Algoritmos - Aula 10

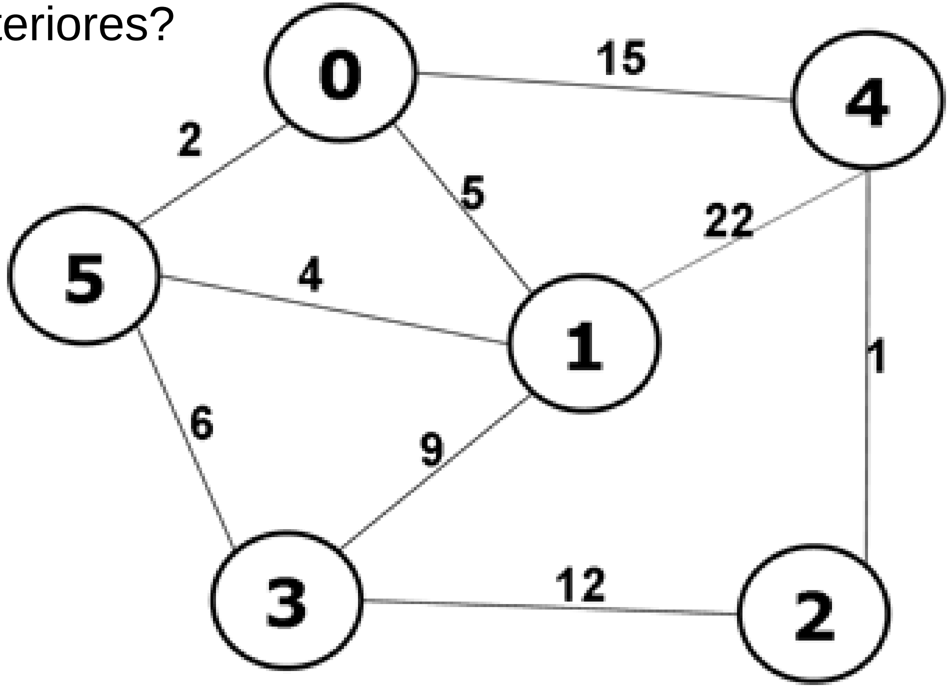
Fernando Raposo

Vamos ver

- Grafos com Pesos
- Otimização
- Algoritmos Gulosos (Greedy)
  - Conceitos
  - Aplicações
- Otimizações
- Menor Caminho
- Exercício Prático

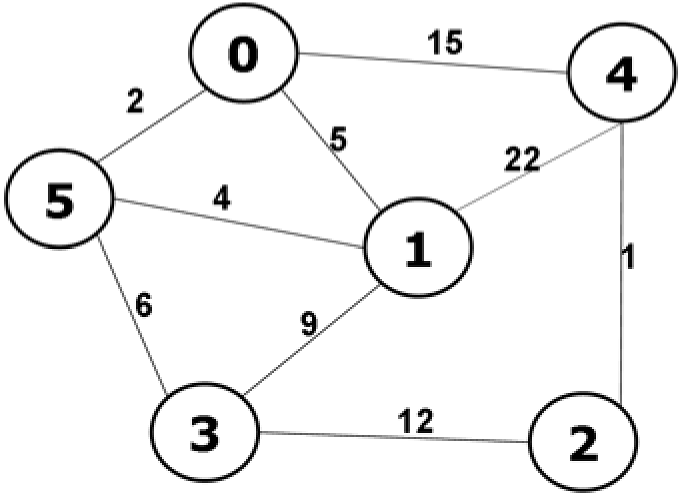
Grafos com Pesos

- Ainda não vimos grafos como o abaixo… qual a sua diferença para os anteriores?



Grafos com Pesos

- Quantos caminhos possíveis existem, por exemplo, partindo do vértice “5” até o vértice “2”?
  - 5 => 1 => 3 =>2 ou;
  - 5 => 3 => 2 ?
- Qual o “custo” ou qual a distância entre eles?
- Qual o **MENOR** custo entre eles?



Otimização

- Pelo visto estamos buscando uma solução **razoável** para esta distância não é?
- Queremos **otimizar** nossos recursos buscando o menor caminho.
- Assim, vamos simplificar um pouco o problema de otimização

Problema do Troco

1. Você trabalha em uma loja e precisa dar um troco de R$289,00 para um cliente;
2. Há disponíveis (ilimitadas) notas de 100, 50, 25, 5 e 1;
3. Como pagar este troco **com o menor número de notas possível**?

Otimização

- A pessoa geralmente vai “preenchendo” o valor inicialmente pelas notas maiores:
    - 2 notas de 100;
    - 1 nota de 50;
    - 1 nota de 25;
    - 2 notas de 5;
    - 1 nota de 1;
- 289 - **100** = 189 - **100** = 89 - **50** = 39 - **25** = 14 - **5** = 9 - **5** = 4 - **1** = 3 -**1** = 2 - **1** = 1 - **1** = Troco dado
- Essa é chamada de estratégia “Gulosa”, ou Gananciosa, você utiliza logo os valores que te dão maior benefício sem arrependimento (Greedy).

Otimização

- Mas… nem sempre a abordagem Gulosa funciona…
- Suponha que temos que dar um troco de R$20,00 e temos notas ilimitadas de: 100, 50, 24, 12, 5, 1.
- Estratégia Gulosa Rodando:
  - 20 - **12** = 8 - **5** = 3 - **1** = 2 - **1** = 1 - **1** = Troco dado
  - 1 nota de 12
  - 1 nota de 5
  - 3 notas de 1
- Mas esta solução não é a melhor. Qual seria?

Otimização

- 4 notas de 5 seriam suficientes para dar o troco;
- **Menos** notas que a solução Gulosa;
- Logo chegamos a conclusão que não podemos usar a estratégia gulosa amplamente;
- É necessário que se prove a sua eficácia.

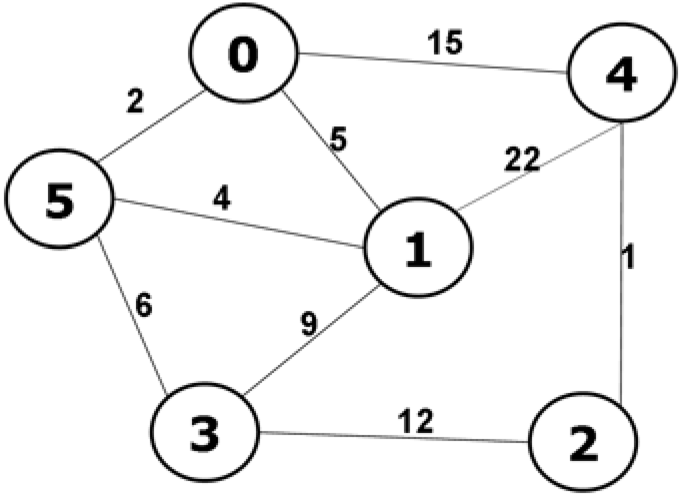
Otimização: Solução

TROCO (N)

```
C ← {100, 50, 25, 10, 5, 1}
Moedas ← {}
Soma ← 0
ENQUANTO Soma ≠ N
    x = máximo de C tal que (Soma + x ≤ N)
    Moedas ← Moedas + {x}
    Soma ← Soma + x
RETORNE Moedas
```

Voltando para Grafos...

- Dado o grafo exibido como descobrir **qual o menor caminho entre dois vértices**?
- Por exemplo: entre 3 e 4
- Vamos enumerar as possibilidades?



Menor Caminho

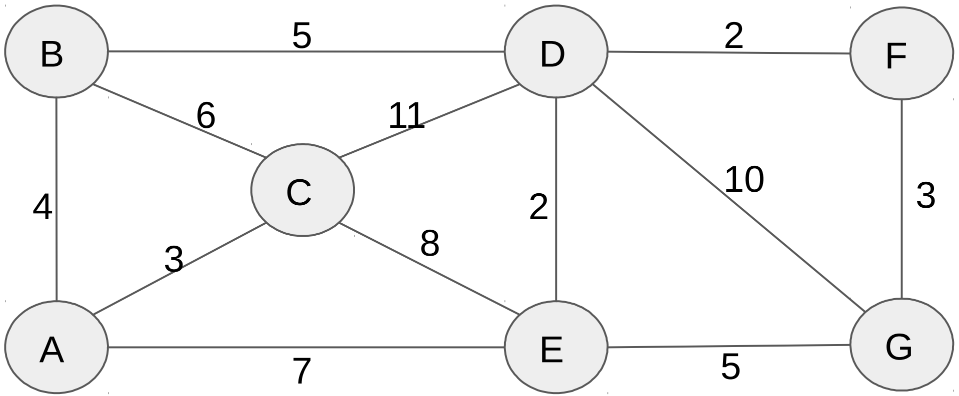
- É possível utilizar a estratégia **Gulosa** para resolver o problema de menor caminho entre dois vértices;
- Concebido pelo cientista Holandês Edsger Dijkstra em 1956;
- Algoritmo:
  - Premissas:
    - Teremos dois conjuntos:
      - Conjunto dos vértices que fazem parte do menor caminho;
      - Conjunto dos vértices que **não** fazem parte do menor caminho;
  - Passos:
    - Toda vez, procurar um vértice no conjunto de vértices que não fazem parte do menor caminho e trazê-lo para o outro grupo.

Menor Caminho entre ***s*** e ***t***: Algoritmo

1. Criar um conjunto arvoreMenorCaminho, que contém os vértices que vão aos poucos fazendo parte do menor caminho de ***s*** a ***t*** (vazio no ínicio);
2. Marcar a distância de cada vértice ao vértice de início (inicialmente atribuir infinito);
3. Atribuir a distância Zero de ***s*** a ***s***;
4. Enquanto conjunto arvoreMenorCaminho não possuir todos os vértices…
    1. Escolher um vértice ***u*** que não está no conjunto arvoreMenorCaminho e que tem a menor distância desde ***s***;
    2. Incluir ***u*** no conjunto arvoreMenorCaminho;
    3. Para cada vértice ***v*** adjacente a ***u***:
        1. Se a soma da distância de *u* + aresta ***u-v*** for menor que a distância atual do vértice, então atualizá-la;

Menor Caminho

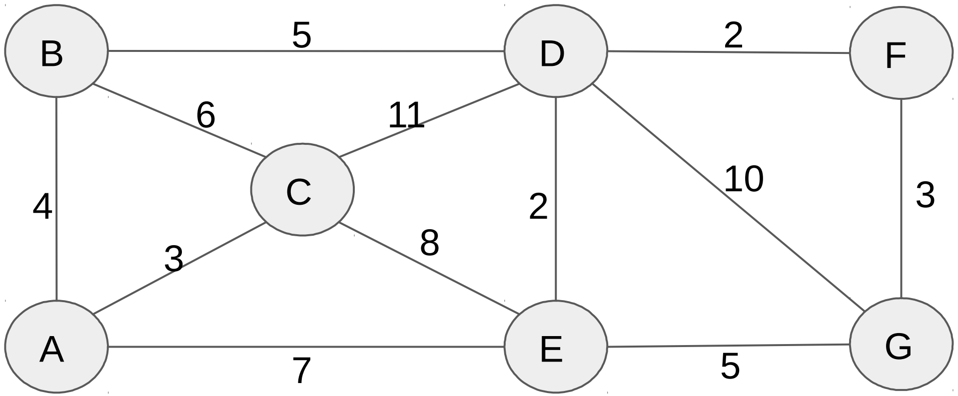
- Qual o menor caminho de “**A**” até “**F**” ?



Menor Caminho, segundo Dijkstra

A	B	C	D	E	F	G
∞	∞	∞	∞	∞	∞	∞

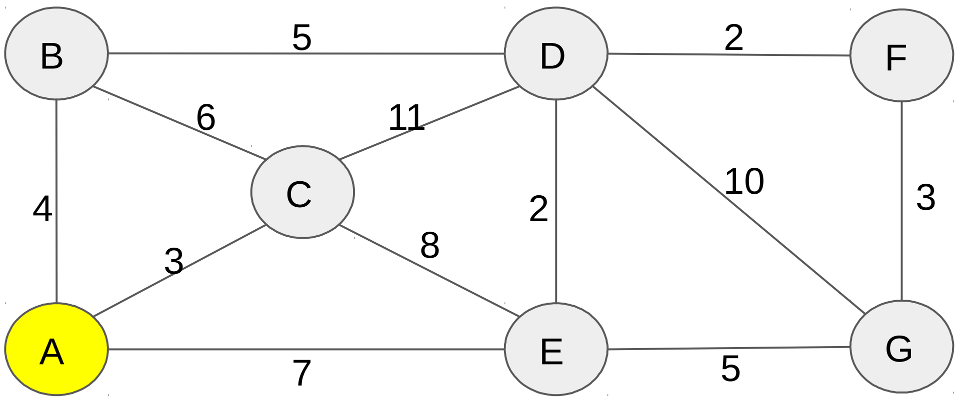
Iniciamos o Algoritmo com uma tabela de distâncias, inicialmente todas infinito.



Menor Caminho, segundo Dijkstra

A	B	C	D	E	F	G
0	**∞**	**∞**	**∞**	**∞**	**∞**	**∞**

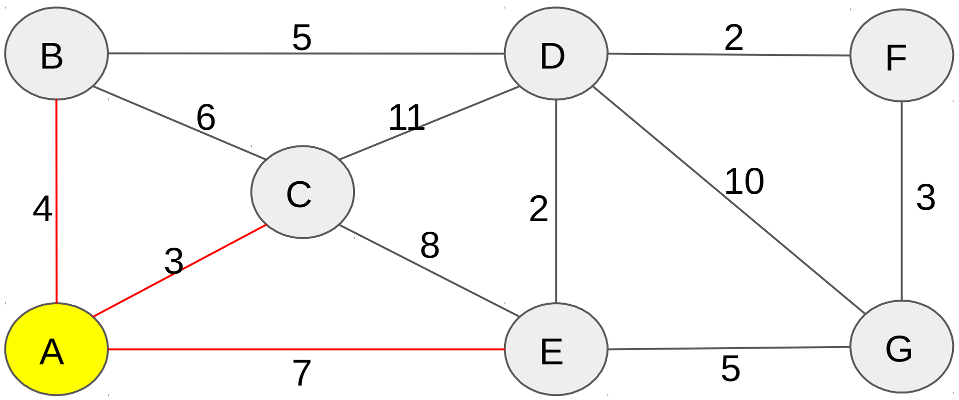
Calculamos a distância de "A" para "A", ou seja, zero. E verificamos seus vizinhos "B", "C" e "E".



Menor Caminho, segundo Dijkstra

A	B	C	D	E	F	G
0	**∞**	**∞**	**∞**	**∞**	**∞**	**∞**

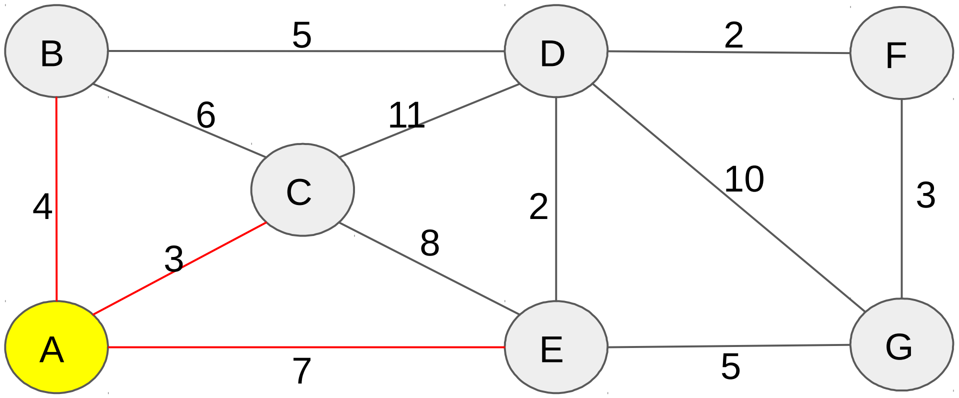
Se a distância até "A" for menor que a distância atual (infinito) , atribuir esta distância.



Menor Caminho, segundo Dijkstra

A	B	C	D	E	F	G
0	**4**	**3**	**∞**	**7**	**∞**	**∞**

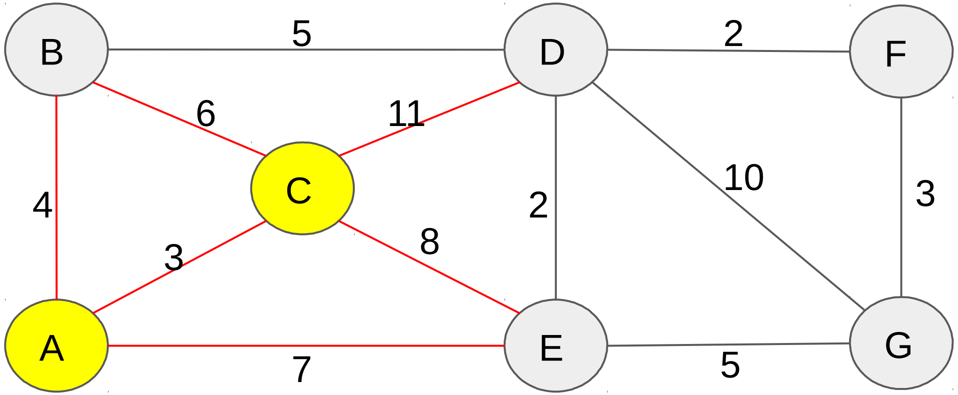
Agora, utilizando a estratégia Gulosa, vamos visitar o vértice com a menor distância calculada. Neste caso “C”



Menor Caminho, segundo Dijkstra

A	B	C	D	E	F	G
0	**4**	**3**	**∞**	**7**	**∞**	**∞**

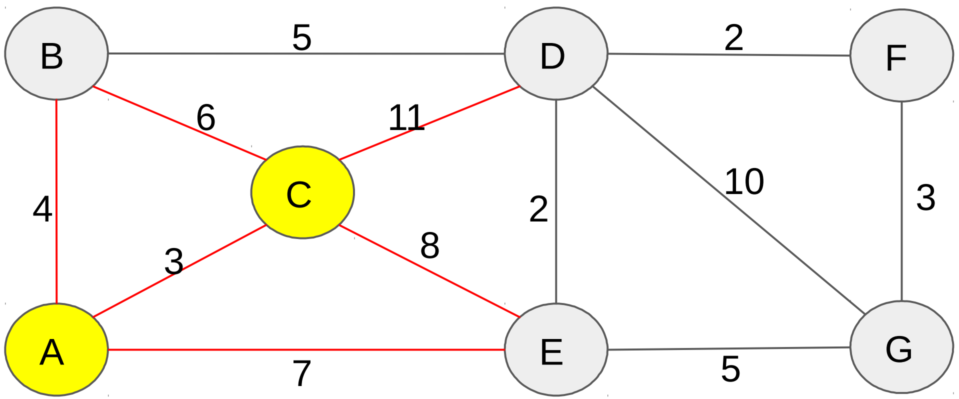
Vamos calcular as distâncias da vizinhança de “C” para “A” e substituir caso o cálculo da distância dê menor que o atualmente calculado. ACB = 9; ACE = 11; ACD = 14



Menor Caminho, segundo Dijkstra

A	B	C	D	E	F	G
0	**4**	**3**	**14**	**7**	**∞**	**∞**

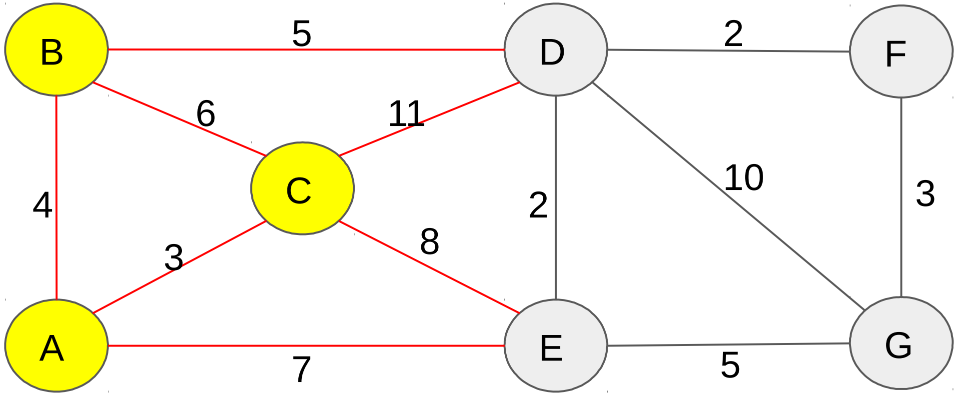
Vamos calcular as distâncias da vizinhança de “C” para “A” e substituir caso o cálculo da distância dê menor que o atualmente calculado. ACB = 9; ACE = 11; **ACD = 14**



Menor Caminho, segundo Dijkstra

A	B	C	D	E	F	G
0	**4**	**3**	**14**	**7**	**∞**	**∞**

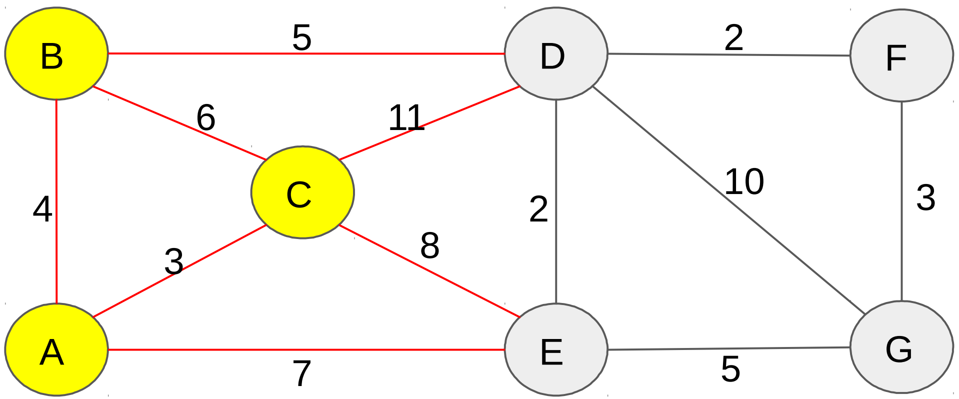
Não há mais o que fazer em “C” Então, utilizando a estratégia Gulosa, vamos visitar o vértice com a menor distância calculada. Neste caso “B”, pois “C” já foi visitado.



Menor Caminho, segundo Dijkstra

A	B	C	D	E	F	G
0	**4**	**3**	**14**	**7**	**∞**	**∞**

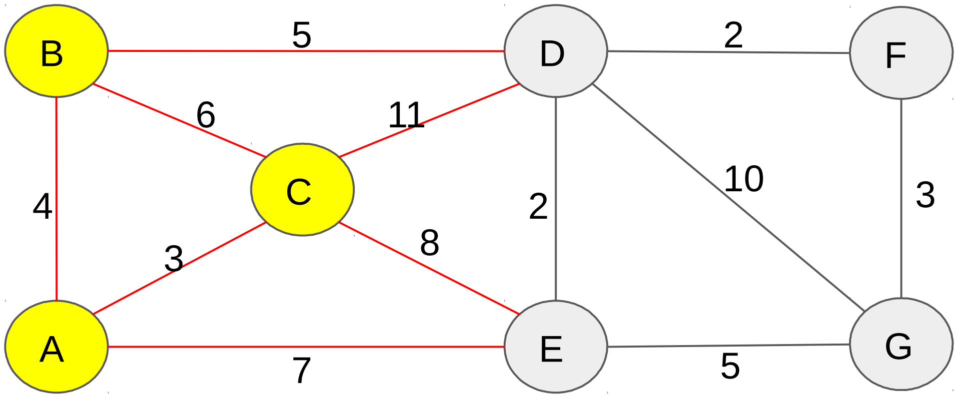
Vamos calcular as distâncias da vizinhança de “B” para “A” e substituir caso o cálculo da distância dê menor que o atualmente calculado. **ABD = 9**; 9 < 14, vamos trocar...



Menor Caminho, segundo Dijkstra

A	B	C	D	E	F	G
0	**4**	**3**	**9**	**7**	**∞**	**∞**

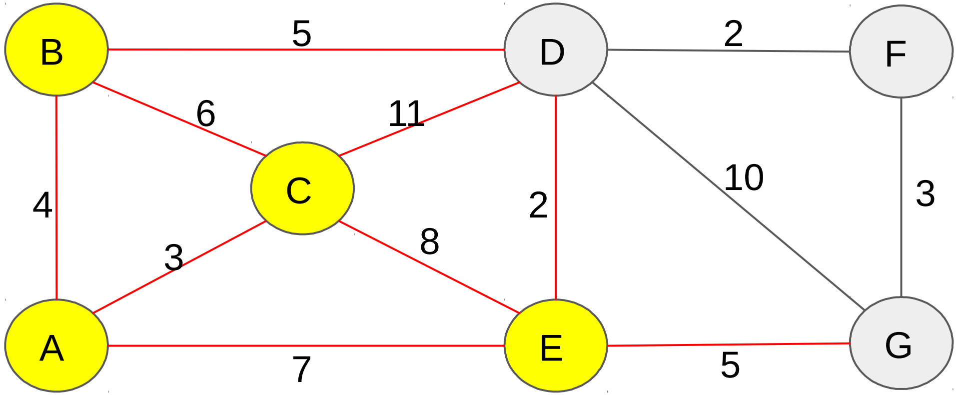
Trocado! Não há mais o que fazer em “B” Então, utilizando a estratégia Gulosa, vamos visitar o vértice com a menor distância calculada. Neste caso “E”, pois “B” e “C” já foram.



Menor Caminho, segundo Dijkstra

A	B	C	D	E	F	G
0	**4**	**3**	**9**	**7**	**∞**	**∞**

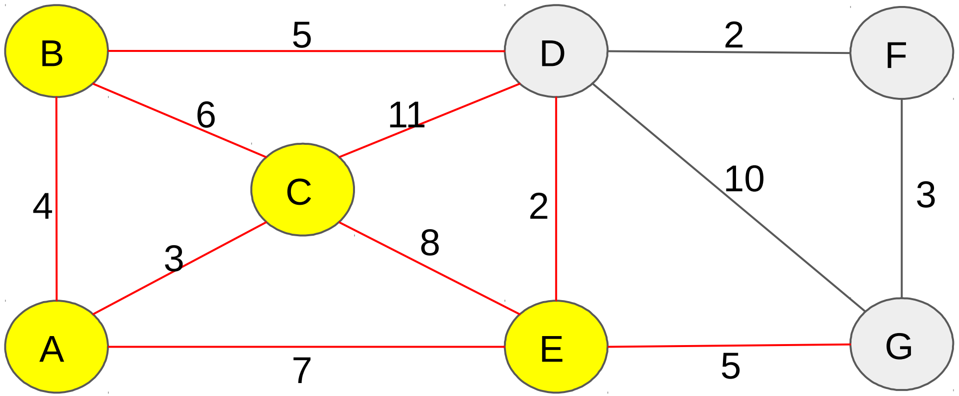
Vamos calcular as distâncias da vizinhança de “E” para “A” e substituir caso o cálculo da distância dê menor que o atualmente calculado. AED = 9; **AEG = 12**, vamos trocar...



Menor Caminho, segundo Dijkstra

A	B	C	D	E	F	G
0	**4**	**3**	**9**	**7**	**∞**	**12**

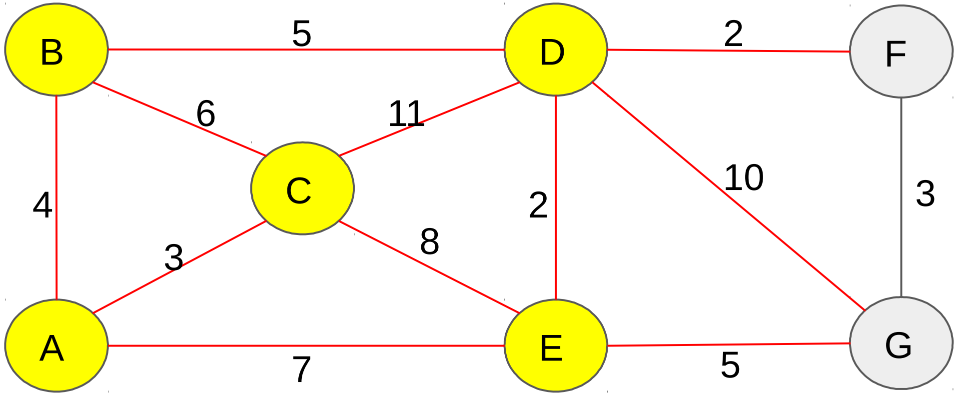
Trocado! Não há mais o que fazer em “E” Então, utilizando a estratégia Gulosa, vamos visitar o vértice com a menor distância calculada. Neste caso “D”, pois outros já foram.



Menor Caminho, segundo Dijkstra

A	B	C	D	E	F	G
0	**4**	**3**	**9**	**7**	**∞**	**12**

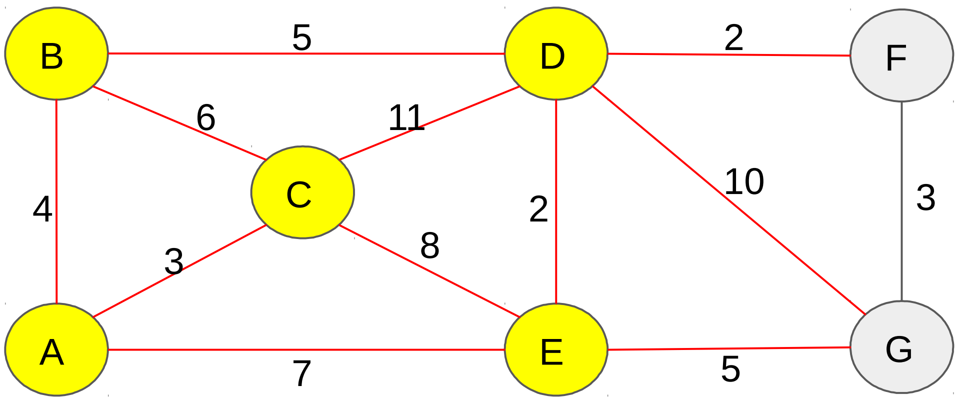
Vamos calcular as distâncias da vizinhança de “D” para “A” e substituir caso o cálculo da distância dê menor que o atualmente calculado. **ABDF = 11**; ABDG=19, vamos trocar...



Menor Caminho, segundo Dijkstra

A	B	C	D	E	F	G
0	**4**	**3**	**9**	**7**	**11**	**12**

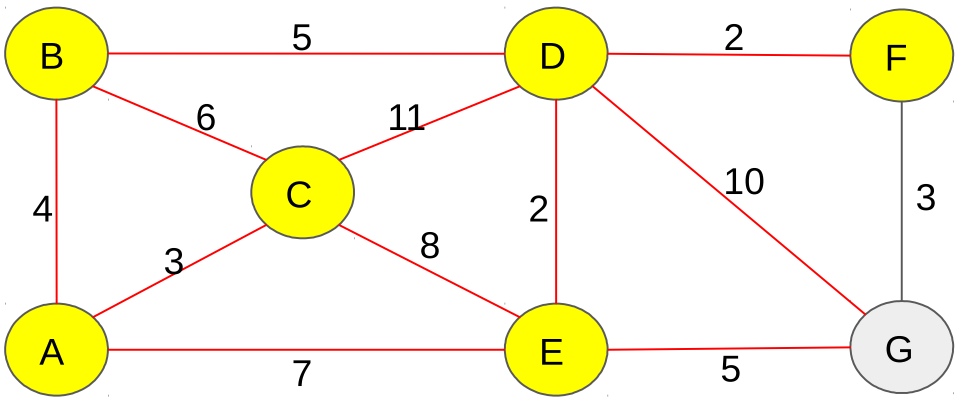
Vamos deixar o algoritmo rodar... Pode ser que haja um caminho menor por G. Então, utilizando a estratégia Gulosa, vamos visitar o vértice com a menor distância calculada **F**



Menor Caminho, segundo Dijkstra

A	B	C	D	E	F	G
0	**4**	**3**	**9**	**7**	**11**	**12**

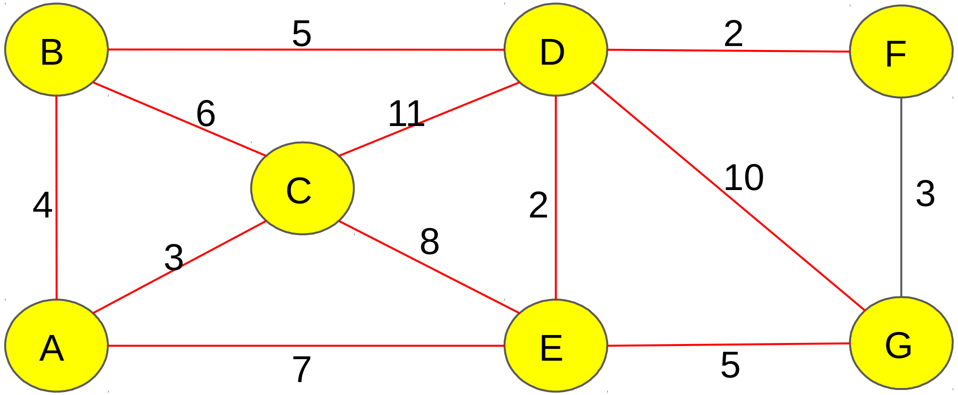
Chegar em F por G é inviável, pois o caminho resultaria em 15… o caminho atual é 11.



Menor Caminho, segundo Dijkstra

A	B	C	D	E	F	G
0	**4**	**3**	**9**	**7**	**11**	**12**

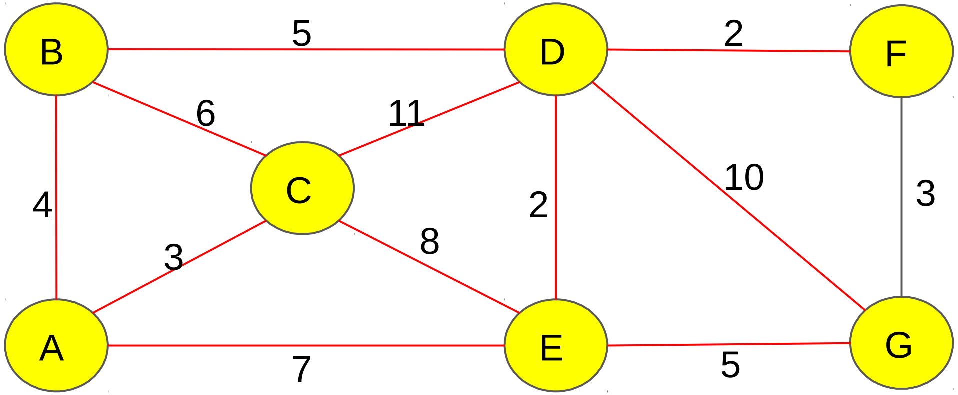
Só sobrou o vértice G para visitar… e o caminho até ele por D ou F não é melhor que o caminho que já existe.



Menor Caminho, segundo Dijkstra

A	B	C	D	E	F	G
0	**4**	**3**	**9**	**7**	**11**	**12**

Opa! Chegamos ao destino. Acabamos de descobrir o menor caminho de “A” -> “F”: É com um custo de 11 e Visitamos = {A, B, C , D, E}



Parte Prática...

- Vamos implementar o algoritmo do Menor Caminho para descobrir o menor caminho do vértice.

```
dijkstra.adicionaAresta( 0, 1);
dijkstra.adicionaAresta( 0, 7);
dijkstra.adicionaAresta( 1, 7);
dijkstra.adicionaAresta( 1, 2);
dijkstra.adicionaAresta( 7, 8);
dijkstra.adicionaAresta( 7, 6);
dijkstra.adicionaAresta( 6, 5);
dijkstra.adicionaAresta( 2, 3);
dijkstra.adicionaAresta( 2, 8);
dijkstra.adicionaAresta( 2, 5);
dijkstra.adicionaAresta( 5, 3);
dijkstra.adicionaAresta( 5, 4);
dijkstra.adicionaAresta( 3, 4);
```


Thank You!