,2,3)` deverá ser `False`.

```
gr=grvazio(4,3)
gr=jaresta(gr,0,1,0)
gr=jaresta(gr,0,1,1)
gr=jaresta(gr,1,2,0)
gr=jaresta(gr,1,3,0)
gr=jaresta(gr,3,3,2)
gr=jaresta(gr,3,0,0)
gr=jaresta(gr,3,0,1)
```



grafo `gr`



grafo `inverse(gr)`

```
def grvazio(n,c):
    return (n,[[] for k in range(c)])

def jaresta(g,i,j,k):
    (n,w)=g
    if i>j:
        i,j=j,i
    w[k]=w[k]+[(i,j)]
    return (n,w)

def nos(g):
    (n,w)=g
    return n

def cores(g):
    (n,w)=g
    return len(w)

def arestaQ(g,i,j,k):
    (n,w)=g
    if i>j:
        i,j=j,i
    return (o,d) in w[k]
```

Instituto Superior Técnico — LMAC/MEBiom **V2**

Elementos de Programação

Dezembro 2020 Mini-teste 4 Duração: 30m

Considere a camada de abstração que disponibiliza o tipo de dados *grafo não-dirigido com arestas coloridas*, onde se assume que os nós de cada grafo são numerados (`0,1,...,n-1` onde `n` é o número de nós do grafo), e também as possíveis cores são numeradas (`0,1,...,c-1` onde `c` é o número de cores considerado para as arestas de cada grafo), com as seguintes operações:

- `grvazio(n,c)`: grafo com `n` nós e sem qualquer aresta (mas com `c` cores para colorir potenciais arestas);
- `jaresta(g,i,j,k)`: grafo que resulta de juntar ao grafo `g` uma aresta de cor `k` a ligar os nós `i` e `j`;
- `nos(g)`: número de nós do grafo `g`;
- `cores(g)`: número de cores potenciais das arestas do grafo `g`;
- `posscores(g,i,j)`: lista de cores das arestas de `g` que ligam os nós `i` e `j`.

Por exemplo, na definição da ilustração abaixo, à esquerda, constrói-se o grafo `gr` que se mostra ao centro, assumindo que às três cores azul, vermelho, verde correspondem os valores 0, 1, 2, respectivamente.

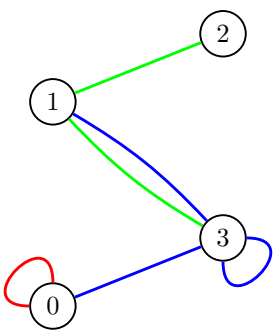
Defina em *Python*, sempre assegurando a independência relativamente à implementação do tipo de dados (disponibilizada na próxima página), as seguintes funções:

(**a**) `edgesfrom` que recebendo como argumento um grafo não-dirigido com arestas coloridas e um nó, calcula a lista de todos os pares `(j,k)` correspondentes a cada aresta de cor `k` que ligue o nó dado com o nó `j`; por exemplo, `edgesfrom(gr,1)` devolverá a lista `[(3,0),(3,2),(2,2)]`;

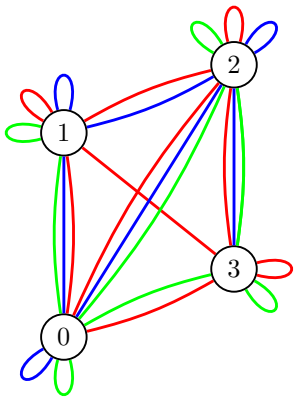
(**b**) `inverse` que recebendo como argumento um grafo não-dirigido com arestas coloridas calcula o grafo inverso, que tem os mesmos nós, mas que tem uma aresta de cor `k` a ligar nós `i` e `j` se e só se tal aresta não existe no grafo dado; por exemplo, `inverse(gr)` corresponderá ao grafo abaixo, à direita;

(**c**) `monoQ` que recebendo como argumento um grafo não-dirigido com arestas coloridas e dois nós, devolve `True` se for possível no grafo dado ligar os nós dados usando arestas de uma única cor, `False` caso contrário; por exemplo, `monoQ(gr,0,2)` deverá ser `False`, e `monoQ(gr,0,1)` deverá ser `True`.

```
gr=grvazio(4,3)
gr=jaresta(gr,0,0,1)
gr=jaresta(gr,0,3,0)
gr=jaresta(gr,3,3,0)
gr=jaresta(gr,3,1,0)
gr=jaresta(gr,3,1,2)
gr=jaresta(gr,1,2,2)
```



grafo `gr`



grafo `inverse(gr)`

```
def grvazio(n,c):
    return (c,[[[] for j in range(n)] for i in range(n)])

def jaresta(g,i,j,k):
    (c,m)=g
    m[i][j]=m[i][j]+[k]
    if i!=j:
       m[j][i]=m[j][i]+[k]
    return (c,m)

def nos(g):
    (c,m)=g
    return len(m)

def cores(g):
    (c,m)=g
    return c

def posscores(g,i,j):
    (c,m)=g
    return m[i][j]
```

Instituto Superior Técnico — LMAC/MEBiom

V3

Elementos de Programação

Dezembro 2020 Mini-teste 4 Duração: 30m

Considere a camada de abstração que disponibiliza o tipo de dados *grafo dirigido com arestas coloridas*, onde se assume que as possíveis cores das arestas de cada grafo são numeradas (`0,1,...,c-1` onde `c` é o número de cores considerado), com as operações:

- `grvazio(w,c)`: grafo cujos nós são os elementos da lista `w`, e sem qualquer aresta (mas com `c` cores para colorir potenciais arestas);
- `jaresta(g,o,d,k)`: grafo que resulta de juntar ao grafo `g` uma aresta de cor `k` do nó `o` para o nó `d`;
- `nos(g)`: lista de nós do grafo `g`;
- `cores(g)`: número de cores potenciais das arestas do grafo `g`;
- `posscores(g,o,d)`: lista de cores das arestas do nó `o` para o nó `d` no grafo `g`.

Por exemplo, na definição da ilustração abaixo, à esquerda, constrói-se o grafo `gr`, assumindo que às três cores azul, vermelho, verde correspondem os valores 0, 1, 2, respectivamente.

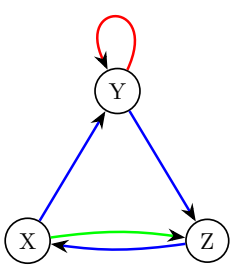
Defina em *Python*, sempre assegurando a independência relativamente à implementação do tipo de dados (disponibilizada na próxima página), as seguintes funções:

(**a**) `edgescolor` que recebendo como argumento um grafo dirigido com arestas coloridas e uma cor, calcula a lista de todos os pares de nós `(o,d)` para os quais existe uma aresta de `o` para `d` da cor dada; por exemplo, `edgescolor(gr,0)` devolverá a lista `[("X","Y"),("Y","Z"),("Z","X")]`;

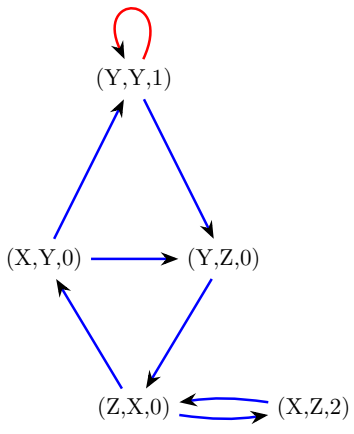
(**b**) `conjugate` que recebendo como argumento um grafo dirigido com arestas coloridas calcula o grafo conjugado, cujos nós são as arestas do grafo dado (no formato `(o,d,k)` onde `o` é a origem, `d` o destino, e `k` a sua cor), cujo número de cores é `n**2` onde `n` é o número de cores do grafo dado, e dadas duas arestas (agora nós) há uma aresta da primeira para a segunda se o destino da primeira é igual à origem da segunda, cuja cor é o produto das cores das duas; por exemplo, `conjugate(gr)` corresponderá ao grafo abaixo, à direita;

(**c**) `regularQ` que recebendo como argumento um grafo dirigido com arestas coloridas e uma cor, devolve `True` se o número de arestas da cor dada com origem em cada nó é igual ao número de arestas da cor dada com destino em cada nó, e é igual para todos os nós, `False` caso contrário; por exemplo, `regularQ(gr,0)` deverá ser `True`.

```
gr=grvazio(["X","Y","Z"],3)
gr=jaresta(gr,"X","Y",0)
gr=jaresta(gr,"Y","Z",0)
gr=jaresta(gr,"Y","Y",1)
gr=jaresta(gr,"Z","X",0)
gr=jaresta(gr,"X","Z",2)
```



grafo `gr`



grafo `conjugate(gr)`

```
def grvazio(w,c):
    return (w,c,[])

def jaresta(g,o,d,k):
    (w,c,arestas)=g
    arestas=arestas+[(o,d,k)]
    return (w,c,arestas)

def nos(g):
    (w,c,arestas)=g
    return w

def cores(g):
    (w,c,arestas)=g
    return c

def posscores(g,o,d):
    (w,c,arestas)=g
    poss=[]
    for (a,b,k) in arestas:
        if a==o and b==d:
            poss=poss+[k]
    return poss
```

Instituto Superior Técnico — LMAC/MEBiom **V4**

Elementos de Programação

Dezembro 2020 Mini-teste 4 Duração: 30m

Considere a camada de abstração que disponibiliza o tipo de dados *grafo dirigido com arestas coloridas*, onde se assume que os nós de cada grafo são numeradas (`0,1,...,n-1` onde `n` é o número de nós considerado), com as operações:

- `grvazio(n,w)`: grafo com `n` nós e sem qualquer aresta (mas com as cores da lista `w` para colorir potenciais arestas);
- `jaresta(g,o,d,k)`: grafo que resulta de juntar ao grafo `g` uma aresta de cor `k` do nó `o` para o nó `d`;
- `nos(g)`: número de nós do grafo `g`;
- `cores(g)`: lista de potenciais cores das arestas do grafo `g`;
- `arestaQ(g,o,d,k)`: `True` se existe no grafo `g` uma aresta de cor `k` do nó `o` para o nó `d`, `False` caso contrário.

Por exemplo, na definição da ilustração abaixo, à esquerda, constrói-se o grafo `gr` ao centro.

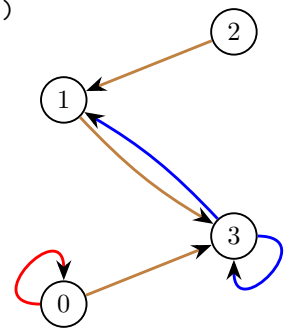
Defina em *Python*, sempre assegurando a independência relativamente à implementação do tipo de dados (disponibilizada na próxima página), as seguintes funções:

(**a**) `dest` que recebendo como argumento um grafo dirigido com arestas coloridas um nó e uma cor, calcula a lista de todos os nós destino de arestas do grafo que são da cor dada e com origem no nó dado; por exemplo, `dest(gr,3,"blue")` devolverá a lista `[1,3]`;

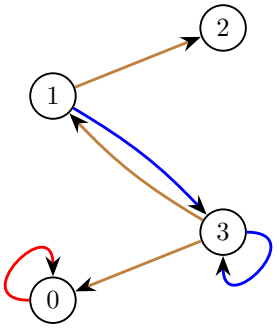
(**b**) `converse` que recebendo como argumento um grafo dirigido com arestas coloridas calcula o grafo em que todas as arestas têm a sua direcção invertida; por exemplo, `converse(gr)` corresponderá ao grafo abaixo, à direita;

(**c**) `accessQ` que recebendo como argumento um grafo dirigido com arestas coloridas, dois nós e uma cor, devolve `True` se é possível aceder a partir do primeiro nó até ao segundo usando apenas arestas da cor dada, `False` caso contrário; por exemplo, `accessQ(gr,2,3,"brown")` deverá ser `True`, e `accessQ(gr,0,1,"blue")` deverá ser `False`.

```
gr=grvazio(4,["blue","red","brown"])
gr=jaresta(gr,0,0,"red")
gr=jaresta(gr,0,3,"brown")
gr=jaresta(gr,3,3,"blue")
gr=jaresta(gr,3,1,"blue")
gr=jaresta(gr,1,3,"brown")
gr=jaresta(gr,2,1,"brown")
```



grafo `gr`



grafo `converse(gr)`

```
def grvazio(n,w):
    return (w,[[[] for j in range(n)] for i in range(n)])

def jaresta(g,o,d,k):
    (wcores,m)=g
    m[o][d]=m[o][d]+[k]
    return (wcores,m)

def nos(g):
    (wcores,m)=g
    return len(m)

def cores(g):
    (wcores,m)=g
    return wcores

def arestaQ(g,o,d,k):
    (wcores,m)=g
    return (k in m[o][d])
```

Instituto Superior Técnico — LMAC/MEBiom

V5

Elementos de Programação

Dezembro 2020 Mini-teste 4 Duração: 30m

Considere a camada de abstração que disponibiliza o tipo de dados *grafo dirigido com arestas coloridas*, com as seguintes operações:

- `grvazio(wnos,wcores)`: grafo cujos nós são os elementos da lista `wnos`, e sem qualquer aresta (mas com as cores da lista `wcores` para colorir potenciais arestas);
- `jaresta(g,o,d,k)`: grafo que resulta de juntar ao grafo `g` uma aresta de cor `k` do nó `o` para o nó `d`;
- `nos(g)`: lista de nós do grafo `g`;
- `cores(g)`: lista de potenciais cores das arestas do grafo `g`;
- `posscores(g,o,d)`: lista de cores das arestas de `g` do nó `o` para o nó `d`.

Por exemplo, nas definições da ilustração abaixo, à esquerda constrói-se o grafo `g1`, e ao centro o grafo `g2`.

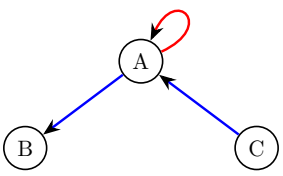
Defina em *Python*, sempre assegurando a independência relativamente à implementação do tipo de dados (disponibilizada na próxima página), as seguintes funções:

(**a**) `edges` que recebendo como argumento um grafo dirigido com arestas coloridas, calcula a lista de todos as arestas do grafo, cada aresta na forma `(o,d,k)` onde `o` é o nó origem da aresta, `d` é o nó destino, e `k` a cor da aresta; por exemplo, `edges(g1)` devolverá a lista `[("A","A","red"),("A","B","blue"),("C","A","blue")]`;

(**b**) `cartesian` que recebendo como argumento dois grafos dirigidos com arestas coloridas, calcula o grafo cujos nós são os pares `(o1,o2)` onde `o1` é um nó do primeiro grafo dado e `o2` é um nó do segundo, cujas cores são todas as cores dos dois grafos, e que tem uma aresta de cor `k` de `(o1,o2)` para `(d1,d2)` se `o1==d1` e `(o2,d2,k)` é uma aresta do segundo grafo, ou se `o2==d2` e `(o1,d1,k)` é uma aresta do primeiro grafo; por exemplo, `cartesian(g1,g2)` corresponderá ao grafo abaixo, à direita;

(**c**) `accessQ` que recebendo como argumento um grafo dirigido com arestas coloridas, dois nós e uma lista de cores, devolve `True` se é possível aceder a partir do primeiro nó até ao segundo usando apenas arestas cuja cor está na lista dada, `False` caso contrário; por exemplo, `accessQ(g1,"C","B",["blue"])` deverá ser `True`.

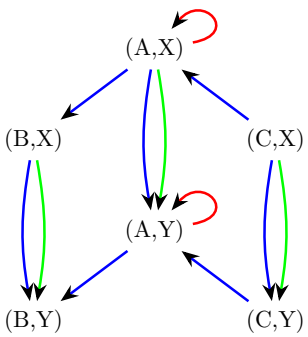
```
g1=grvazio(["A","B","C"],["blue","red"])
g1=jaresta(g1,"A","A","red")
g1=jaresta(g1,"A","B","blue")
g1=jaresta(g1,"C","A","blue")
```



grafo `g1`


```
g2=grvazio(["X","Y"],["blue","green"])
g2=jaresta(g2,"X","Y","green")
g2=jaresta(g2,"X","Y","blue")
```

grafo `g2`



grafo `cartesian(g1,g2)`

```
def grvazio(wnos,wcores):
    return (wnos,wcores,[[[] for j in range(len(wcores))] for i in range(len(wnos))])

def jaresta(g,o,d,k):
    (wnos,wcores,m)=g
    i=wnos.index(o)
    j=wcores.index(k)
    m[i][j]=m[i][j]+[d]
    return (wnos,wcores,m)

def nos(g):
    (wnos,wcores,m)=g
    return wnos

def cores(g):
    (wnos,wcores,m)=g
    return wcores

def posscores(g,o,d):
    (wnos,wcores,m)=g
    i=wnos.index(o)
    poss=[]
    for j in range(len(wcores)):
        if d in m[i][j]:
            poss=poss+[wcores[j]]
    return poss
```

Instituto Superior Técnico — LMAC/MEBiom

V6

Elementos de Programação

Dezembro 2020 Mini-teste 4 Duração: 30m

Considere a camada de abstração que disponibiliza o tipo de dados *grafo não-dirigido com arestas coloridas*, com as seguintes operações:

- `grvazio(wnos,wcores)`: grafo cujos nós são os elementos da lista `wnos`, e sem qualquer aresta (mas com as cores da lista `wcores` para colorir potenciais arestas);
- `jaresta(g,i,j,k)`: grafo que resulta de juntar ao grafo `g` uma aresta de cor `k` entre os nós `i` e `j`;
- `nos(g)`: lista de nós do grafo `g`;
- `cores(g)`: lista de potenciais cores das arestas do grafo `g`;
- `arestaQ(g,i,j,k)`: `True` se existe no grafo `g` uma aresta de cor `k` entre os nó `i` e `j`, `False` caso contrário.

Por exemplo, nas definições da ilustração abaixo, à esquerda constrói-se o grafo `gr`.

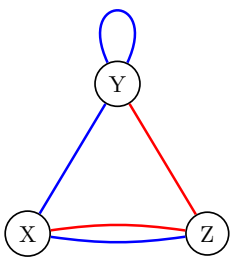
Defina em *Python*, sempre assegurando a independência relativamente à implementação do tipo de dados (disponibilizada na próxima página), as seguintes funções:

(**a**) `edgeswith` que recebendo como argumento um grafo não-dirigido com arestas coloridas e um nó `i`, calcula a lista de todos os pares `(j,k)` correspodentes a cada aresta de cada cor `k` que liga o nó dado com o nó `j`; por exemplo, `edgeswith(gr,"Y")` devolverá a lista `[("X","blue"),("Y","blue"),("Z","red")]`;

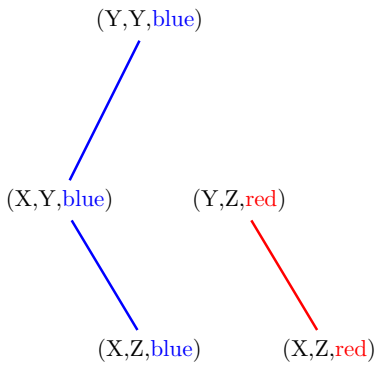
(**b**) `conjpercolor` que recebendo como argumento um grafo não-dirigido com arestas coloridas calcula o grafo conjugado por cores, cujos nós são as arestas do grafo dado (no formato `(o,d,k)` onde `o` é a origem, `d` o destino, e `k` a sua cor, onde `d` não ocorre antes de `o` na lista de nós do grafo), e dadas duas arestas distintas (agora nós) há uma aresta entre elas se partilham algum nó e têm a mesma cor, que será também a cor da aresta; por exemplo, `conjpercolor(gr)` corresponderá ao grafo abaixo, à direita.

(**c**) `monoregularQ` que recebendo como argumento um grafo não-dirigido com arestas coloridas, devolve `True` se o número de nós a que cada nó está ligado por alguma aresta é igual para todos os nós, `False` caso contrário; por exemplo, `monoregularQ(gr)` deverá ser `False` (o nó `Y` está ligado a todos os três nós, mas os nós `X,Z` só estão ligados a dois nós cada um).

```
gr=grvazio(["X","Y","Z"],["blue","red"])
gr=jaresta(gr,"X","Y","blue")
gr=jaresta(gr,"Y","Z","red")
gr=jaresta(gr,"Y","Y","blue")
gr=jaresta(gr,"Z","X","blue")
gr=jaresta(gr,"X","Z","red")
```



grafo `gr`



grafo `conjpercolor(gr)`

```
def grvazio(wnos,wcores):
    return (wnos,wcores,[])

def jaresta(g,i,j,k):
    (wnos,wcores,w)=g
    w=w+[(i,j,k),(j,i,k)]
    return (wnos,wcores,w)

def nos(g):
    (wnos,wcores,w)=g
    return wnos

def cores(g):
    (wnos,wcores,w)=g
    return wcores

def arestaQ(g,i,j,k):
    (wnos,wcores,w)=g
    return ((i,j,k) in w)
```

Elementos de Programação

Dezembro 2020 Mini-teste 4 Duração: 30m

Considere, em *Python*, a camada de abstração que disponibiliza o tipo de dados *grafo não-dirigido com nós coloridos*, com as seguintes operações:

- `grvazio(paresnoscor)`: grafo sem arestas, cujos nós e respectivas cores correspondem aos pares `(no,cor)` da lista `paresnoscor`;
- `jaresta(g,i,j)`: grafo que resulta de juntar ao grafo `g` uma aresta a ligar os nós `i` e `j`;
- `nos(g)`: lista de nós do grafo `g`;
- `cores(g)`: lista de cores dos nós do grafo `g`;
- `arestaQ(g,i,j)`: `True` se existe no grafo `g` uma aresta entre os nós `i` e `j`, `False` caso contrário;
- `nosdecor(g,c)`: lista dos nós do grafo `g` cuja cor é `c`.

Por exemplo, na ilustração abaixo, à esquerda define-se o grafo `g1`, e ao centro o grafo `g2`.

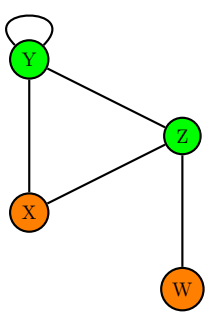
Defina em *Python*, sempre assegurando a independência relativamente à implementação do tipo de dados (disponibilizada na próxima página), as seguintes funções:

(**a**) `edges` que recebendo como argumento um grafo não-dirigido com nós coloridos, calcula a lista de todas as arestas do grafo, cada aresta na forma `(i,j)` onde `i` e `j` são os nós ligados pela aresta (onde `j` não ocorre antes de `i` na lista de nós do grafo); por exemplo, `edges(g1)` devolverá a lista `[("X","Y"),("X","Z"),("Y","Y"),("Y","Z"),("Z","W")]`;

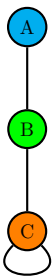
(**b**) `multipartQ` que recebendo como argumento um grafo não-dirigido com nós coloridos, devolve `True` se todas as arestas do grafo ligam nós de cores diferentes, `False` caso contrário; por exemplo, `multipartQ(g2)` deverá ser `False` (seria `True` se `g2` não tivesse uma aresta do nó `C` para si próprio);

(**c**) `colorcart` que recebendo como argumento dois grafos não-dirigidos com nós coloridos, calcula o grafo cujos nós são pares `(i1,i2)` onde `i1` é um nó do primeiro grafo dado e `i2` é um nó do segundo, ambos da mesma cor, que é herdada pelo par, e que tem uma aresta entre `(i1,i2)` e `(j1,j2)` sempre que `i1==j1` e `i2,j2` sejam ligados por uma aresta do segundo grafo, ou que `i2==j2` e `i1,j1` sejam ligados por uma aresta do primeiro grafo; por exemplo, `colorcart(g1,g2)` corresponderá ao grafo abaixo, à direita.

```
g1=grvazio([("X","orange"),("Y","green"),("Z","green"),("W","orange")])
g1=jaresta(g1,"X","Y")
g1=jaresta(g1,"Y","Y")
g1=jaresta(g1,"X","Z")
g1=jaresta(g1,"Y","Z")
g1=jaresta(g1,"W","Z")
```

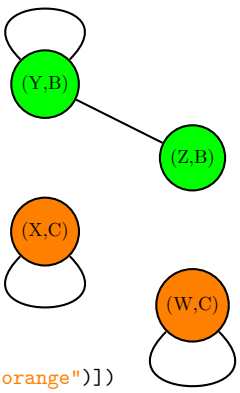


grafo `g1`



grafo `g2`

```
g2=grvazio([("A","blue"),("B","green"),("C","orange")])
g2=jaresta(g2,"A","B")
g2=jaresta(g2,"C","B")
g2=jaresta(g2,"C","C")
```



grafo `colorcart(g1,g2)`

```
def grvazio(paresnoscor):
    wnos=[a for (a,c) in paresnoscor]
    wcores=list({c for (a,c) in paresnoscor})
    m=[[0 for j in range(len(wnos))] for i in range(len(wnos))]
    return (wnos,[(c,[o for (o,k) in paresnoscor if k==c]) for c in wcores],m)

def jaresta(g,i,j):
    (wnos,nosporcor,m)=g
    ii=wnos.index(i)
    jj=wnos.index(j)
    m[ii][jj]=1
    m[jj][ii]=1
    return (wnos,nosporcor,m)

def nos(g):
    (wnos,nosporcor,m)=g
    return wnos

def cores(g):
    (wnos,nosporcor,m)=g
    wcores=[c for (c,w) in nosporcor]
    return wcores

def arestaQ(g,i,j):
    (wnos,nosporcor,m)=g
    ii=wnos.index(i)
    jj=wnos.index(j)
    return (m[ii][jj]==1)

def nosdecor(g,c):
    (wnos,nosporcor,m)=g
    res=[]
    found=False
    i=0
    while not(found) and i<len(nosporcor):
        (k,w)=nosporcor[i]
        if k==c:
            found=True
            res=w
        else:
            i=i+1
    return res
```

Instituto Superior Técnico — LMAC/MEBiom **V8**

Elementos de Programação

Dezembro 2020 Mini-teste 4 Duração: 30m

Considere a camada de abstração que disponibiliza o tipo de dados *grafo dirigido com nós coloridos*, com as seguintes operações:

- `grvazio(paresnoscor)`: grafo sem arestas, cujos nós e respectivas cores correspondem aos pares `(no,cor)` da lista `paresnoscor`;
- `jaresta(g,o,d)`: grafo que resulta de juntar ao grafo `g` uma aresta do nó `o` para o nó `d`;
- `nos(g)`: lista de nós do grafo `g`;
- `destinos(g,o)`: lista dos nós destino das arestas do grafo `g` que têm origem no nó `o`;
- `cordeno(g,o)`: cor do nó `o` do grafo `g`.

Por exemplo, na ilustração abaixo, à esquerda define-se o grafo `g1`, e ao centro o grafo `g2`.

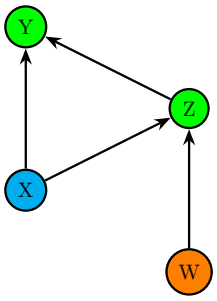
Defina em *Python*, sempre assegurando a independência relativamente à implementação do tipo de dados (disponibilizada na próxima página), as seguintes funções:

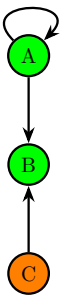
(**a**) `edges` que recebendo como argumento um grafo dirigido com nós coloridos, calcula a lista de todas as arestas do grafo, cada aresta na forma `(o,d)` onde `o` é o nó origem e `d` o nó destino; por exemplo, `edges(g2)` devolverá a lista `[("A","A"),("A","B"),("C","B")]`;

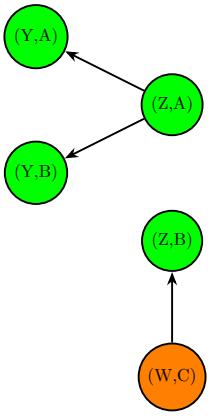
(**b**) `coloringQ` que recebendo como argumento um grafo dirigido com nós coloridos, devolve `True` se cada aresta do grafo tem origem e destino em nós de cores diferentes, `False` caso contrário; por exemplo, `coloringQ(g1)` deverá ser `False` (devido à aresta de `Z` para `Y`);

(**c**) `coltensor` que recebendo como argumento dois grafos dirigidos com nós coloridos, calcula o grafo cujos nós são pares `(o1,o2)` onde `o1` é um nó do primeiro grafo dado e `o2` é um nó do segundo, ambos da mesma cor, que é herdada pelo par, e que tem uma aresta de `(o1,o2)` para `(d1,d2)` sempre que haja uma aresta de `o1` para `d1` no primeiro grafo e uma aresta de `o2` para `d2` no segundo; por exemplo, `coltensor(g1,g2)` corresponderá ao grafo abaixo, à direita.

```
g1=grvazio([("X","blue"),("Y","green"),("Z","green"),("W","orange")])
g1=jaresta(g1,"X","Y")
g1=jaresta(g1,"X","Z")
g1=jaresta(g1,"Z","Y")
g1=jaresta(g1,"W","Z")
```








```
g2=grvazio([("A","green"),("B","green"),("C","orange")])
g2=jaresta(g2,"A","A")
g2=jaresta(g2,"A","B")
g2=jaresta(g2,"C","B")
```

grafo `g1` grafo `g2` grafo `coltensor(g1,g2)`

```
def grvazio(paresnoscor):
    return (paresnoscor,[])

def jaresta(g,o,d):
    (paresnoscor,ar)=g
    ar=ar+[(o,d)]
    return (paresnoscor,ar)

def nos(g):
    (paresnoscor,ar)=g
    wnos=[]
    for (o,c) in paresnoscor:
        wnos=wnos+[o]
    return wnos

def destinos(g,o):
    (paresnoscor,ar)=g
    return [y for (x,y) in ar if x==o]

def cordeno(g,o):
    (paresnoscor,ar)=g
    found=False
    i=0
    while not(found) and i<len(paresnoscor):
        (a,c)=paresnoscor[i]
        if o==a:
            found=True
            res=c
        else:
            i=i+1
    return res
```