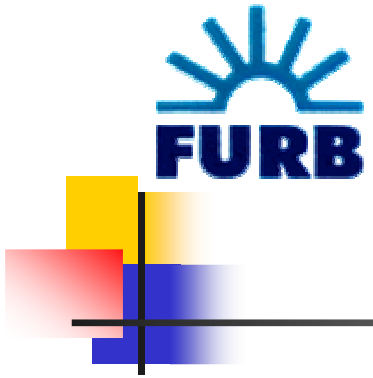
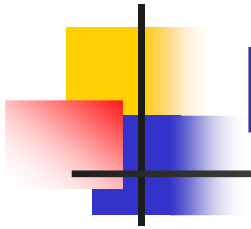


Universidade Regional de Blumenau
Centro de Ciências Exatas e Naturais

Desenvolvimento de um protótipo para o auxílio de distribuição de cargas

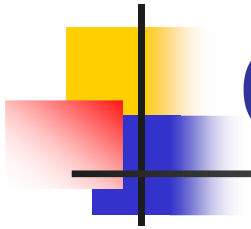
Aluno: Evandro Sestrem

Orientador: Maurício Capobianco Lopes



Roteiro

- Grafos
- Algoritmo de Dijkstra
- Listas de Fibonacci
- Protótipo



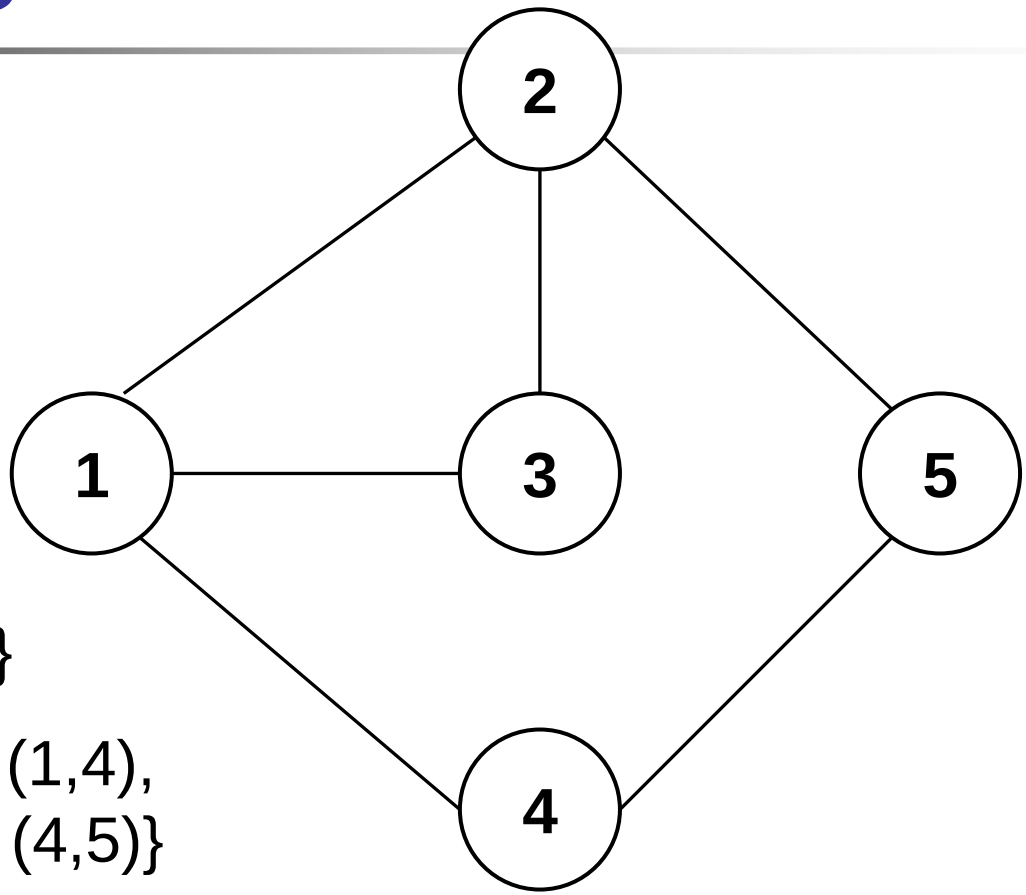
Objetivo

Implementar um protótipo que auxilie a distribuição de cargas

Objetivos específicos

- Construção de um site onde o transportador possa cadastrar suas cargas e possibilidades de entregas;
- O uso da teoria dos grafos para determinar o melhor caminho para as entregas.

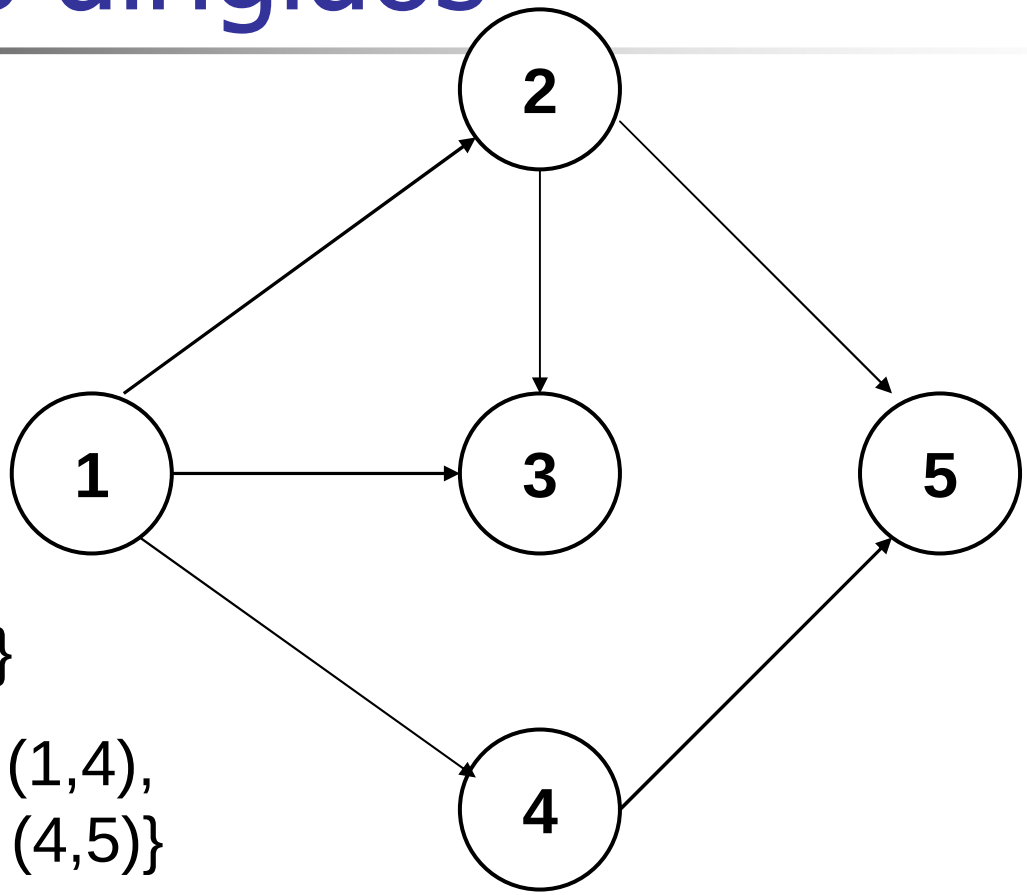
Grafos



V = {1, 2, 3, 4, 5}

E = {(1,2), (1,3), (1,4),
(2,3), (2,5), (4,5)}

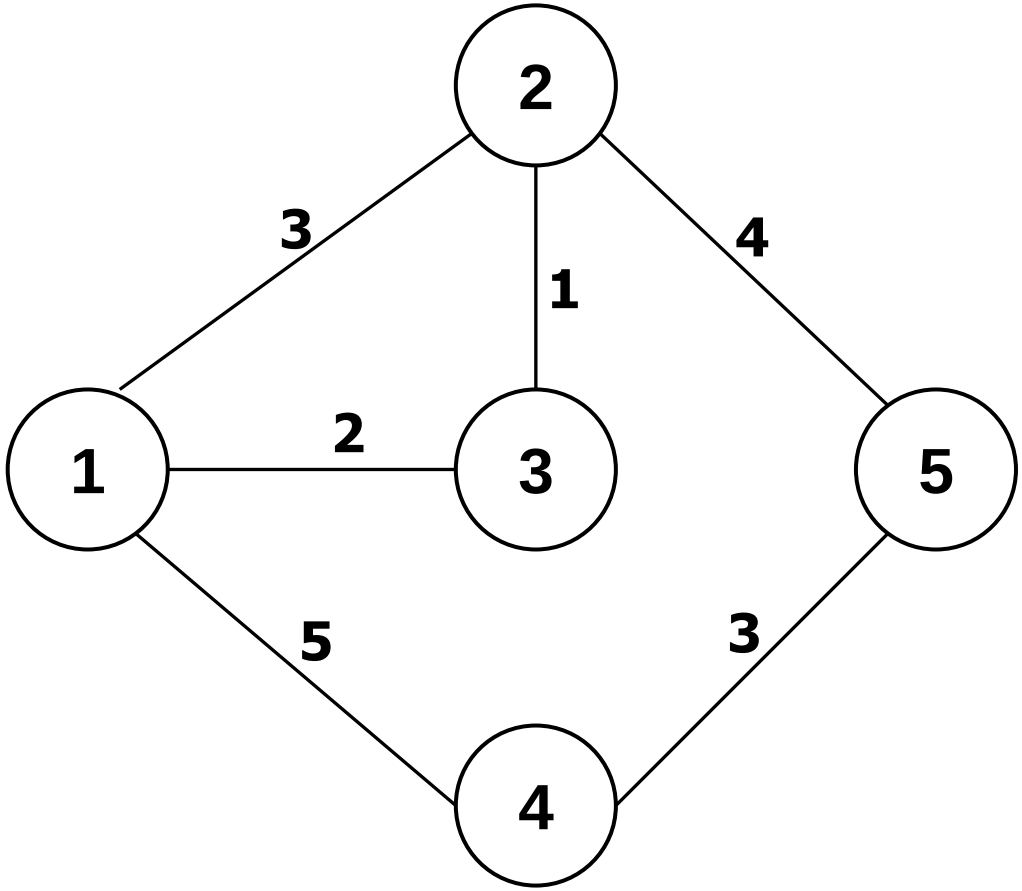
Grafos dirigidos

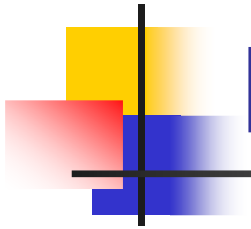


V = {1, 2, 3, 4, 5}

E = {(1,2), (1,3), (1,4),
(2,3), (2,5), (4,5)}

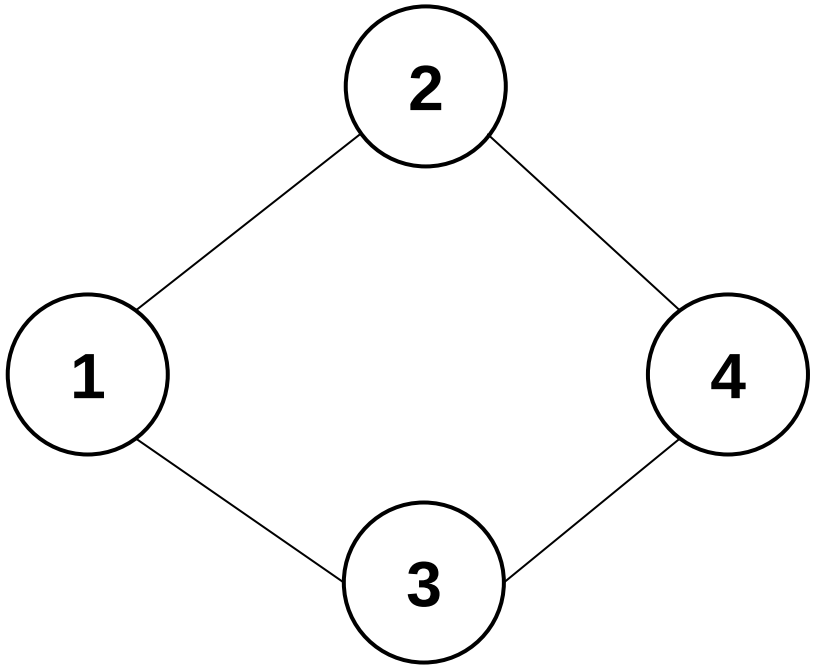
Grafos valorados





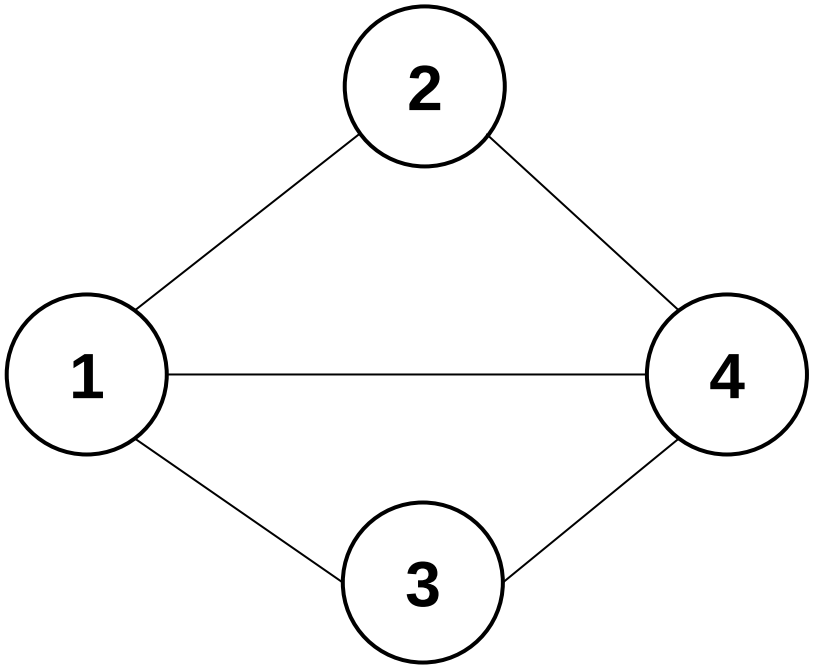
Representação de Grafos

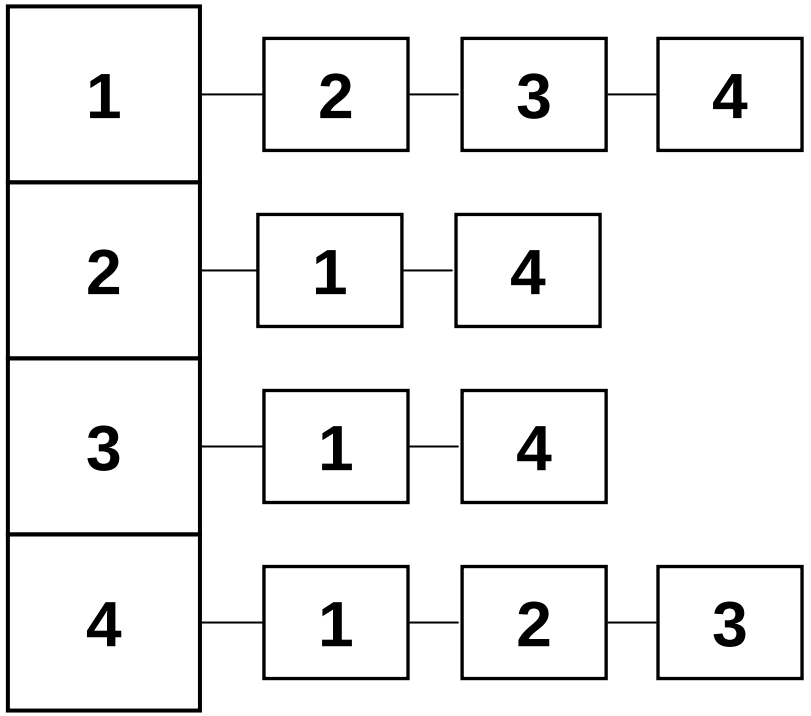
Matriz de Adjacências

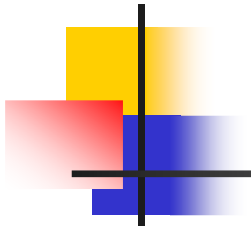


	1	2	3	4
1	0	1	1	0
2	1	0	1	1
3	1	1	0	1
4	0	1	1	0

Estrutura de Adjacências

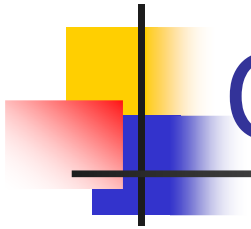




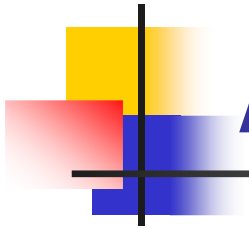


Busca em Grafos

- Busca em Profundidade
- Busca em Largura



Problema do Caminho Mais Curto



Algoritmo de Dijkstra

Dijkstra (G,w,s)

Initialize-Single-Source(G,s)

S ← { }

Q ← V[G]

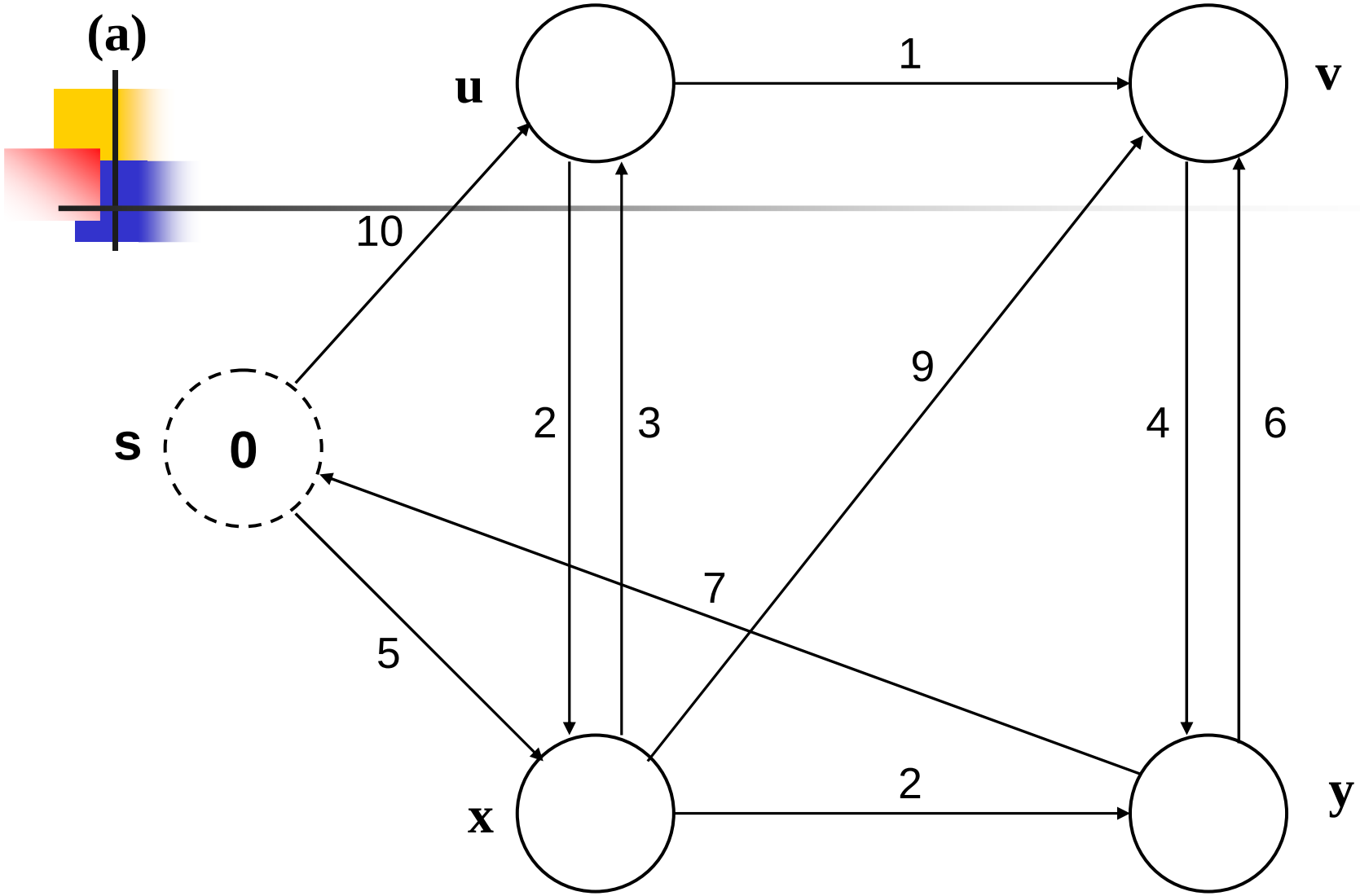
while Q <> { } do

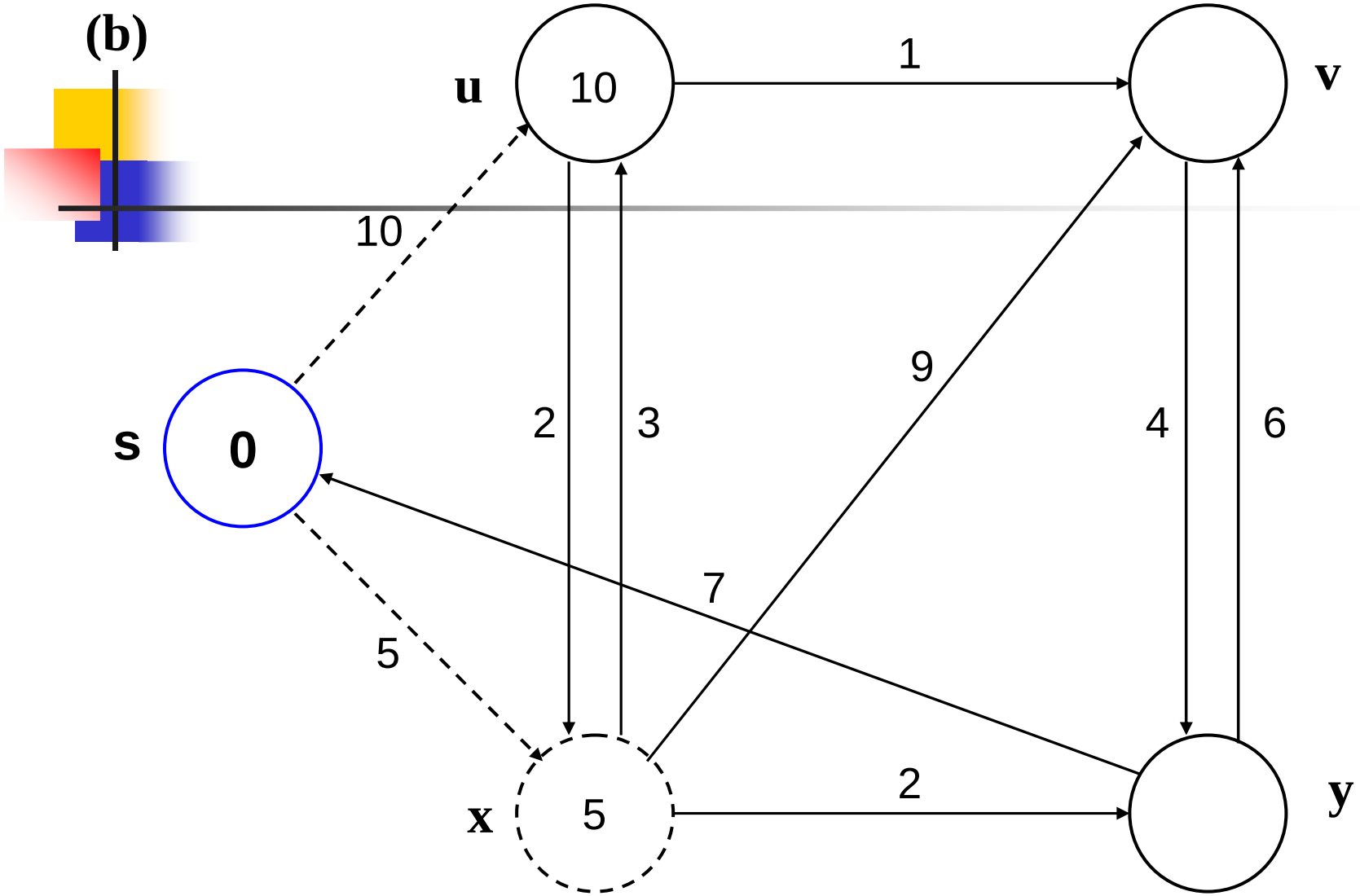
    u ← ExtractMin(Q)

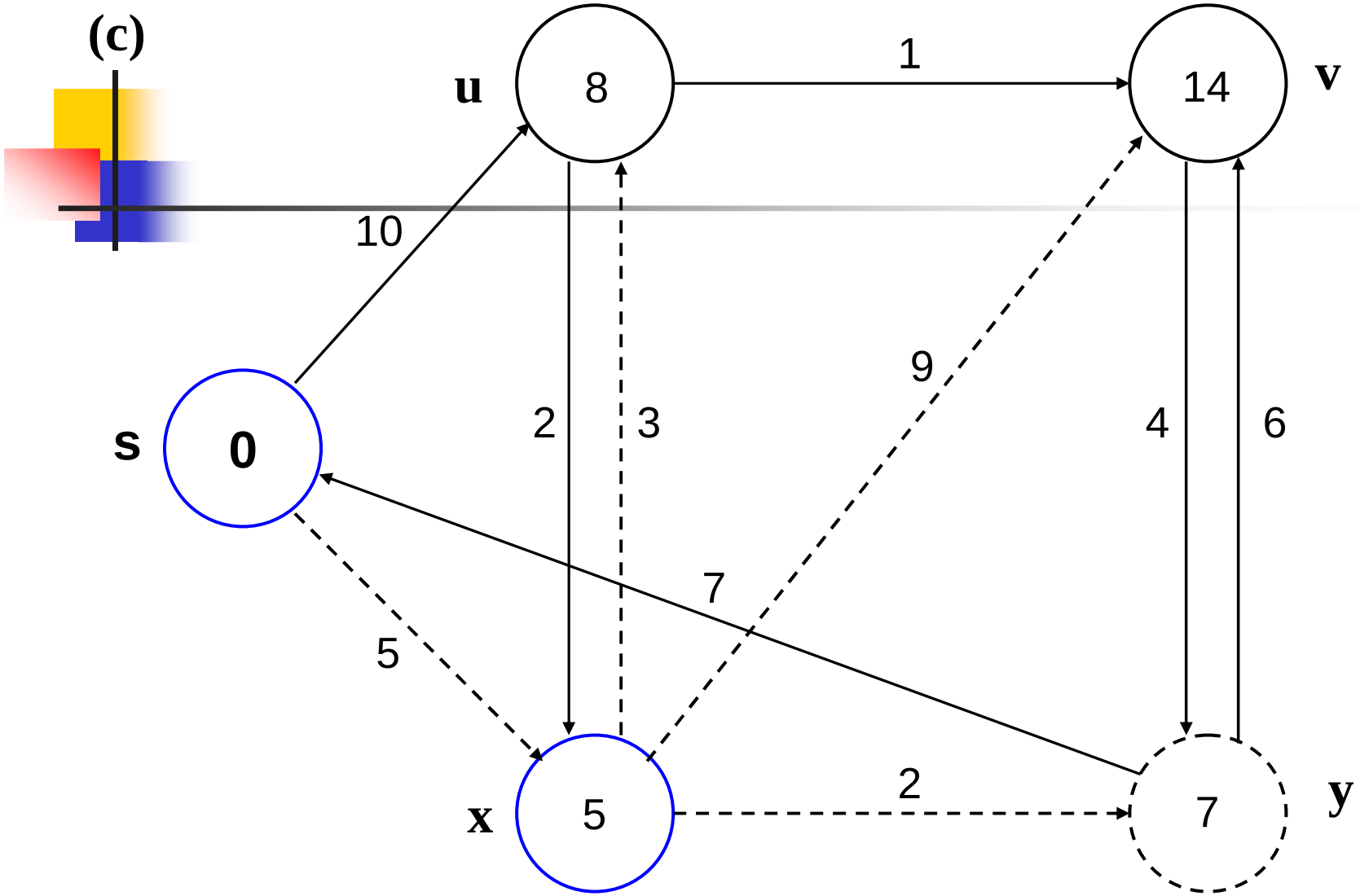
    S ← S U {u}

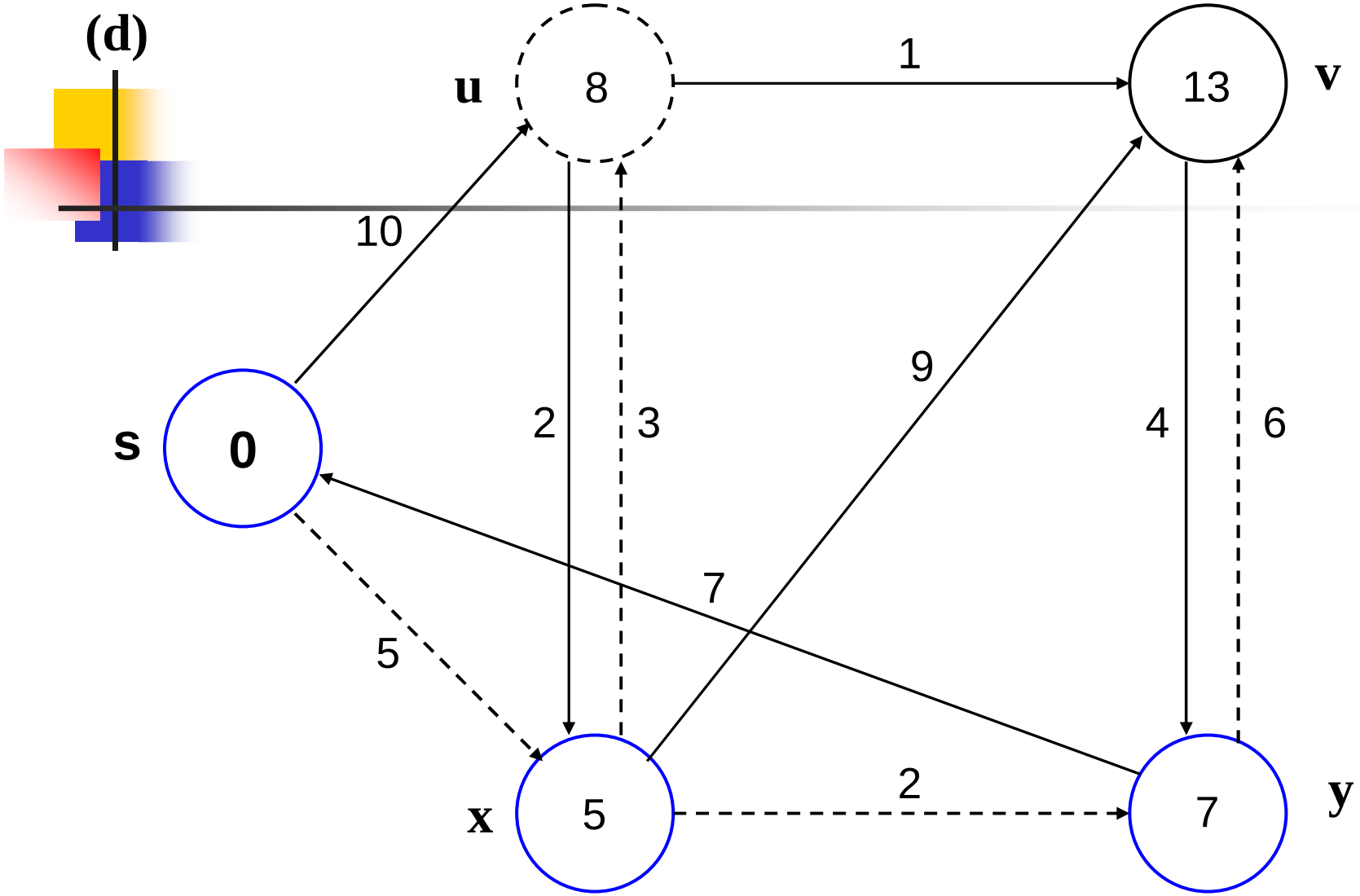
    for each vertex v ∈ Adj[u] do

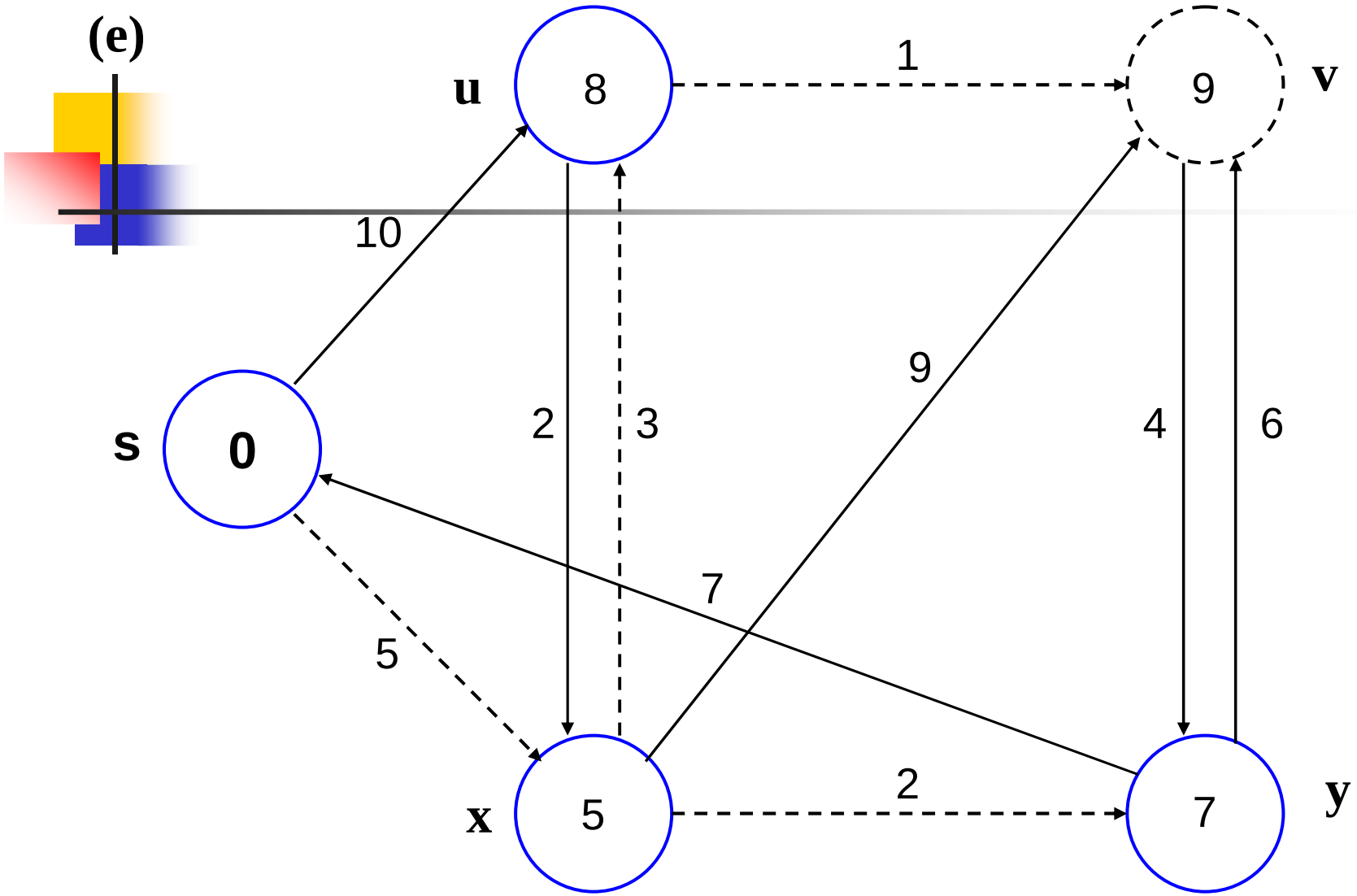
        Relax(u,v,w)

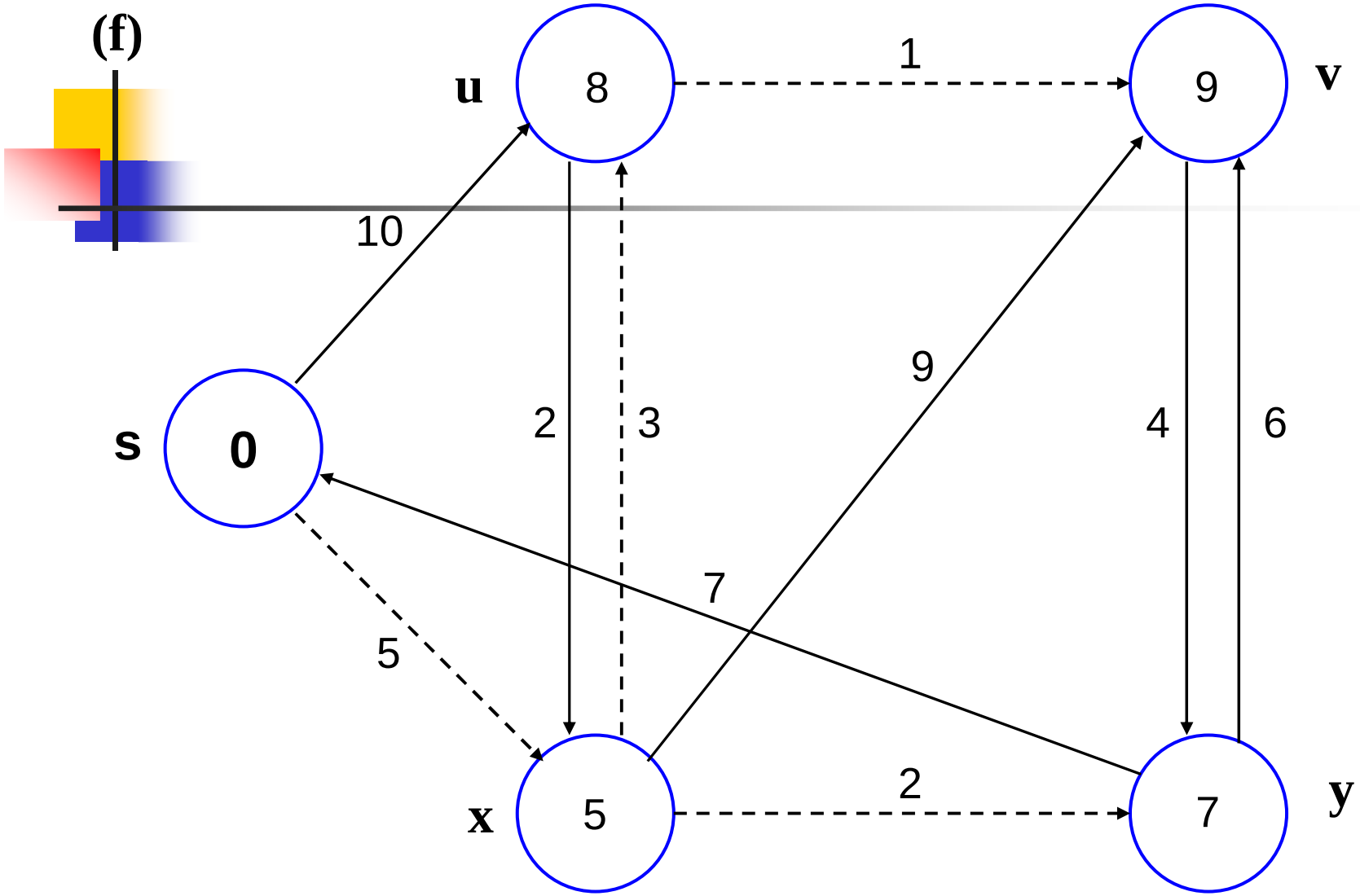

(a)
u
v
1
10
s
0
2
3
9
4
6
7
5
x
2
y

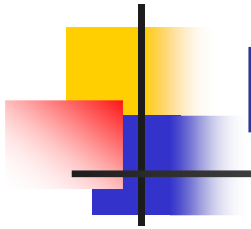

(b)
u
10
1
v
10
9
s
0
2
3
4
6
7
5
x
5
2
y


(c)
u
8
1
14
v
10
s
0
2
3
9
4
6
7
5
x
5
2
7
y


(d)
u
8
1
13
v
10
9
s
0
2
3
4
6
7
5
x
5
2
7
y


(e)
u
8
1
9
v
10
s
0
2
3
9
4
6
7
5
x
5
2
7
y


(f)
u
8
1
9
v
10
s
0
2
3
9
4
6
7
5
x
5
2
7
y

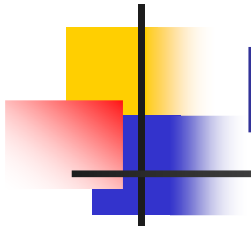


Performance

Algoritmo de Dijkstra com Lista de Fibonacci

Dijkstra (G,w,s)

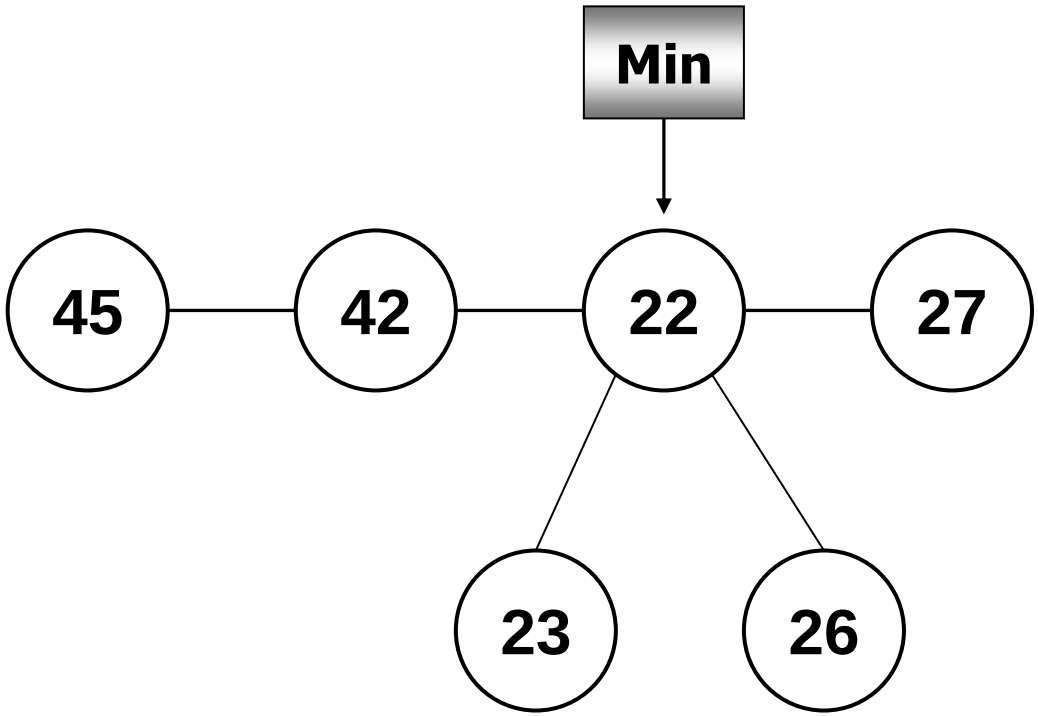
```
For all vertices v
        if v = S then v.cost := 0
        else v.cost := cInfinity
        Inserir(v, H)
while H <> { } do
        M ← ExtrairMinimo (H)
        for each vertex A attached to M
        w ← cost of edge from M to A
    if (M.cost + w < A.cost) then
                DiminuirChave(A,M.Cost + w)
                A.BackLink := M
```



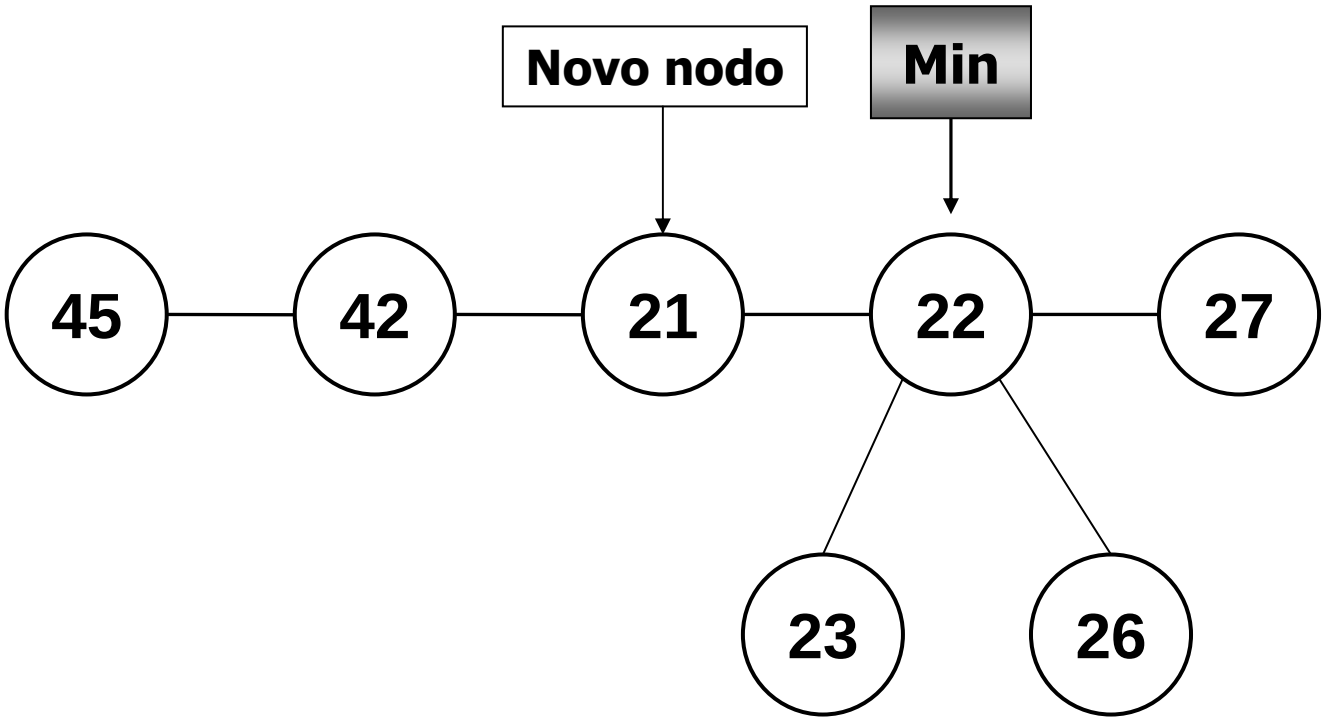
Lista de Fibonacci

- Inserir
- ExtrairMínimo
- DiminuirChave
- União
- Mínimo
- Deletar

Estrutura de uma lista de Fibonacci

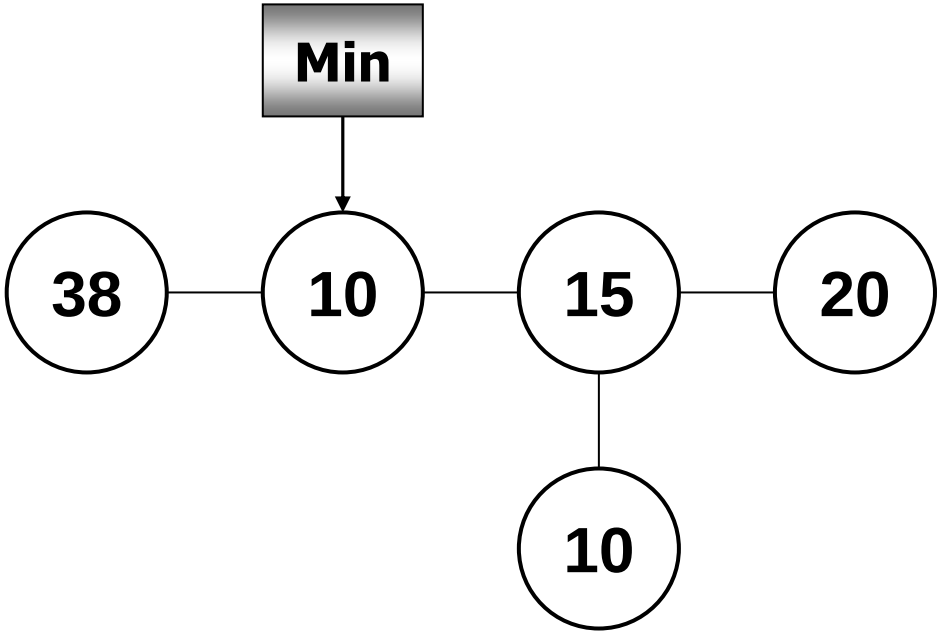


Inserir



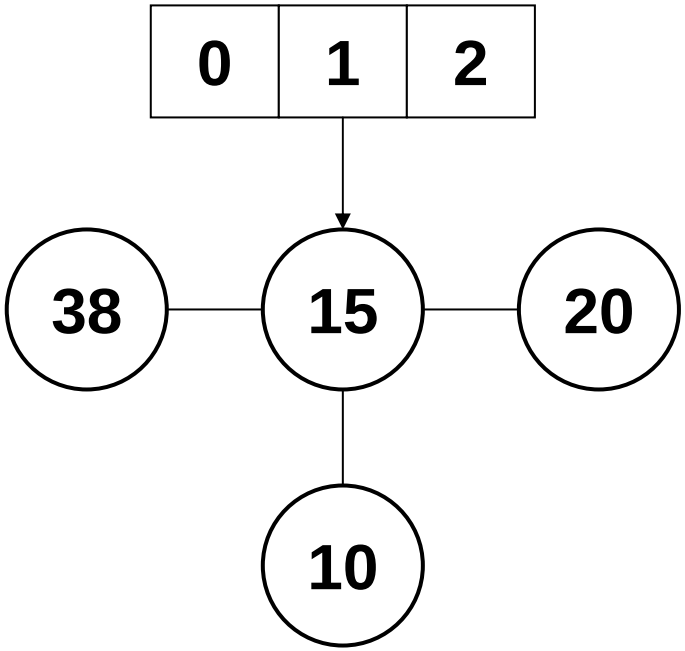
ExtrairMínimo

(a)



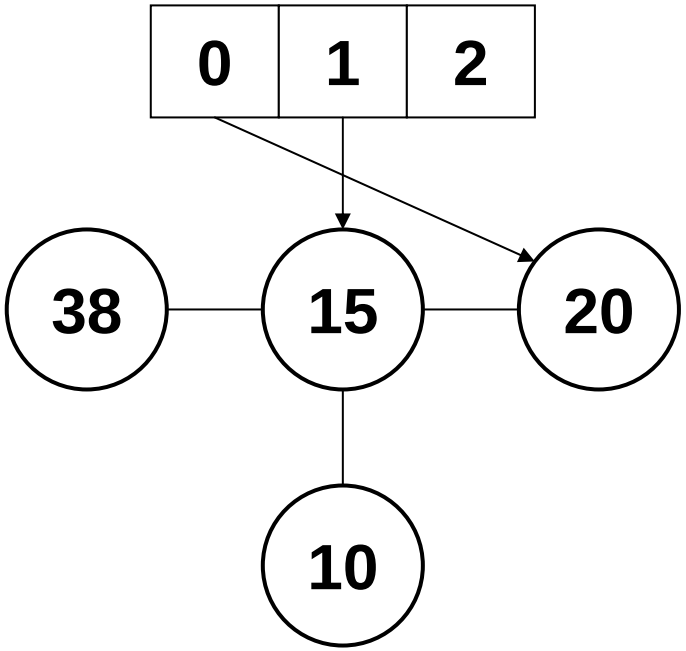
ExtrairMínimo (cont.)

(b)

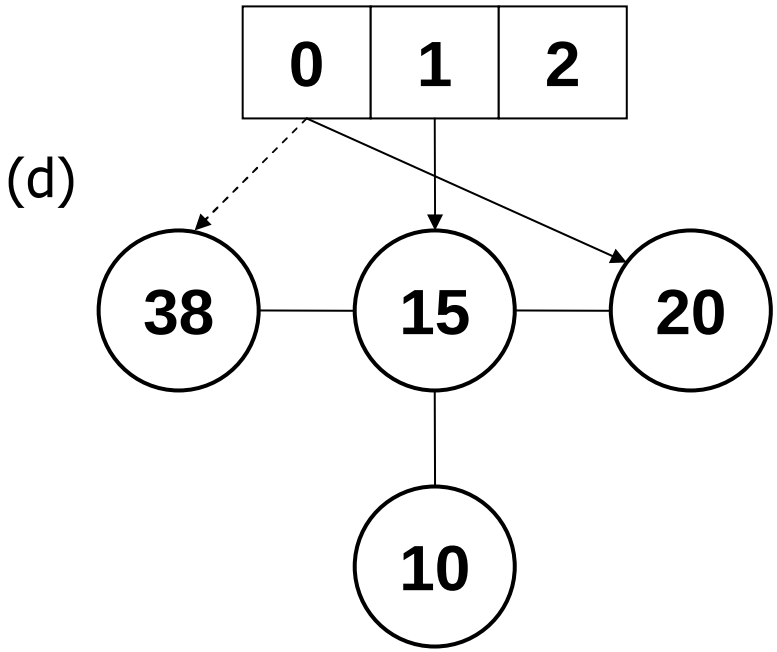


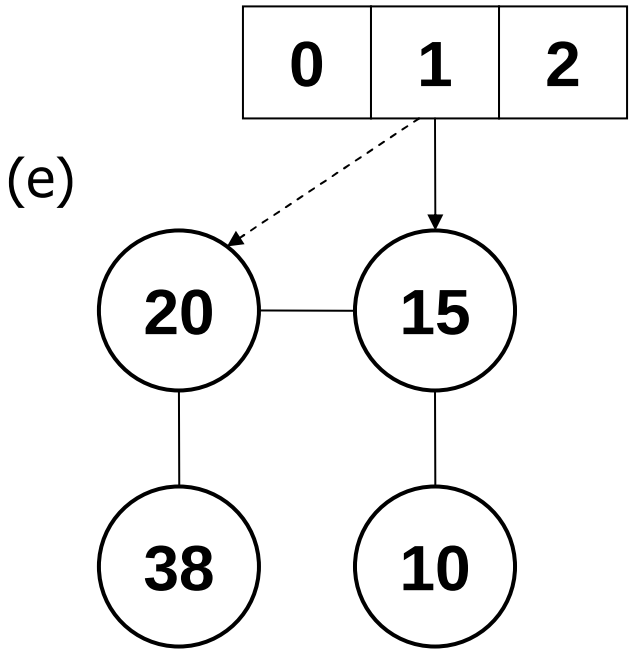
ExtrairMínimo (cont.)

(c)



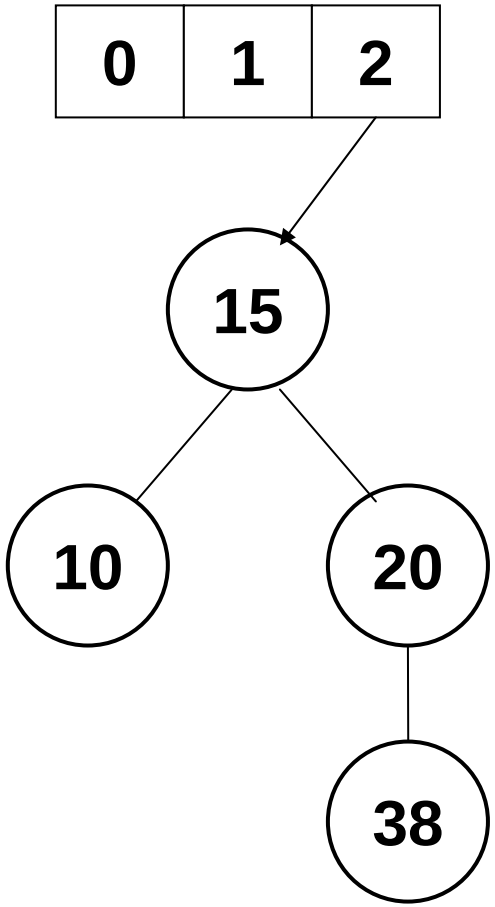
ExtrairMínimo (cont.)





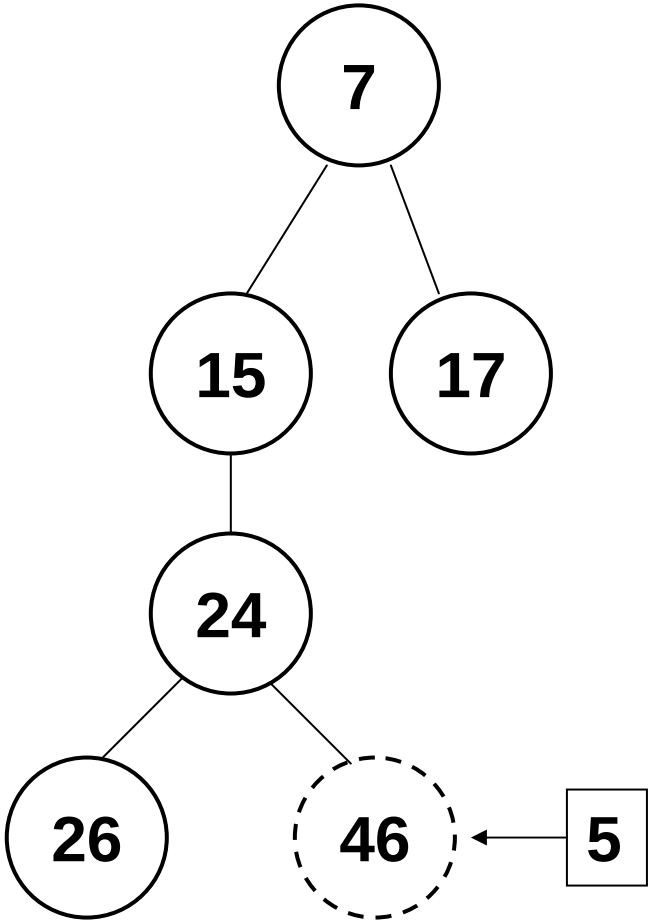
ExtrairMínimo (cont.)

(f)



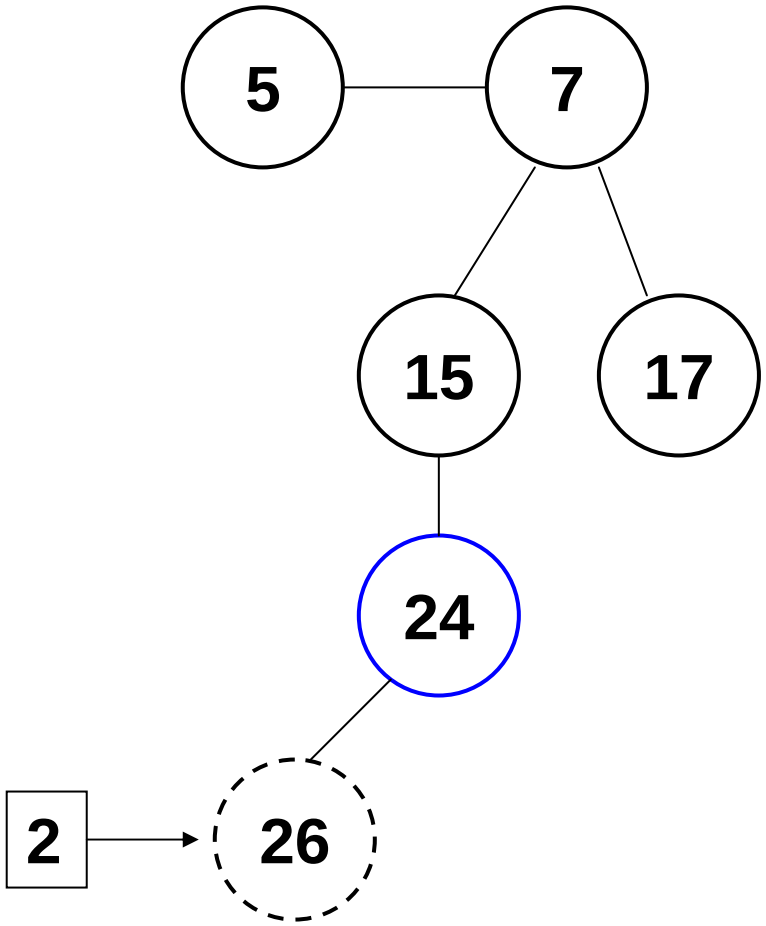
DiminuirChave

(a)

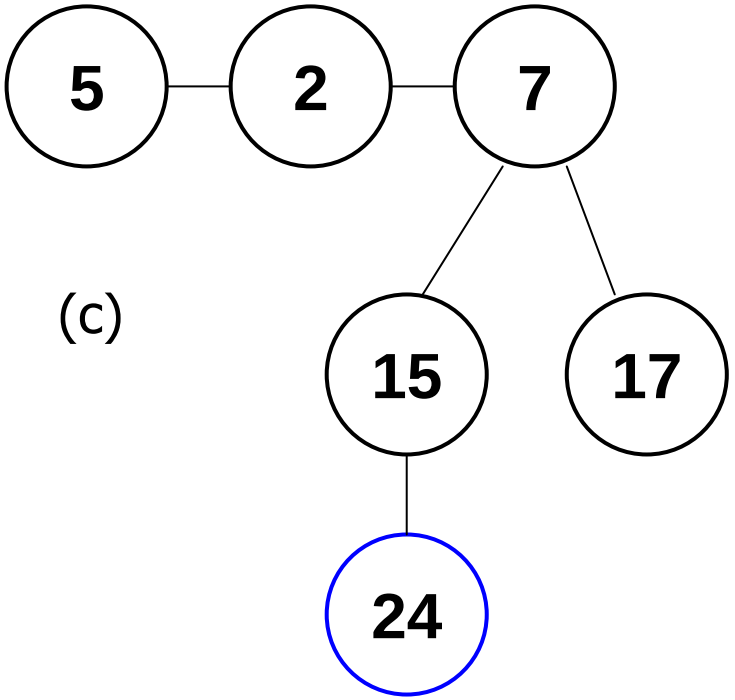


DiminuirChave (cont.)

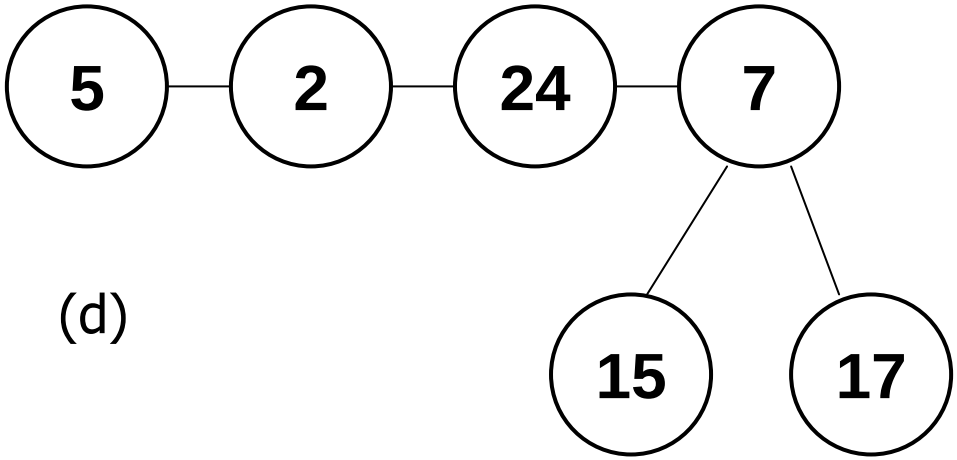
(b)



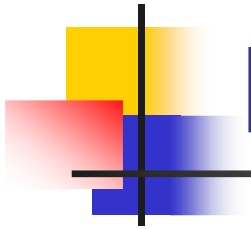
DiminuirChave (cont.)



(c)

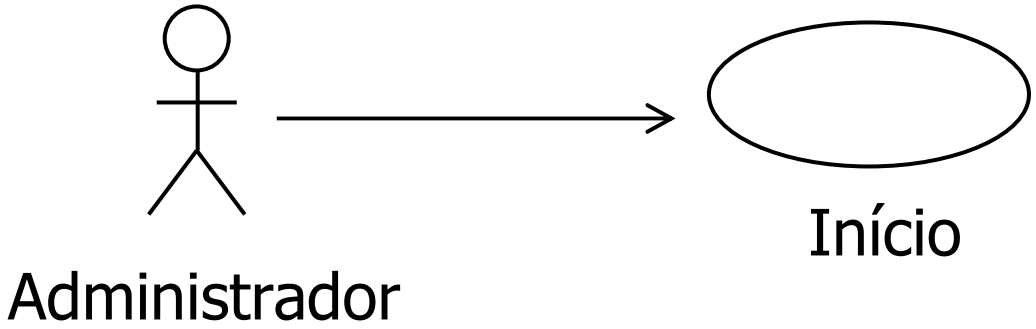


(d)

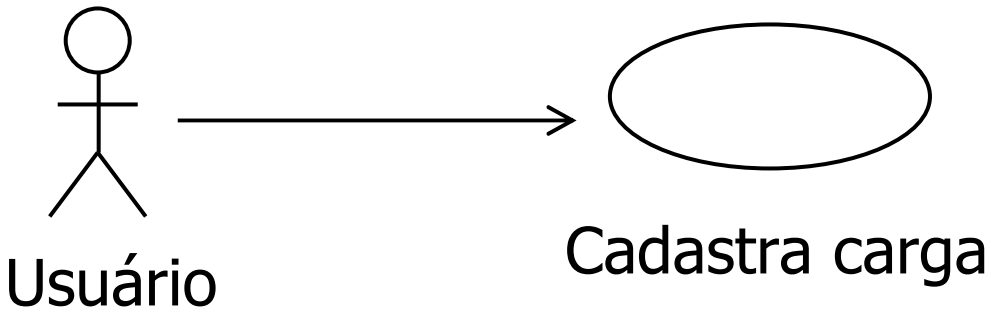


Protótipo

Caso de uso “Início”



Caso de uso “Cadastra carga”



Caso de uso “Cadastra entrega”

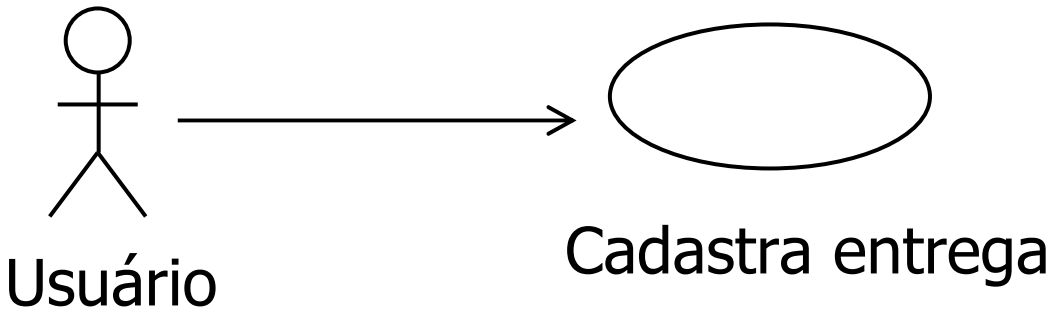


Diagrama de seqüência “Início”

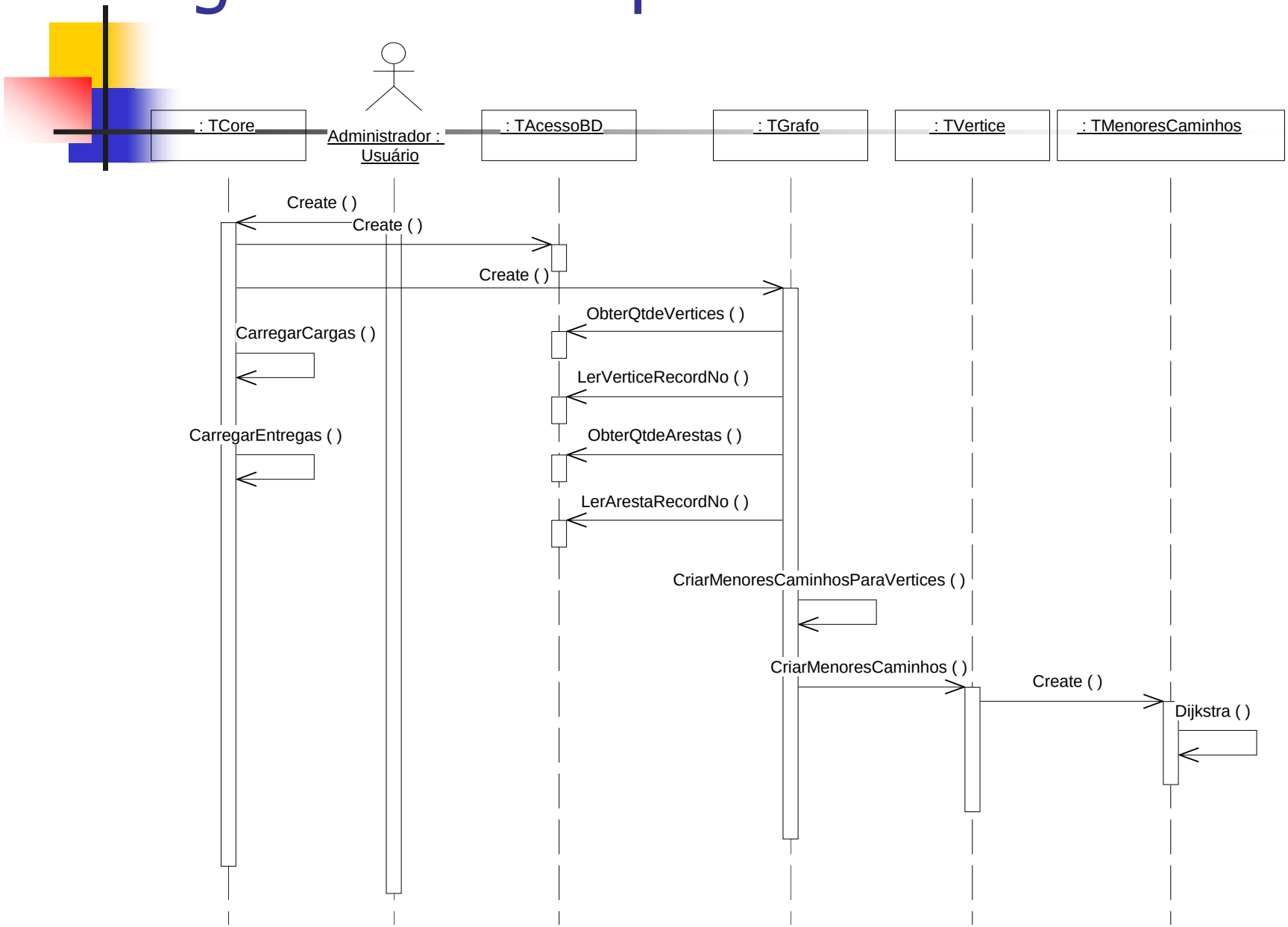


Diagrama de seqüência “Cadastrar carga”

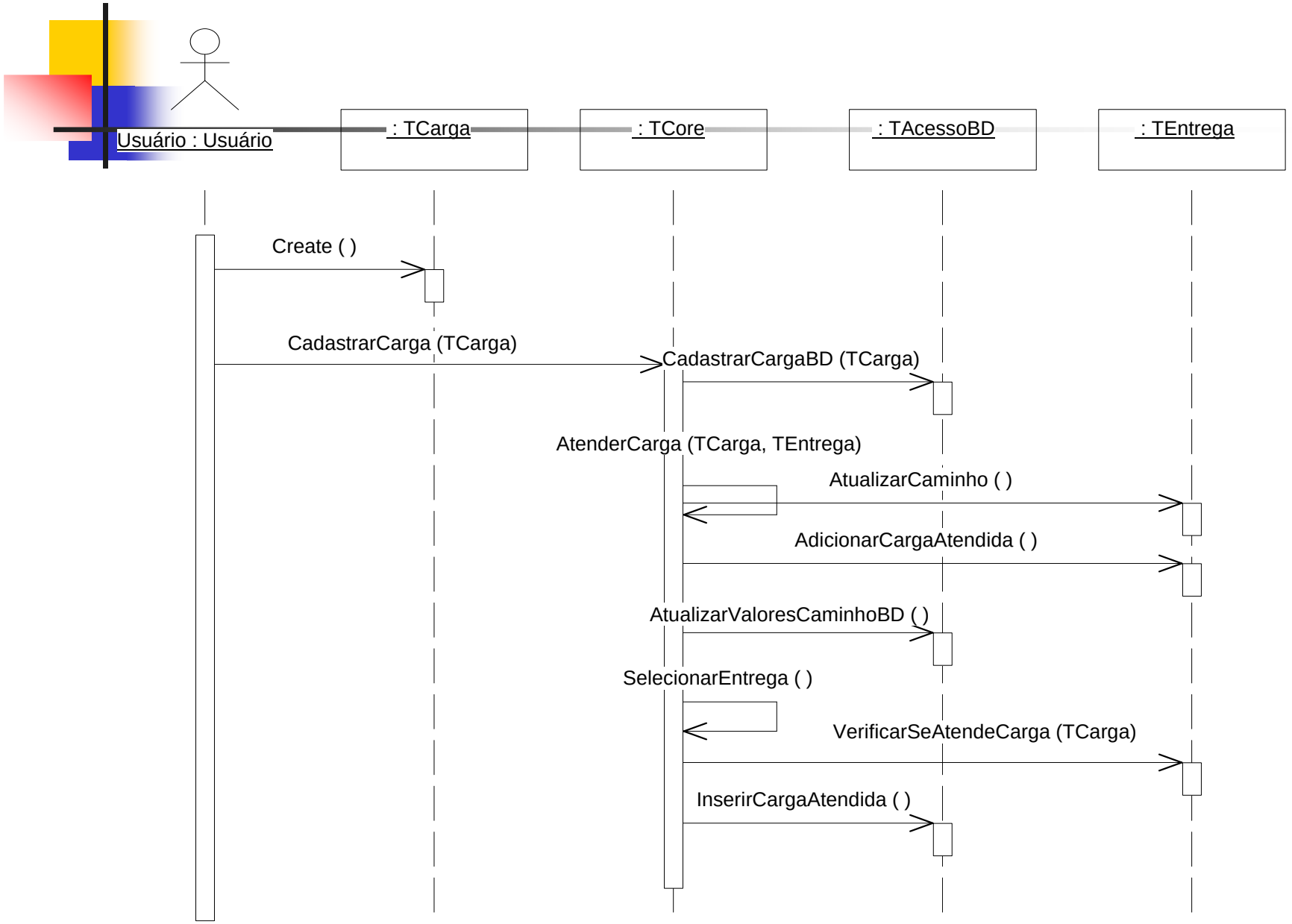
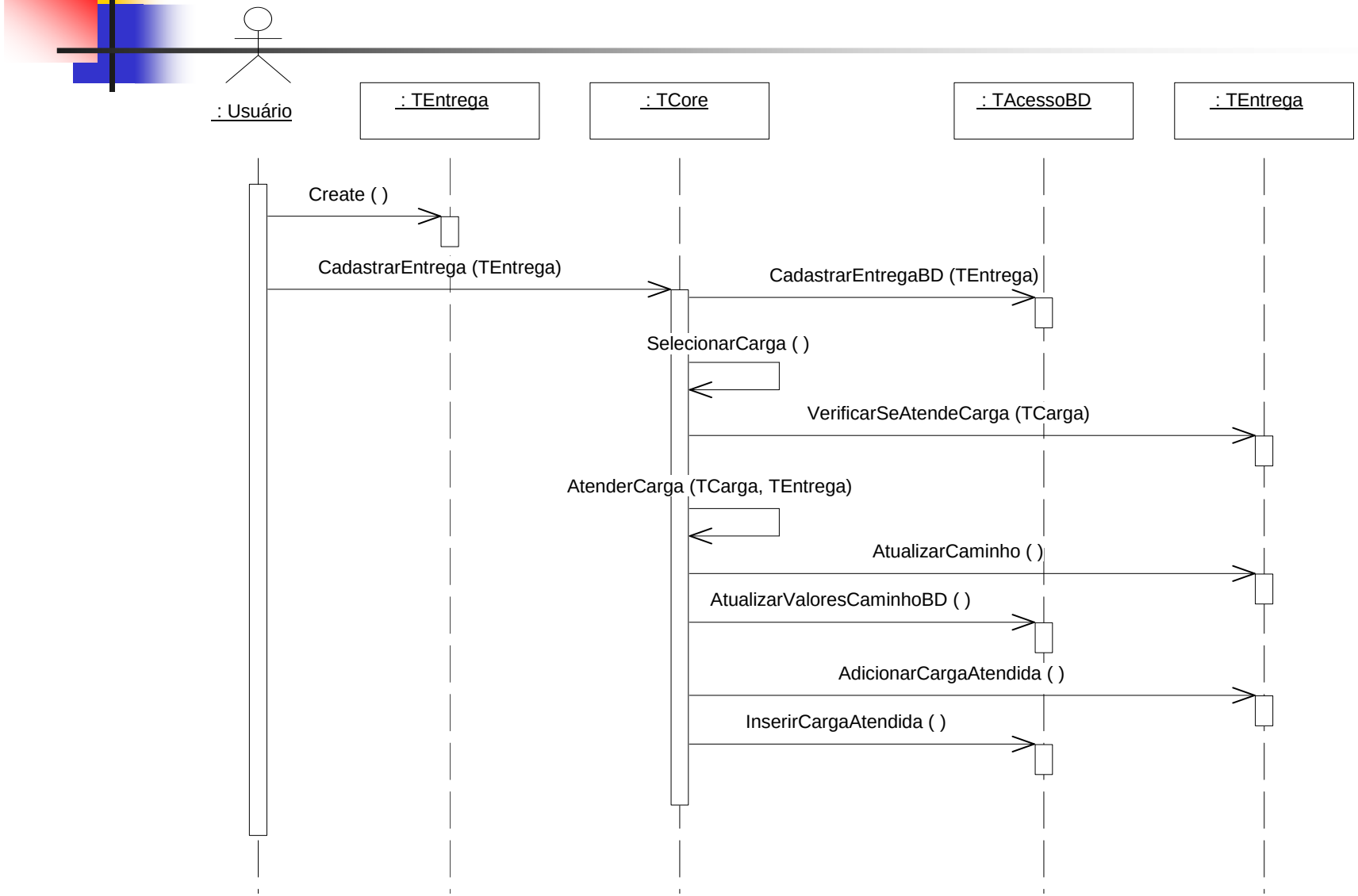
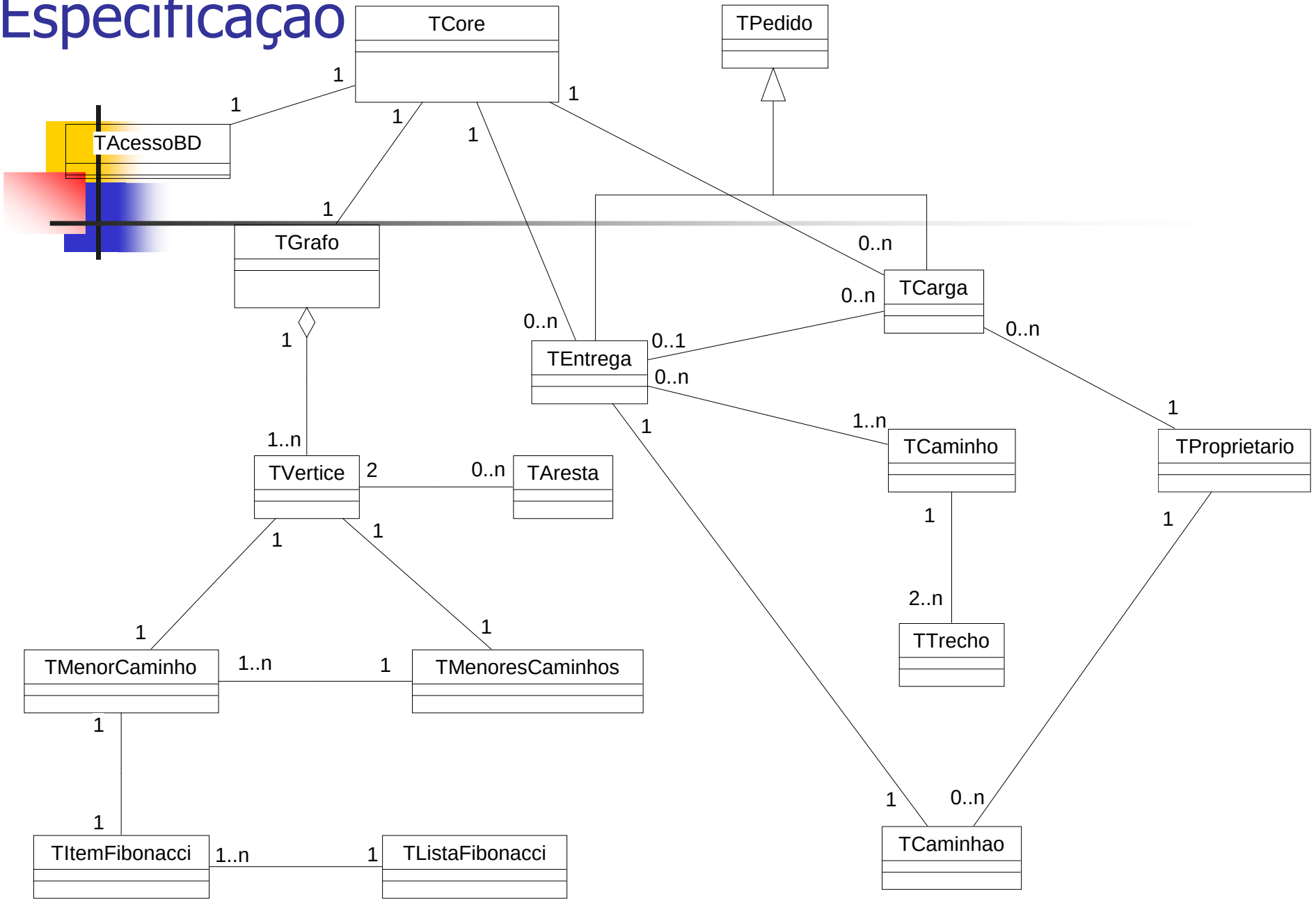


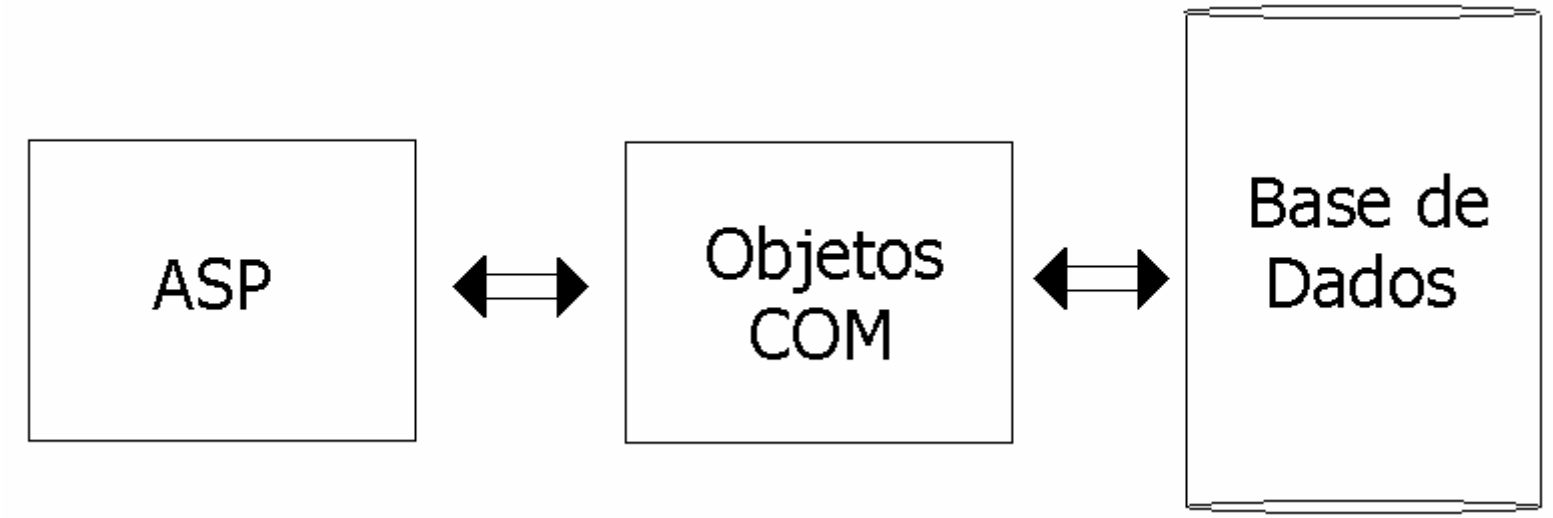
Diagrama de seqüência “Cadastrar entrega”

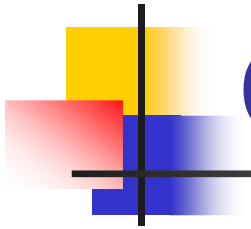


Especificação



Fluxo macro

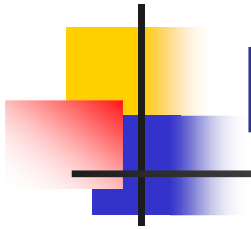




Conclusões

•O objetivo principal, que é implementar um protótipo que auxilie a distribuição de cargas, foi atendido.

•O algoritmo de *Dijkstra* com listas de *Fibonacci* mostrou-se bastante eficiente. As listas de *Fibonacci* contribuem para a otimização do algoritmo de *Dijkstra*, porém, sua implementação pode ser considerada complexa.

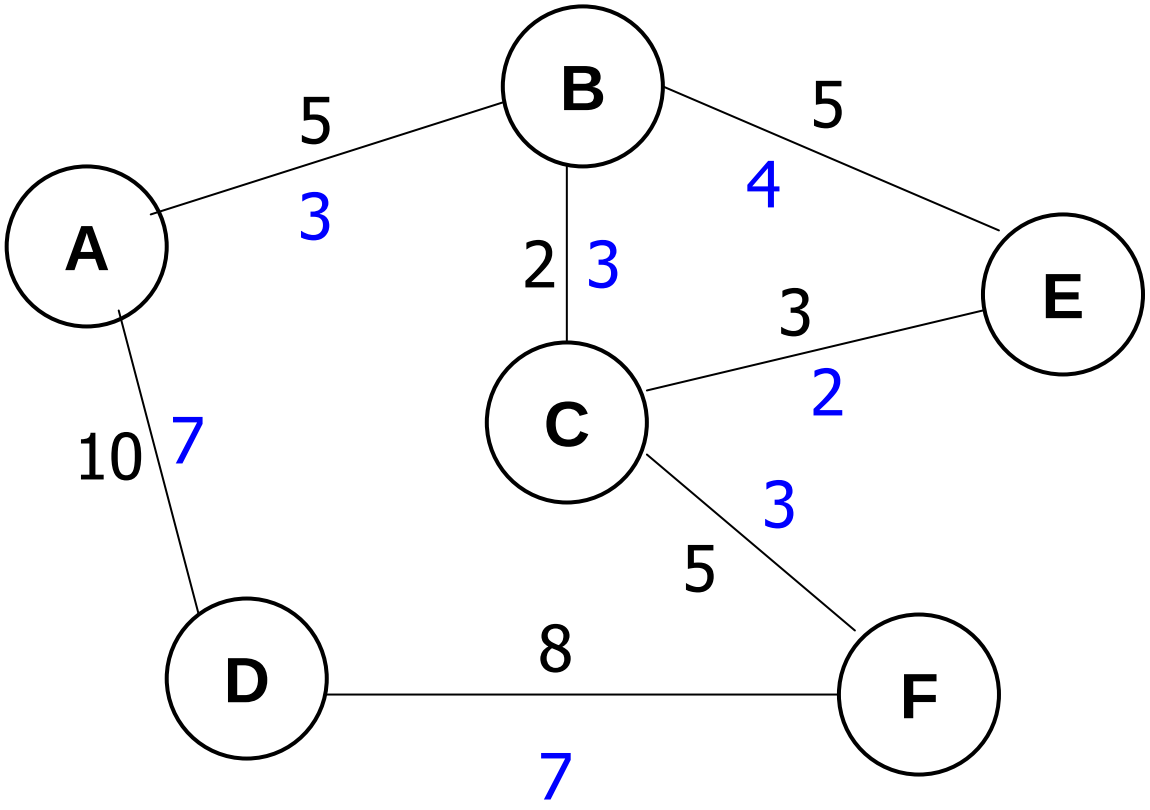


Extensões

- Estudar outros algoritmos de menor caminho e/ou utilizar outros tipos de filas de prioridades.

- Planejar entregas que atendam as cargas cadastradas.

Apresentação do Protótipo



Apresentação do Protótipo (cont.)