

Algoritmo de Prim

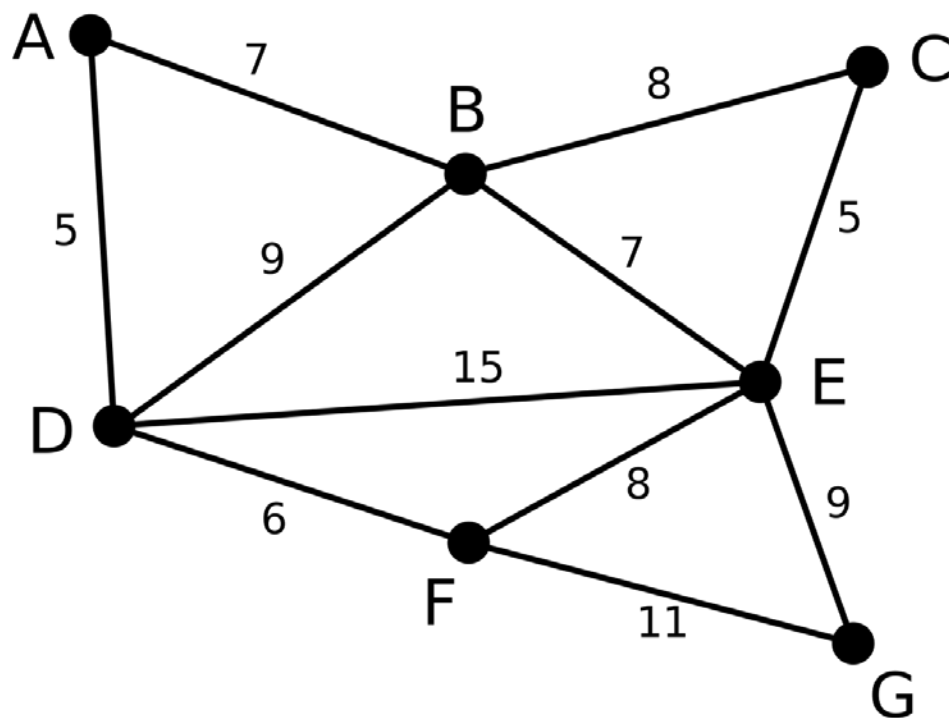
Estructuras de Datos

Grado en Ingeniería Informática, del
Software y de Computadores

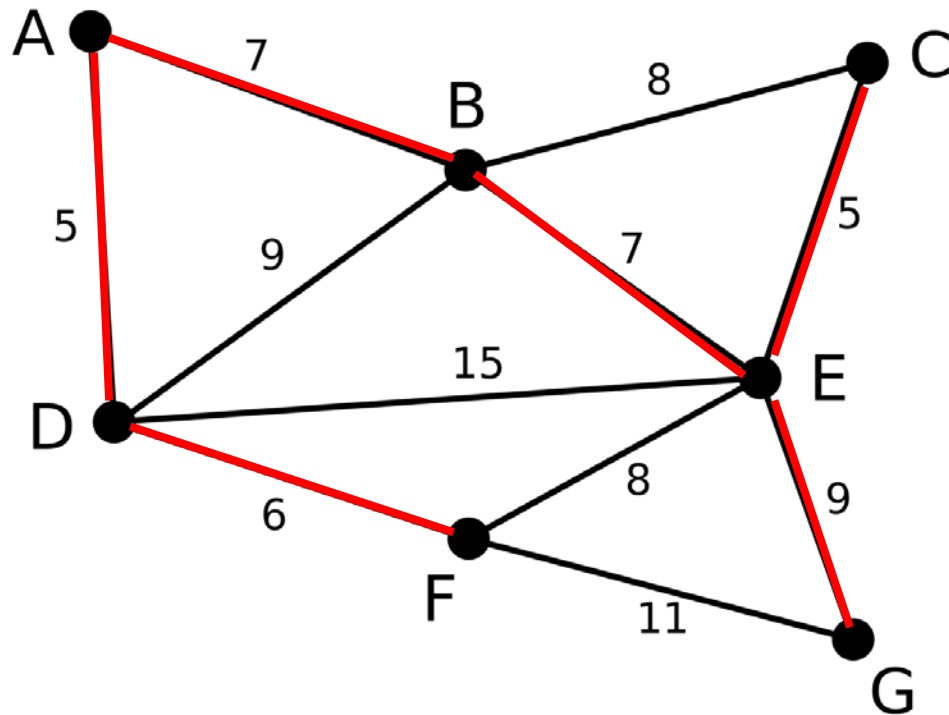
Objetivo del algoritmo de Prim

- Dado un grafo no orientado, conexo y con pesos, encontrar un árbol de expansión de peso mínimo
- El peso del árbol de expansión es la suma de los pesos de las aristas que lo componen
- Se trata de un algoritmo voraz: en cada paso elige el mínimo local

Dado el siguiente grafo



Obtenemos el siguiente árbol



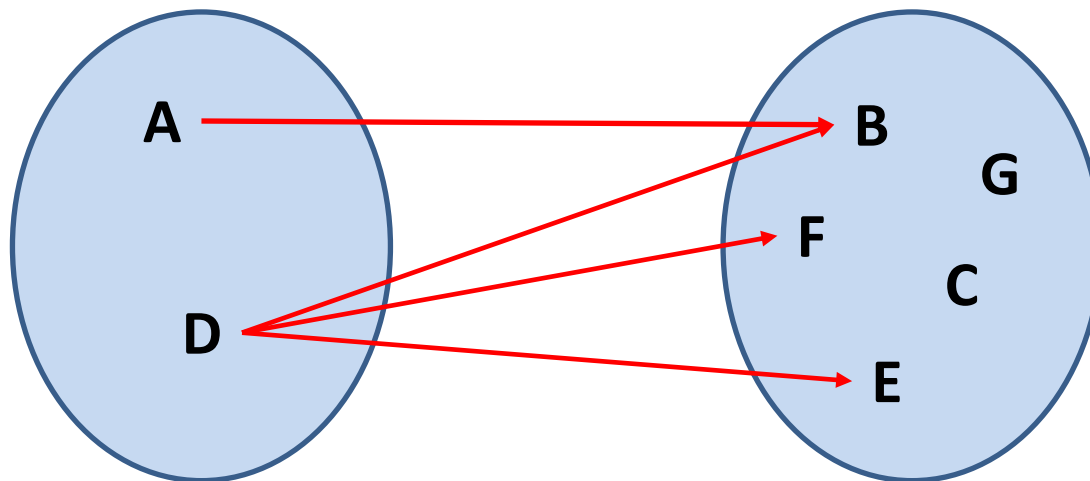
Estructuras de datos del algoritmo

El algoritmo maneja tres estructuras:

- VS = conjunto de nodos visitados
- RS = conjunto de nodos por visitar
- ST = árbol de expansión con nodos en VS

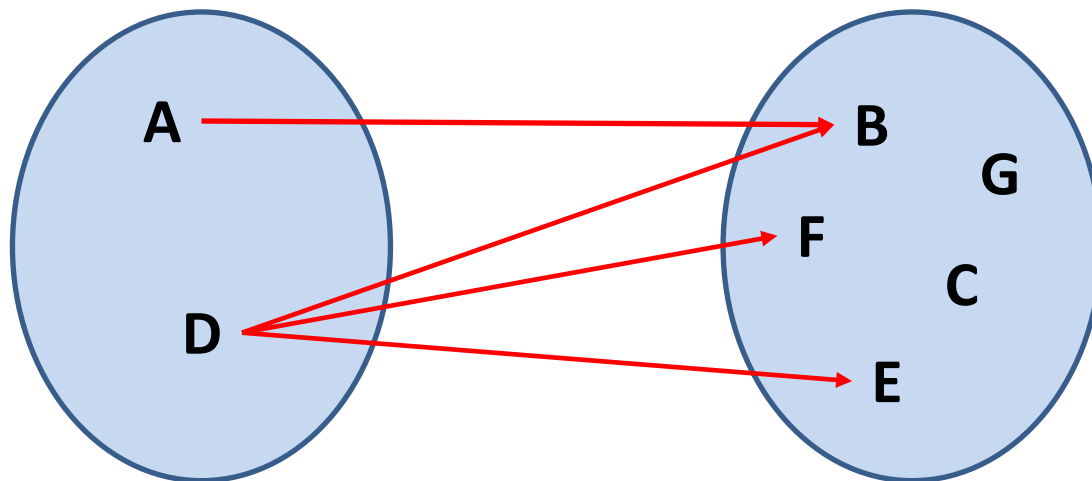
Corte y cruce de un grafo

- **Corte:** partición de sus vértices en dos conjuntos disjuntos
- **Cruce:** arista que une vértices de conjuntos distintos de un corte



Prim, cortes y cruces

- VS y RS forman un corte
- En cada paso se determinan los cruces (v, r) de VS a RS y se selecciona el mínimo
- El vértice r pasa de RS a VS



El algoritmo de Prim

Dados $G=(V, E)$ y v_0 un vértice cualquiera de G

1. Inicializar $VS = \{v_0\}$, $RS = V \setminus \{v_0\}$, $ST = \{\}$

2. Mientras RS no sea vacío

Sea (v, w, r) una arista de menor coste w de un vértice v de VS a un vértice r de RS (cruce)

$VS = VS \cup \{r\}$

$RS = RS \setminus \{r\}$

$ST = ST \cup \{(v, w, r)\}$

3. Devolver ST

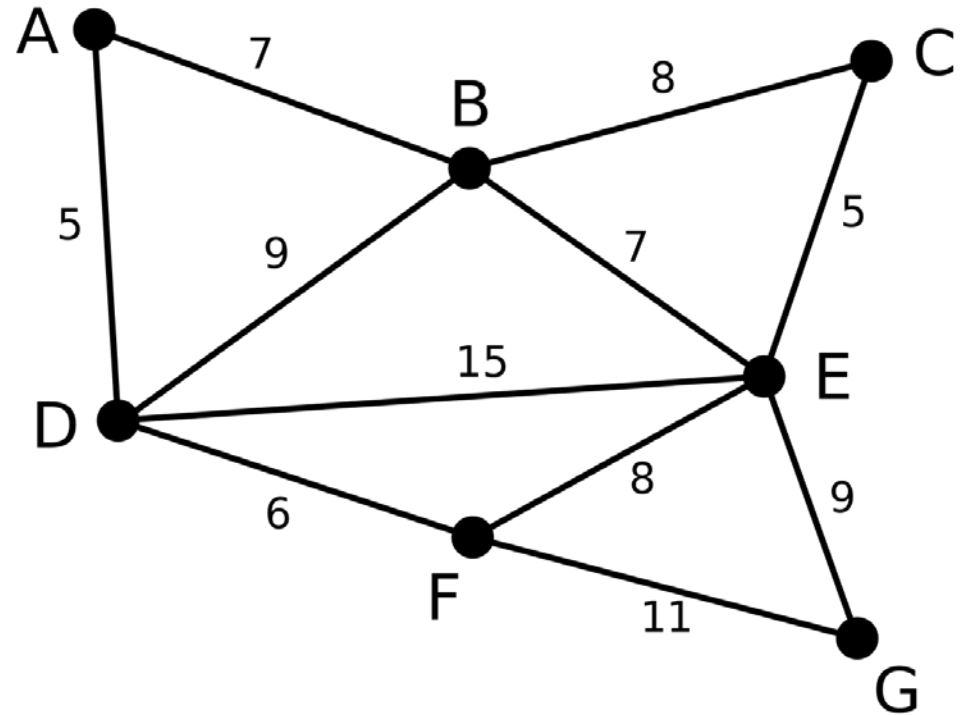
Inicialización

- Elegir inicial: D

$VS = \{D\}$

$RS = \{A, B, C, E, F, G\}$

$ST = \{\}$



Añadir la arista más ligera de VS a RS

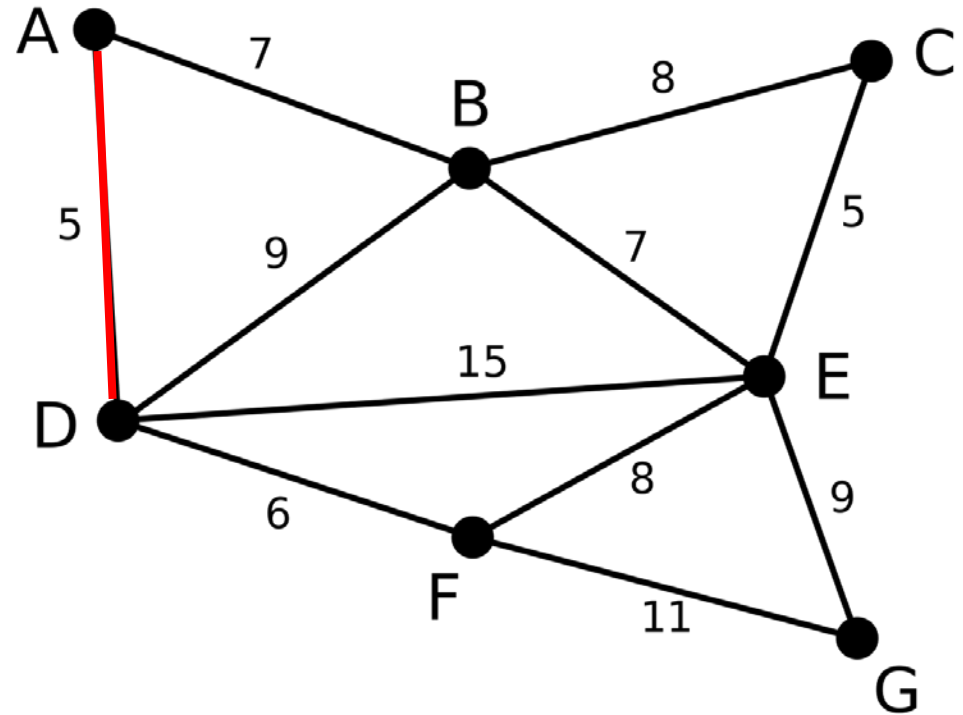
VS = {D}

RS = {A, B, C, E, F, G}

ST = {}

Aristas de VS a RS:

- D 5 A
- D 9 B
- D 15 E
- D 6 F



Añadir la arista más ligera de VS a RS

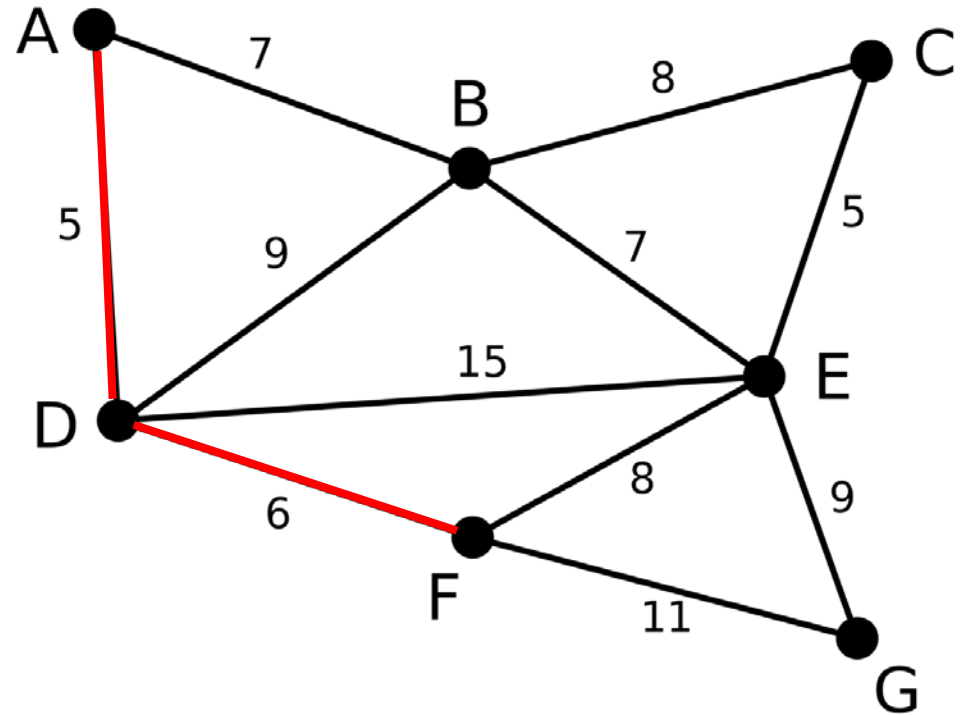
VS = {D, A}

RS = {B, C, E, F, G}

ST = {(D, A)}

Aristas de VS a RS:

- D 9 B
- D 15 E
- **D 6 F**
- A 7 B



Añadir la arista más ligera de VS a RS

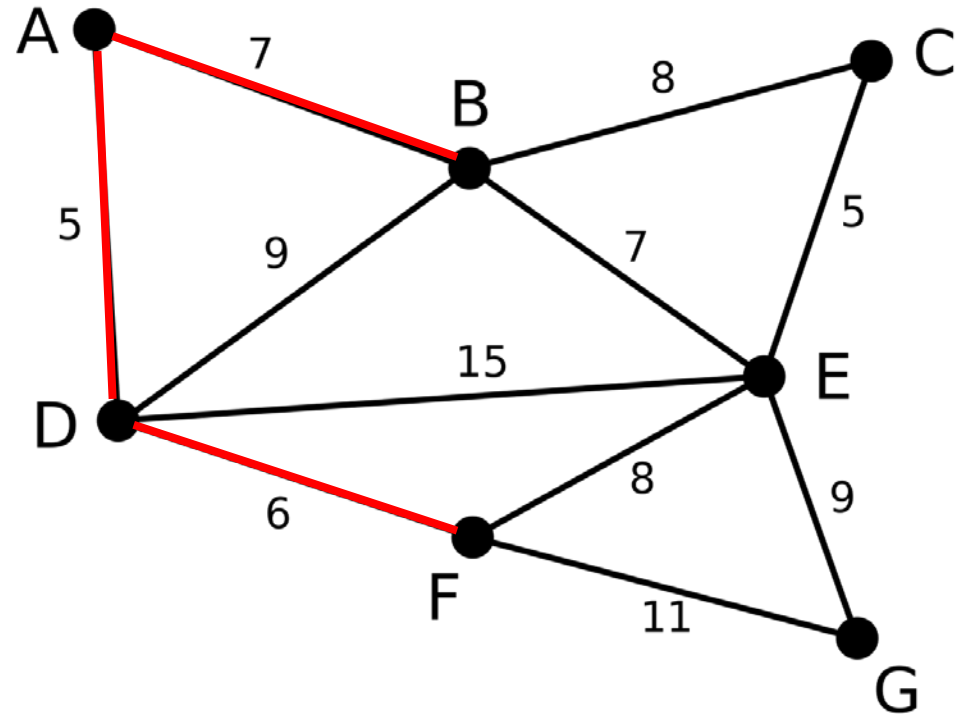
VS = {D, A, F}

RS = {B, C, E, G}

ST = {(D, A), (D, F)}

Aristas de VS a RS:

- D 9 B
- D 15 E
- **A 7 B**
- F 8 E
- F 11 G



Añadir la arista más ligera de VS a RS

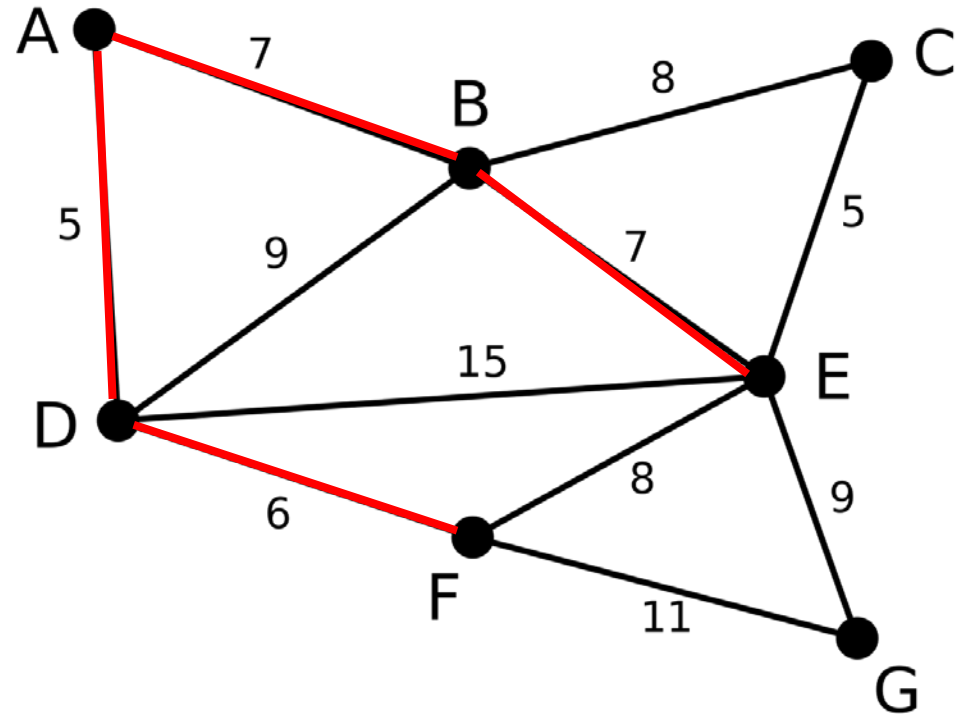
VS = {D, A, F, B}

RS = {C, E, G}

ST = {(D, A), (D, F), (A, B)}

Aristas de VS a RS:

- D 15 E
- F 8 E
- F 11 G
- B 8 C
- **B 7 E**



Añadir la arista más ligera de VS a RS

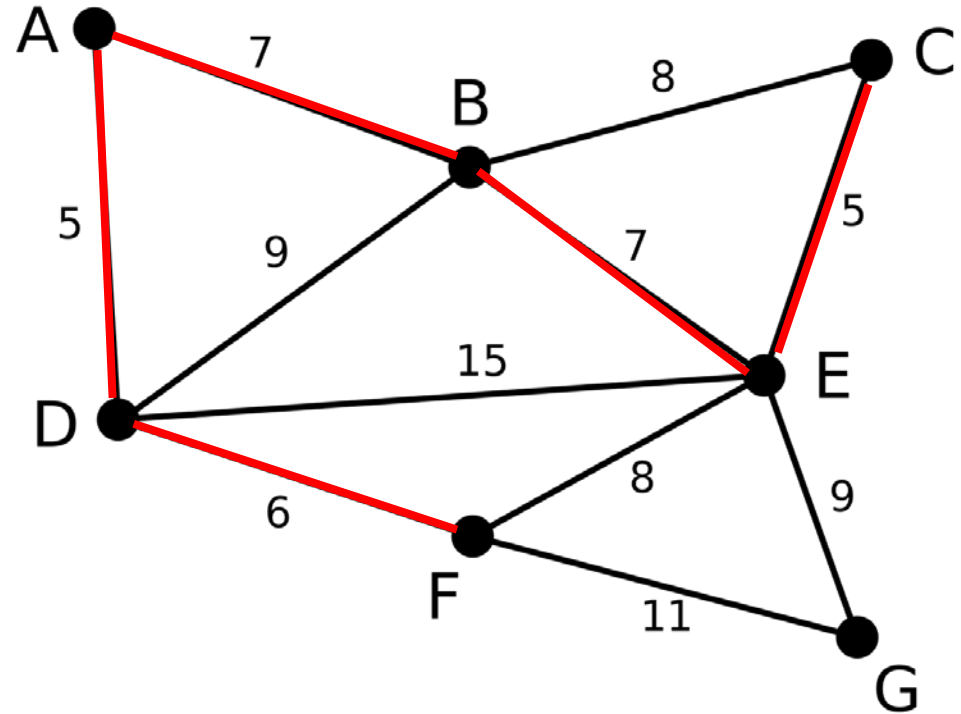
VS = {D, A, F, B, E}

RS = {C, G}

ST = {(D, A), (D, F), (A, B),
(B, E)}

Aristas de VS a RS:

- F 11 G
- B 8 C
- E 5 C
- E 9 G



Añadir la arista más ligera de VS a RS

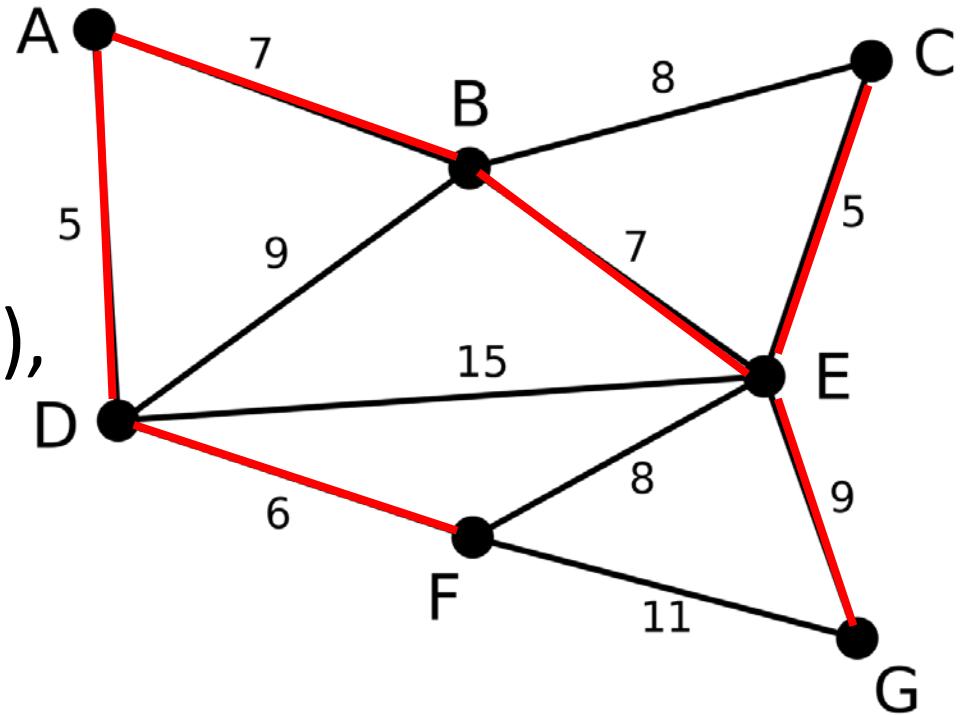
VS = {D, A, F, B, E, C}

RS = {G}

ST = {(D, A), (D, F), (A, B),
(B, E), (E, C)}

Aristas de VS a RS:

- F 11 G
- E 9 G

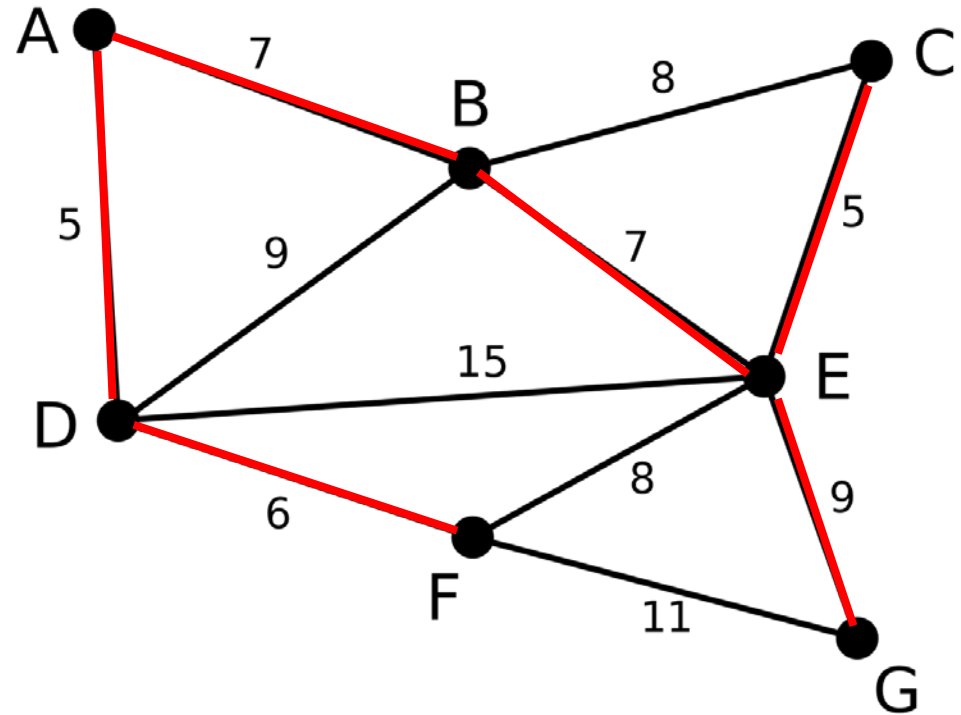


RS vacío, VS y ST completos

VS = {D, A, F, B, E, C, G}

RS = {}

ST = {(D,A), (D,F), (A,B),
(B,E), (E,C), (E,G)}



Grafos con peso en Haskell

Tipo **WeightedGraph** *a w*

- *a* tipo de los vértices
- *w* tipo del peso de las aristas (instancia de **Ord**)

Dos constructores:

- **mkWeightedGraphAdj**
- **mkWeightedGraphEdges**

Un grafo con peso en Haskell

```
data WeightedEdge a w = WE a w a
```

```
g1 :: WeightedGraph Char Int
```

```
g1 = mkWeightedGraphEdges
```

```
    ['a', 'b', 'c', 'd', 'e']
```

```
    [ WE 'a' 3 'b', WE 'a' 7 'd'
```

```
    , WE 'b' 4 'c', WE 'b' 2 'd'
```

```
    , WE 'c' 5 'd', WE 'c' 6 'e'
```

```
    , WE 'd' 5 'e'
```

```
    ]
```

Sucesores de un nodo

successors :: WeightedGraph a w -> a -> [(a, w)]

> successors g1 'a'
[('b',3),('d',7)]

> successors g1 'e'
[('c',6),('d',5)]

El tipo `Ordering`

```
data Ordering = LT | EQ | GT
```

```
compare :: Ord a => a -> a -> Ordering
```

```
> compare 5 5
```

```
EQ
```

```
> compare 5 6
```

```
LT
```

```
> compare 6 5
```

```
GT
```

La función **comparaArcos**

Compara arcos teniendo en cuenta solo pesos:

```
> comparaArcos (WE 'a' 3 'c') (WE 'b' 1 'd')  
GT
```

```
> comparaArcos (WE 'a' 3 'c') (WE 'b' 3 'd')  
EQ
```

La función `minimumBy`

```
minimumBy :: Ord a => (a -> a -> Ordering) -> [a] -> a
```

```
> minimumBy comparaArcos [WE 'a' 3 'b',  
                           WE 'a' 2 'c',  
                           WE 'a' 7 'd']
```

```
WE 'a' 2 'c'
```

```
> minimumBy comparaArcos [WE 'a' 3 'b',  
                           WE 'x' 2 'y']
```

```
WE 'x' 2 'y'
```