Teoria dos Grafos
Aula 11

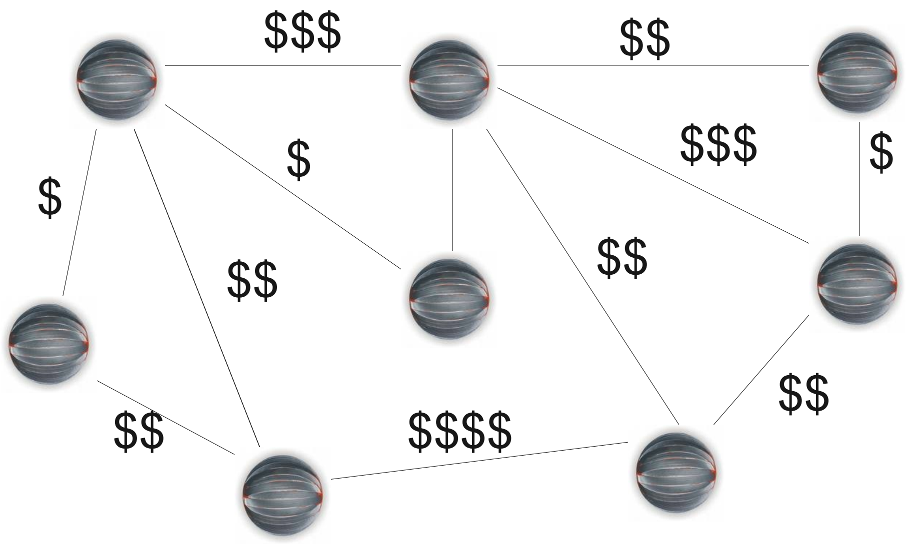
Aula passada

- Grafos com pesos
- Dijkstra
- Implementação
- Fila de prioridades e Heap
- Dijkstra (o próprio)

Aula de hoje

- MST
- Algoritmos de Prim e Kruskal
- Propriedades da MST

Projetando uma Rede



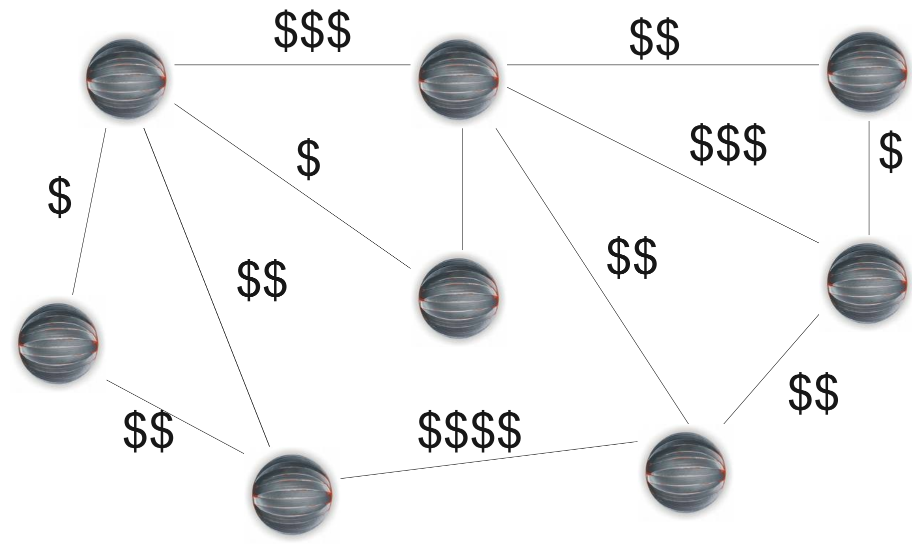
- Conjunto de localidades (ex. cidades)
- Custo para conectá-los diretamente (ex. construir estradas)

- Garantir a conectividade
  - de qualquer lugar, chegamos a qualquer outro

- **Problema:** Como conectar as localidades de forma a minimizar o custo total?

Projetando uma Rede

- Abstração via grafos
  - Vértices: localidades
  - Arestas com pesos: custo de conexão direta entre localidades



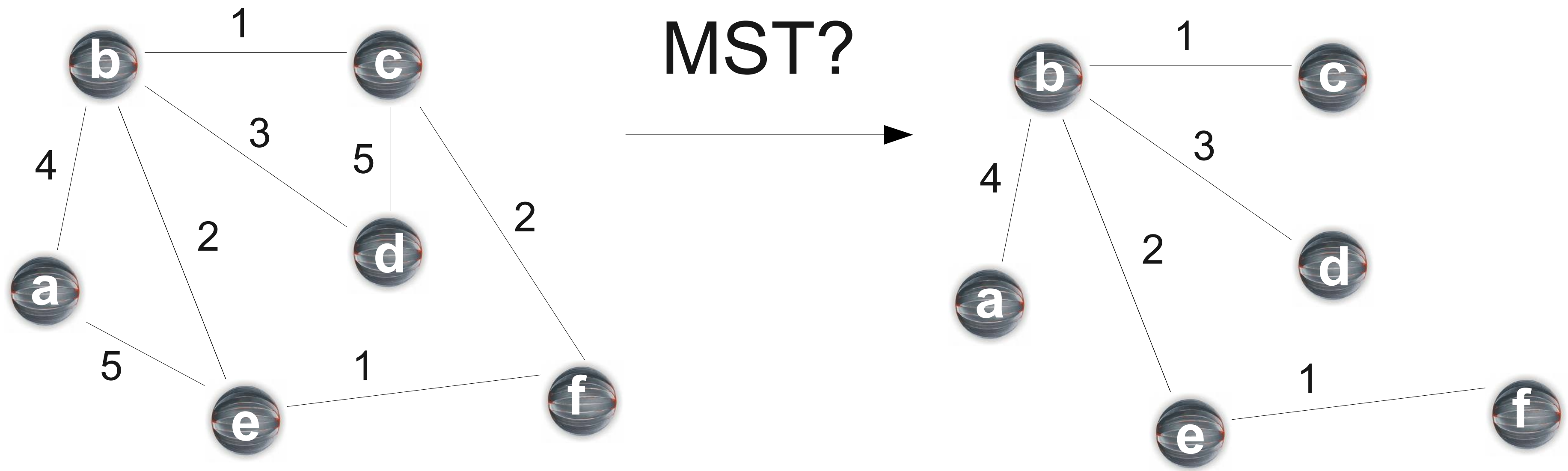
- Como será a “cara” do resultado?
- Subgrafo de G, Árvore, Árvore geradora!

- Qualquer árvore? ← **Não!**

Árvore Geradora de Custo Mínimo!

MST

- Minimum Spanning Tree (MST)
  - árvore geradora de custo mínimo
- Exemplo



Custo desta MST? **11**

- MST é única?

Descobrindo a MST

- **Problema:** Obter a MST de um grafo
- Grafo com pesos idênticos? ← *BFS to the rescue!*
  - qualquer árvore geradora tem custo mínimo
- Grafo com pesos diferentes?

Idéias?

Descobrindo a MST – Idéias I

- Modificar BFS
  - BFS constrói uma árvore geradora
- Dado vértice inicial *s*
- Construir árvore geradora mínima
- Expandir "fronteira" na direção correta

Qual é a "direção" correta?

- Direção de menor custo
- Adicionar vértice que aumenta o custo total o menos possível

Descobrindo a MST – Idéias I

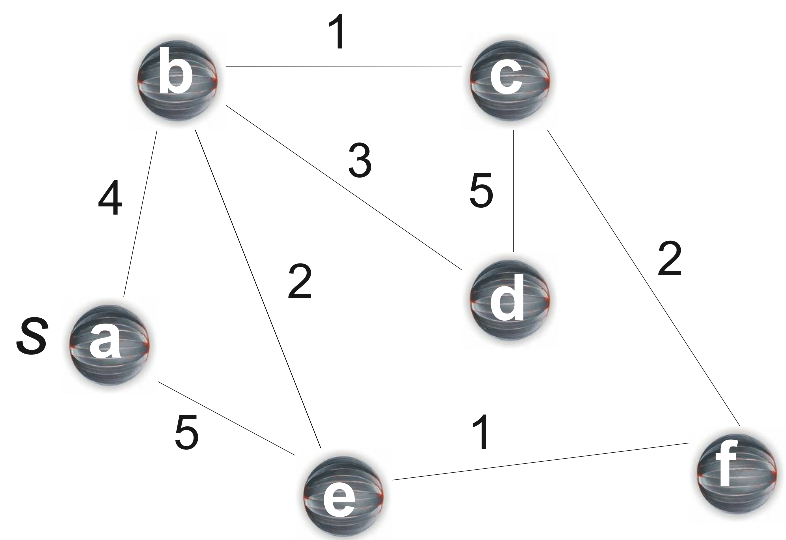
- Dado G=(V, E)
- Construir MST, T=(S, E')
- Inicialmente S e E' estão vazios
- Selecionar *s*, vértice inicial
- Adicionar vértices em T na ordem mais barata possível
  - próximo vértice aumenta custo total o mínimo possível

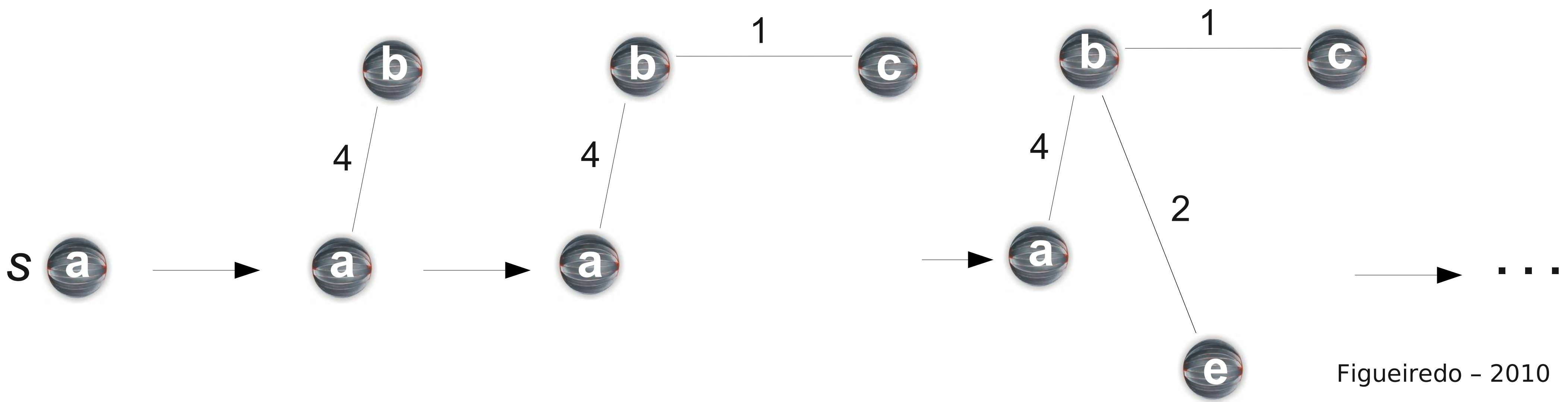
Algoritmo de *Prim*

- Muito parecido com qual algoritmo?

Algoritmo de *Prim*

- **Idéia:** crescer T de forma mais barata possível
- Exemplo





Figueiredo – 2010

Algoritmo de Prim

- Como tornar a idéia em algoritmo (eficiente)?
  - adicionar o vértice que aumenta o custo o menos possível

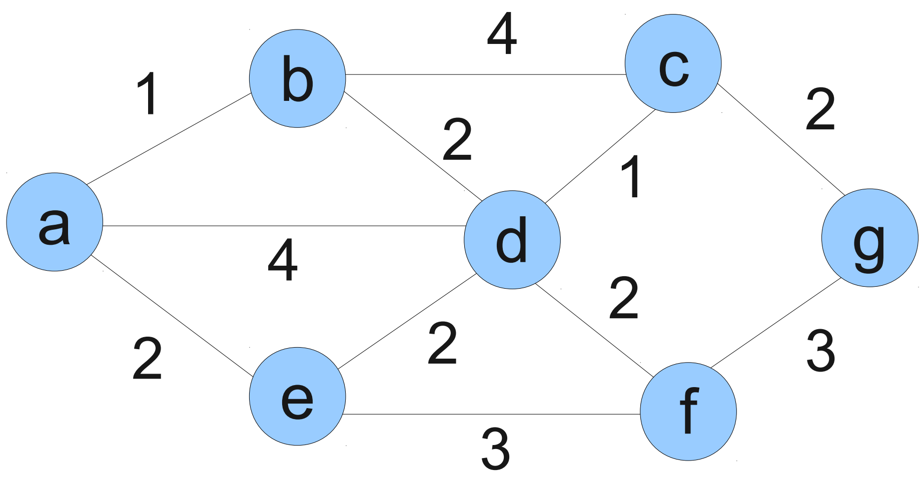
- **Idéias:**
  - Manter um conjunto de vértices da árvore
  - Manter custo para adicionar cada vértice até o momento
  - Adicionar o vértice de menor custo
  - Atualizar custos

Algoritmo de Prim

```
Prim(G, o)
Para cada vértice v
   custo[v] = infinito
Define conjunto S = 0 // vazio
custo[o] = 0
Enquanto S != V
   Selecione u em V-S, tal que custo[u] é mínimo
   Adicione u em S
   Para cada vizinho v de u faça
      Se custo[v] > w((u,v)) então
         custo[v] = w((u,v))
```

- Como o algoritmo executa?

Executando o Algoritmo



- Manter tabela com passos e custos

Complexidade

- Qual é a complexidade do algoritmo?
- Complexidade idêntica a Dijkstra

```
Prim(G, o)
Para cada vértice v
    custo[v] = infinito
Define conjunto S = 0 // vazio
custo[o] = 0
Enquanto S != V
    Selecione u em V-S, tal que custo[u] é mínima
    Adicione u em S
    Para cada vizinho v de u faça
       Se custo[v] > w((u,v)) então
          custo[v] = w((u,v))
```

- Usando filas de prioridade baseada em heap
  - n operações de remoção, m de atualização
- O((m+n)log n) = O(m log n)

Descobrindo a MST – Idéias II

- Outra abordagem, diferente de *BFS*
  - mas também gulosa
- Aresta de menor peso está na MST?
- Aresta segundo menor peso está na MST?
- Aresta de terceiro menor peso?

Cuidado com ciclos!

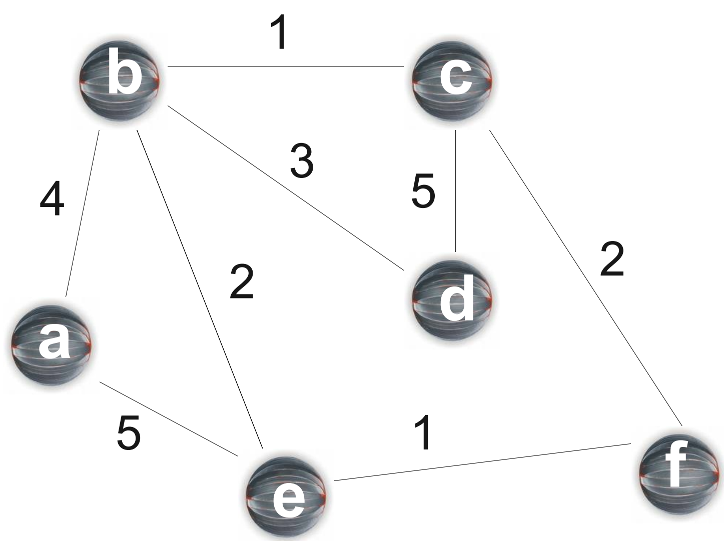
Descobrindo a MST – Idéias II

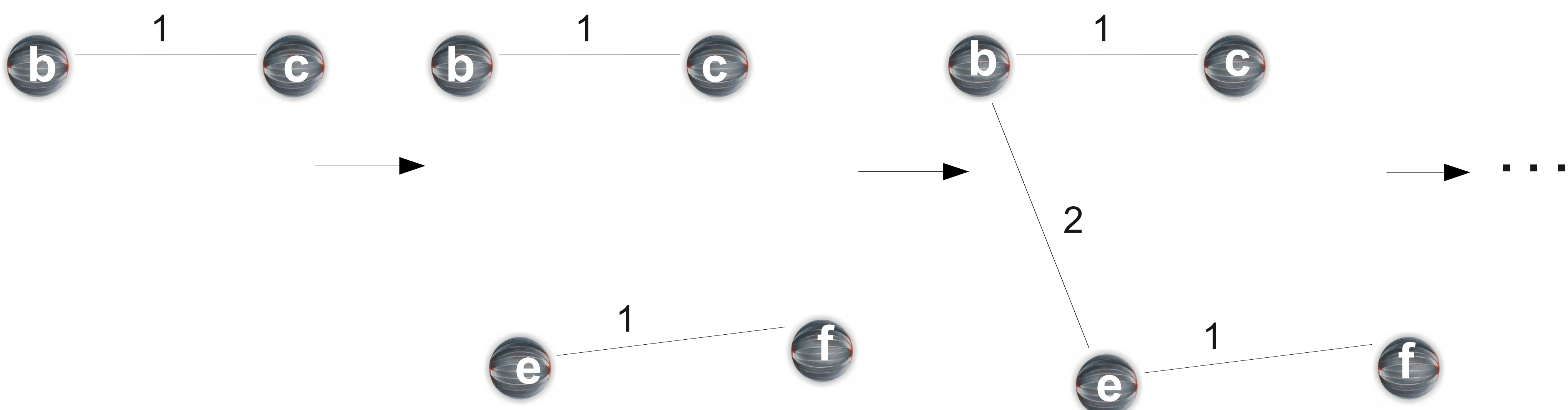
- Dado G=(V, E)
- Construir MST, T=(V, E')
- Inicialmente E' está vazio
- Adicionar arestas de E em ordem crescente
- Se aresta gerar um ciclo em T, então descarte-a e continue

Algoritmo de *Kruskal*

Algoritmo de *Kruskal*

- **Idéia:** construir T adicionado arestas de menor peso
- Exemplo





Analizando o Algoritmo

- Algoritmos de *Prim* e *Kruskal* produzem sempre uma MST?
- Mas isto é óbvio?
- Como provar que algoritmo sempre produz resultado desejado – uma MST

- Duas propriedades de uma MST
  - Propriedade do ciclo (*cycle property*)
  - Propriedade do corte (*cut property*)

Propriedade do Ciclo

- Para qualquer ciclo C do grafo, se o peso de uma aresta *e* do ciclo for maior do que de todas as outras arestas do ciclo, então *e* não pertence a MST

Prova por contradição

- Assuma que aresta *e* pertence a MST, T
- Mostrar que existe outra árvore geradora T' com custo menor que não utiliza *e*
- Concluir que aresta *e* não pode pertencer a MST