CURSO DE PROGRAMACIÓN COMPETITIVA URJC - 2023

Sesión 5 (8ª Semana)

- Isaac Lozano (isaac.lozano@urjc.es)
- Raúl Martín (raul.martin@urjc.es)
- **Sergio Salazar** (s.salazarc.2018@alumnos.urjc.es)
- Francisco Tórtola (f.tortola.2018@alumnos.urjc.es)
- Cristian Pérez (c.perezc.2018@alumnos.urjc.es)
- Xuqiang Liu (x.liu1.2020@alumnos.urjc.es)
- Alicia Pina (a.pinaz.2020@alumnos.urjc.es)
- Sara García (s.garciarod.2020@alumnos.urjc.es)
- Raúl Fauste (r.fauste.2020@alumnos.urjc.es)

El problema del viajante (de Tinder)

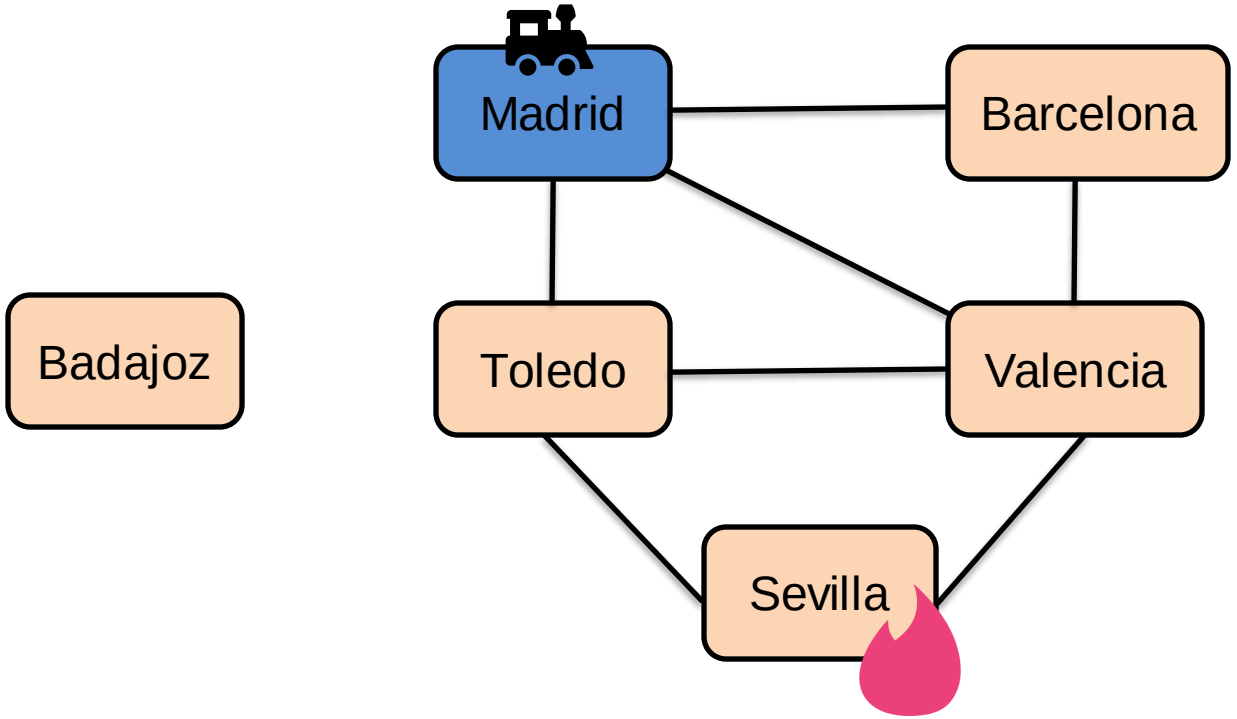
Has hecho match con la pareja de tus sueños. Lo tiene todo, es inteligente, guapa y encima sabe programación competitiva...

Solo tiene un problema...

VIVE EN SEVILLA

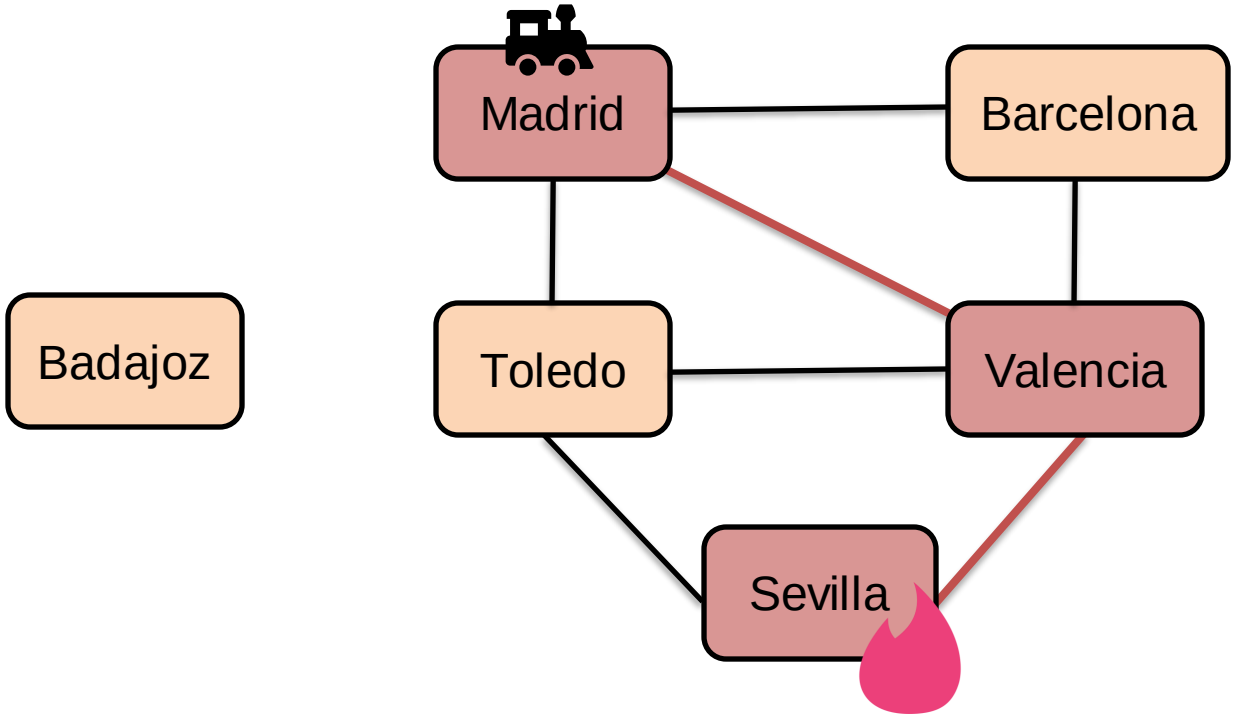
El problema del viajante (de Tinder)

¿Cómo llegamos a Sevilla?



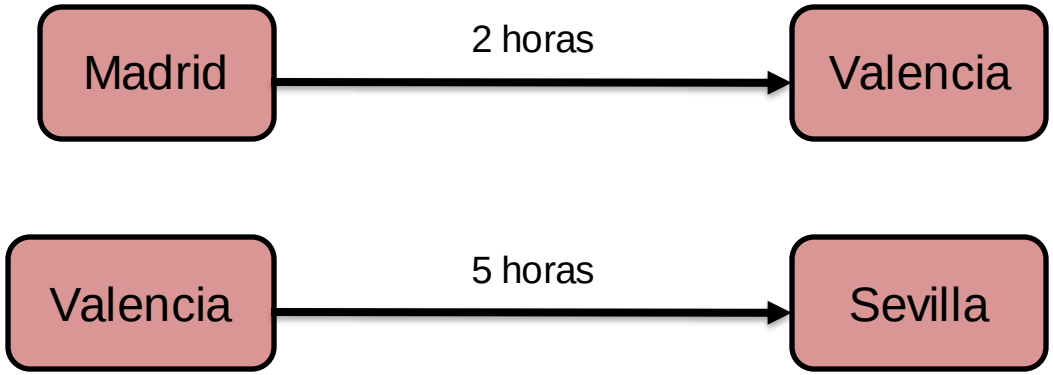
El problema del viajante (de Tinder)

Podemos pasar por Valencia y haremos un único transbordo



El problema del viajante (de Tinder)

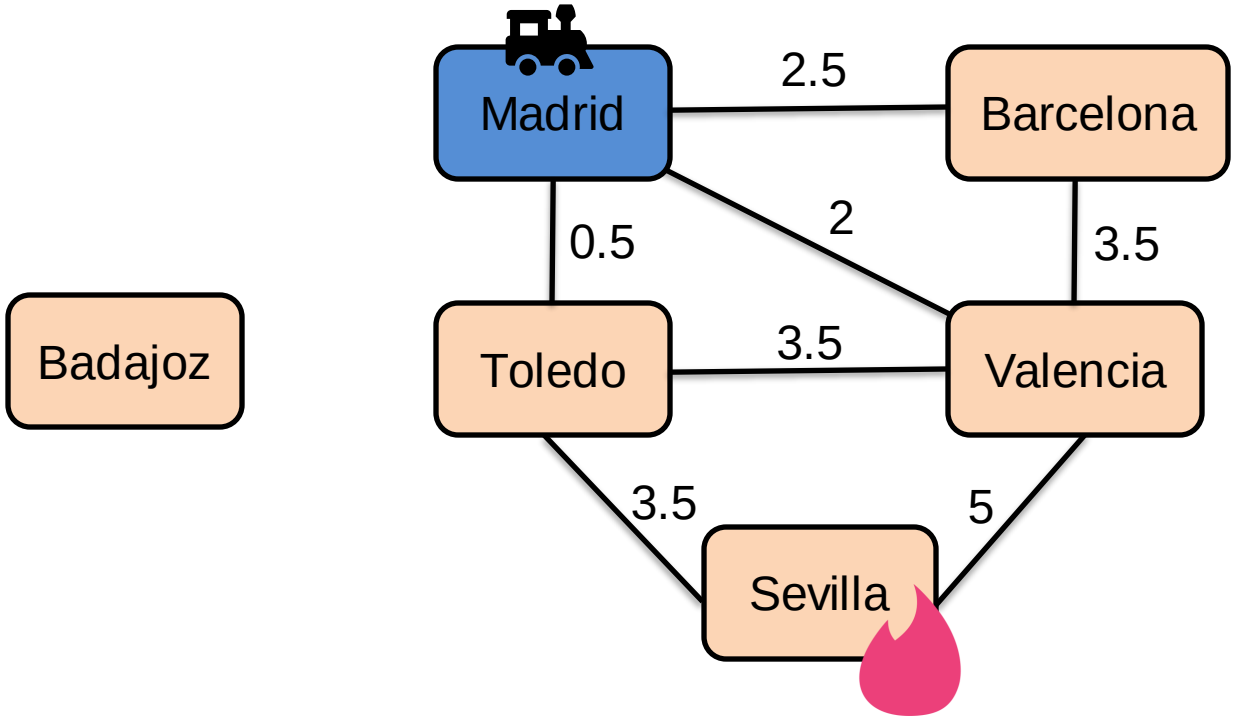
Podemos pasar por Valencia y haremos un único transbordo



7 horas ????

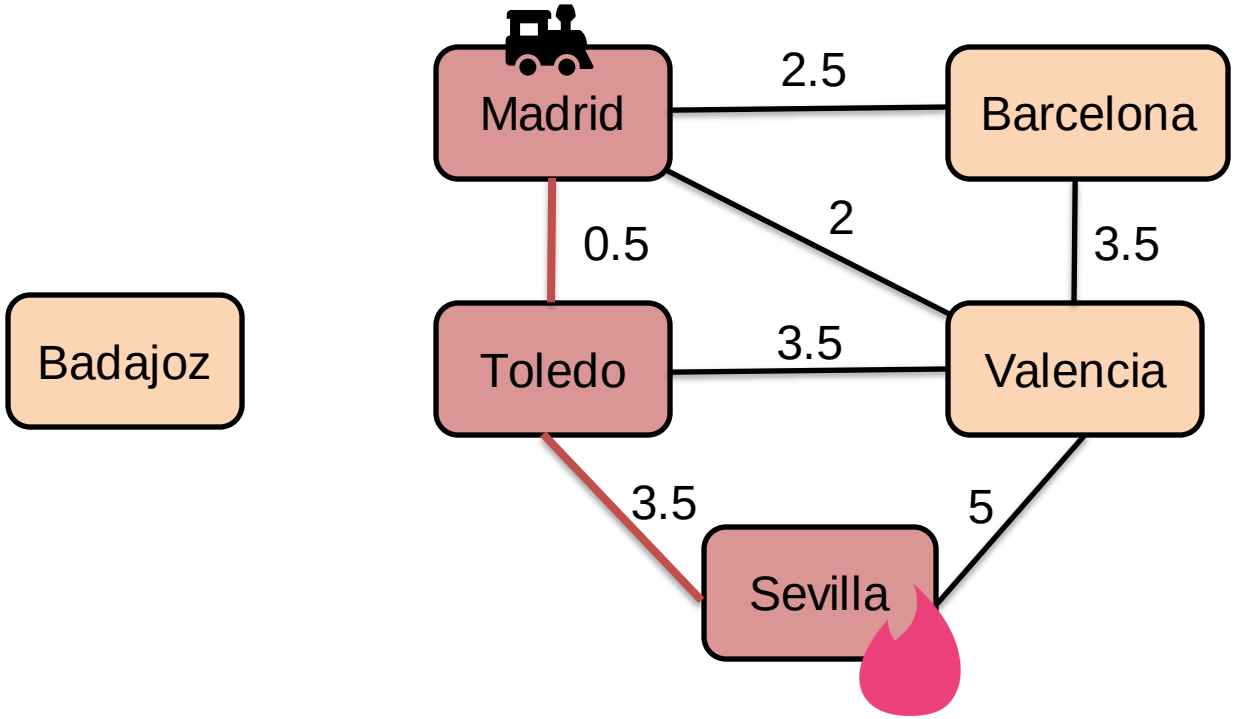
El problema del viajante (de Tinder)

¿Y ahora?



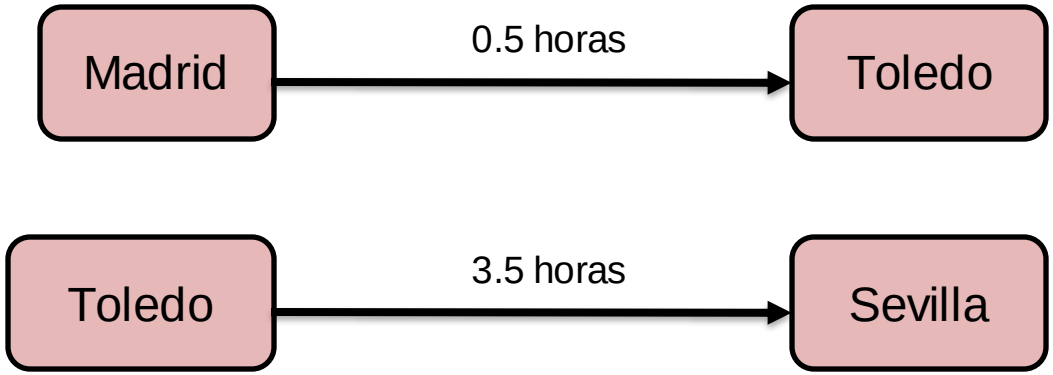
El problema del viajante (de Tinder)

Podemos pasar por Toledo



El problema del viajante (de Tinder)

Podemos pasar por Toledo

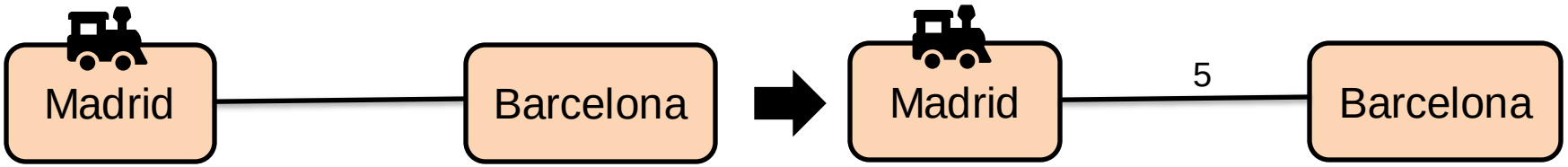


4 horas

Grafos Ponderados

A veces no todas las aristas son igual de importantes!!!

Introducimos PESOS en ellas.



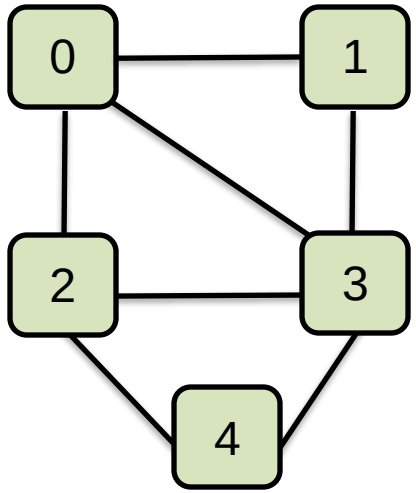
Contenidos

- Representación de grafos ponderados
- Algoritmos de camino más corto
- Algoritmos de Árboles de Recubrimiento

Representación

MATRIZ DE ADYACENCIA

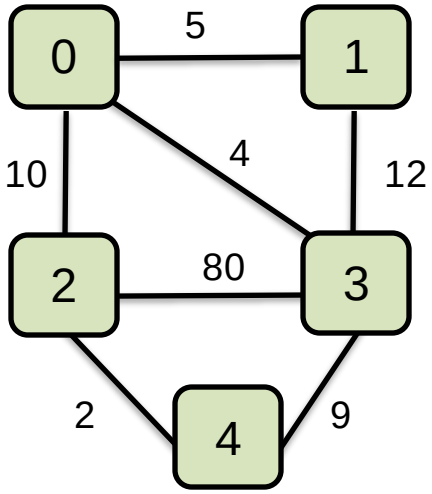
X	0	1	2	3	4
0	1	1	1	1	0
1	1	1	0	1	0
2	1	0	1	1	1
3	1	1	1	1	1
4	0	0	1	1	1



Representación

MATRIZ DE ADYACENCIA

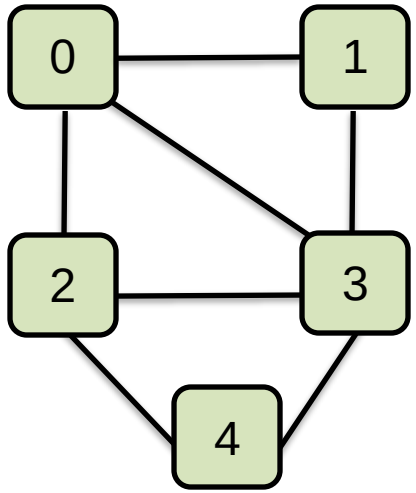
X	0	1	2	3	4
0	0?	5	10	4	∞
1	5	0?	∞	12	∞
2	10	∞	0?	80	2
3	4	12	80	0?	9
4	∞	∞	2	9	0?



Representación

LISTA DE ARISTAS

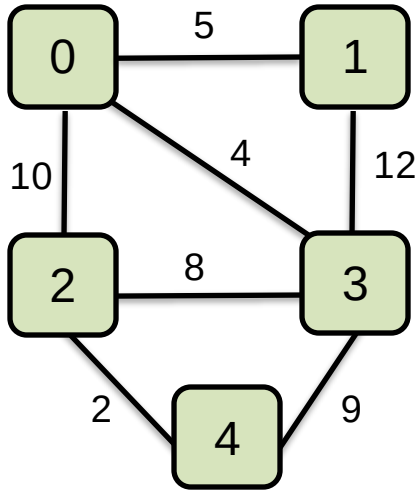
(0,1)
(0,2)
(0,3)
(1,3)
(2,3)
(2,4)
(3,4)



Representación

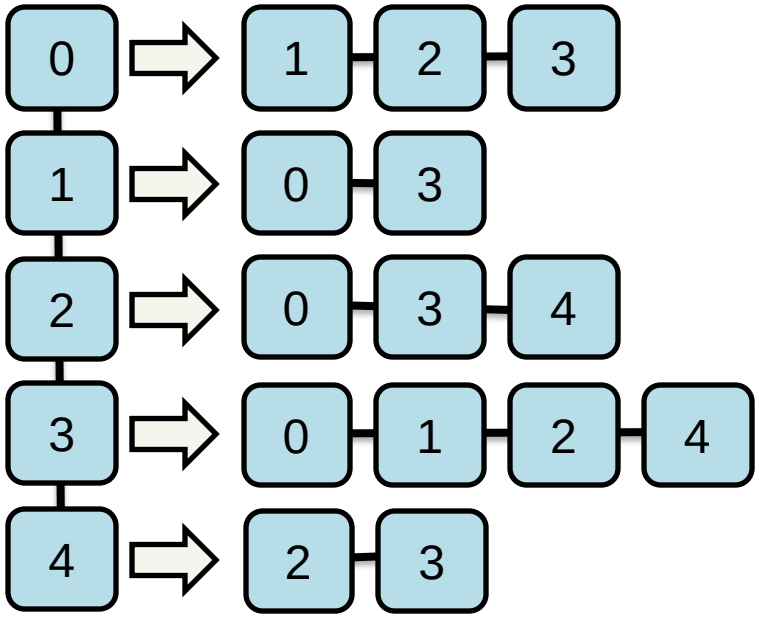
LISTA DE ARISTAS

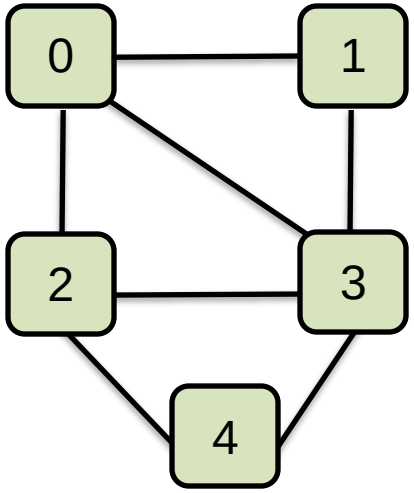
(0,1,**5**)
(0,2,**10**)
(0,3,**4**)
(1,3,**12**)
(2,3,**80**)
(2,4,**2**)
(3,4,**9**)



Representación

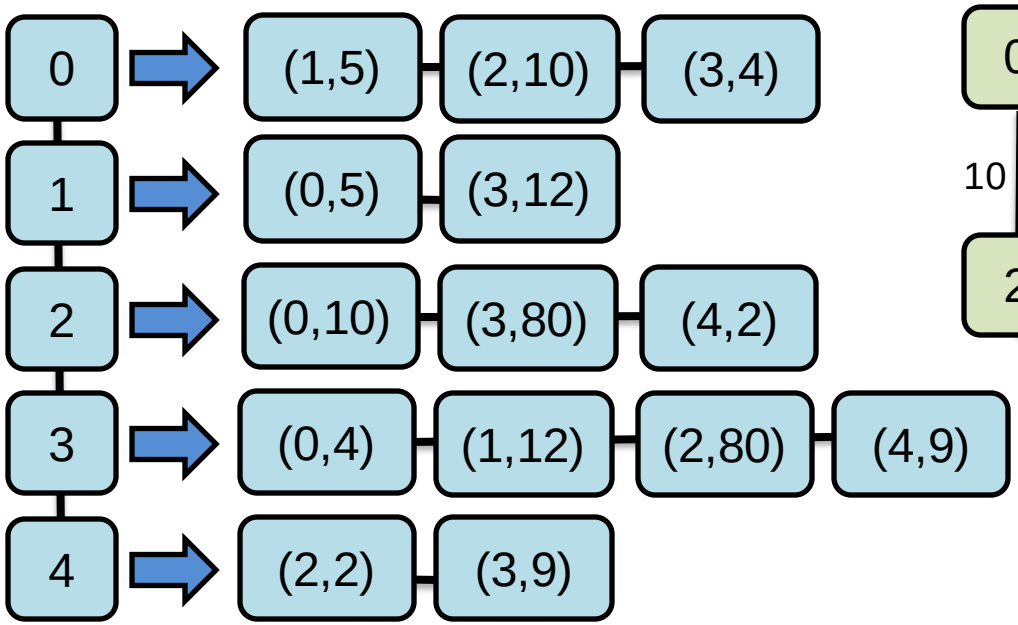
LISTA DE ADYACENCIAS

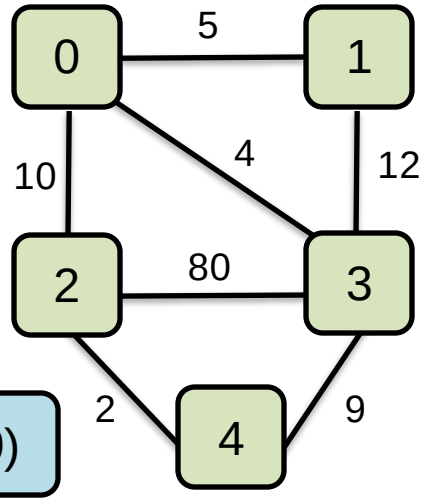




Representación

LISTA DE ADYACENCIAS





Representación

LISTA DE ADYACENCIAS

```
class IntPar{
    int nodo, distancia
        IntPar(n,d):
            nodo = n
            distancia = d
}

Arraylist<IntPar>[] = grafo[N]
```

Problemas y Algoritmos

Algoritmos de Caminos más Cortos

Problema 1: ¿CUÁL ES EL CAMINO MÁS CORTO DEL NODO i AL NODO j?

Idea: Problema del viajante (de Tinder)

Algoritmos de Caminos más Cortos

ALGORITMO DE DIJKSTRA

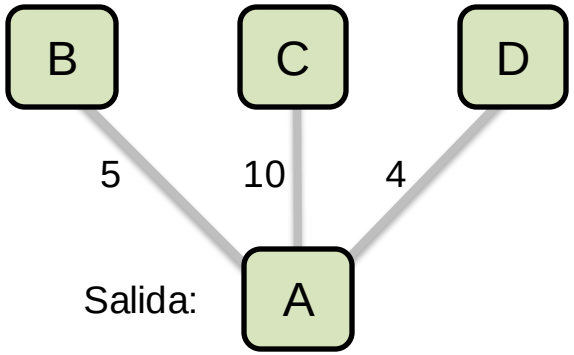
- Algoritmo Voraz

IDEA: Si siempre escojo el camino más corto, cuando llegue al final habré llegado en el camino más corto

Algoritmo de Dijkstra

Objetivo: E

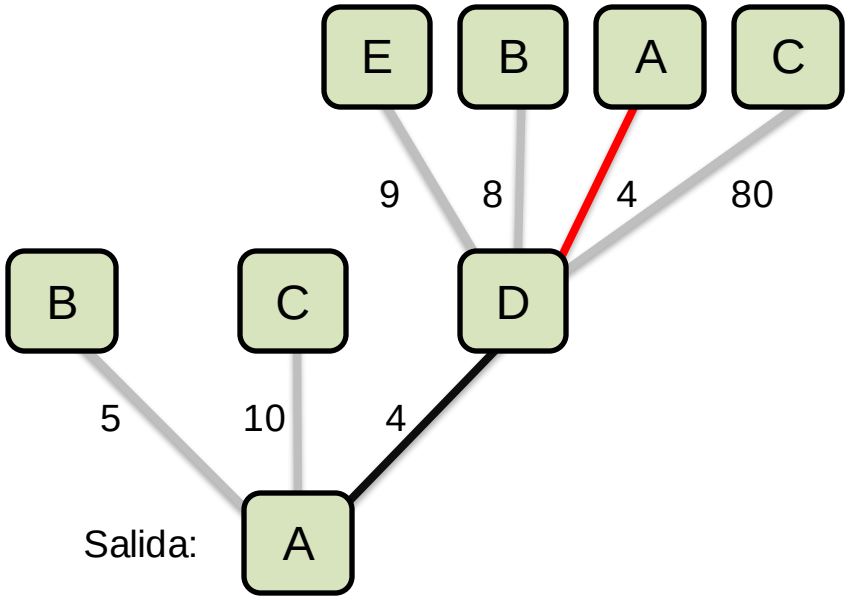
A	0
B	∞
C	∞
D	∞
E	∞



Algoritmo de Dijkstra

Objetivo: E

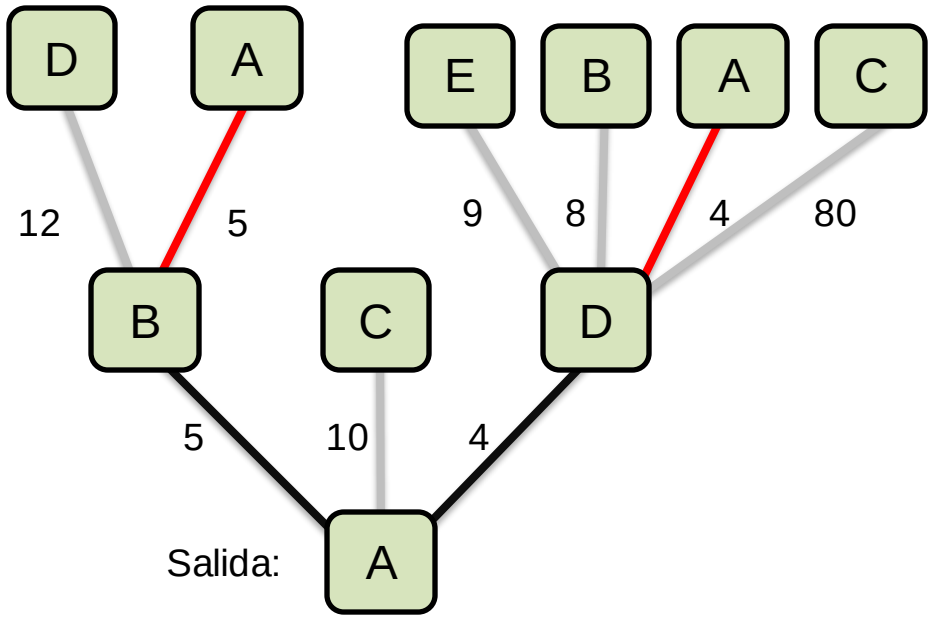
A	0
B	∞
C	∞
D	4
E	∞



Algoritmo de Dijkstra

Objetivo: E

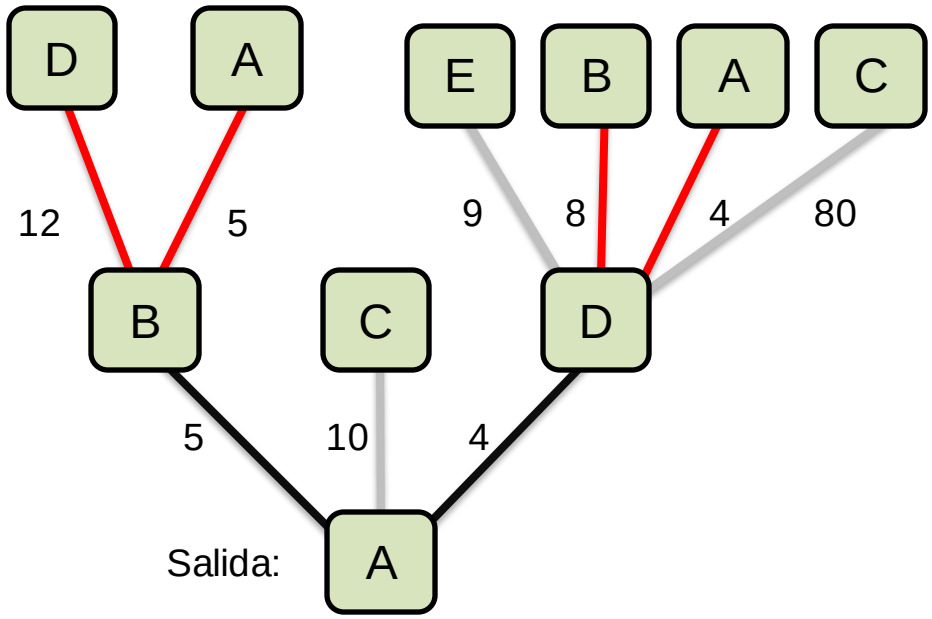
A	0
B	5
C	∞
D	4
E	∞



Algoritmo de Dijkstra

Objetivo: E

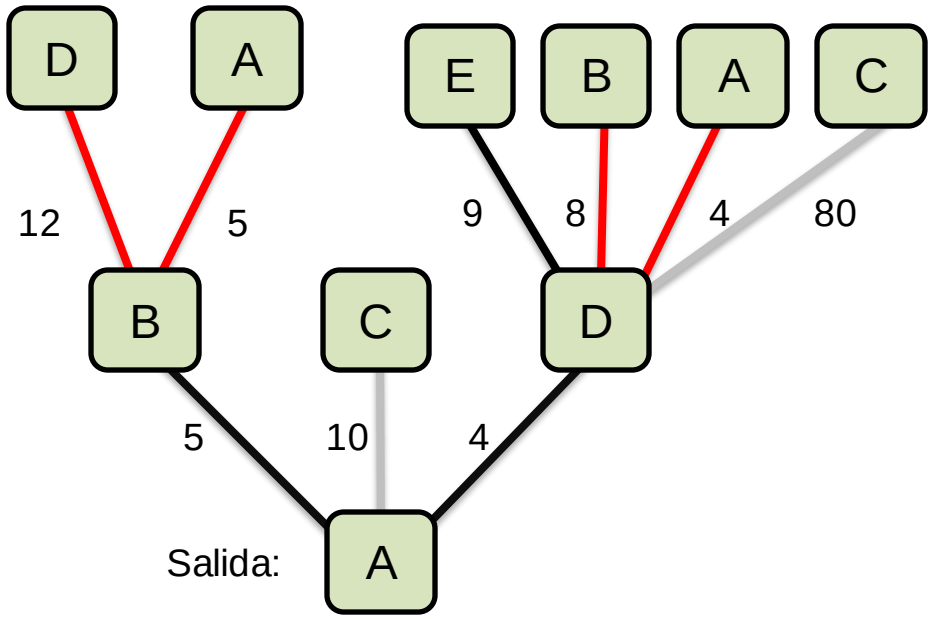
A	0
B	5
C	∞
D	4
E	∞



Algoritmo de Dijkstra

Objetivo: E

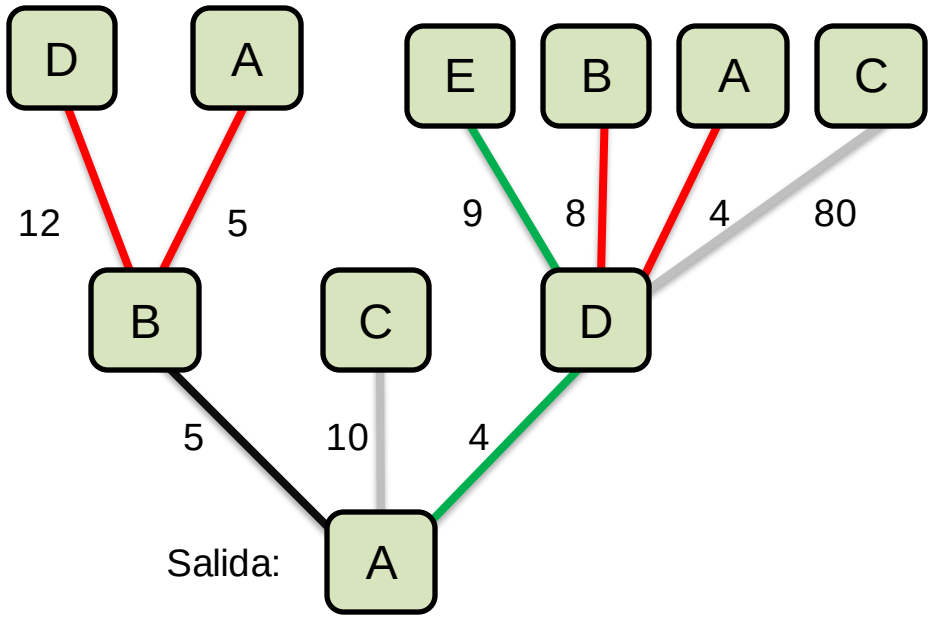
A	0
B	5
C	∞
D	4
E	13



Algoritmo de Dijkstra

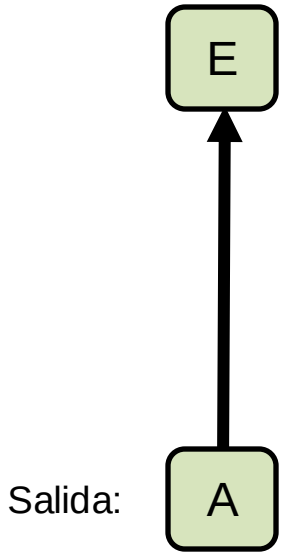
Objetivo: E

A	0
B	5
C	∞
D	4
E	13



Algoritmo de Dijkstra

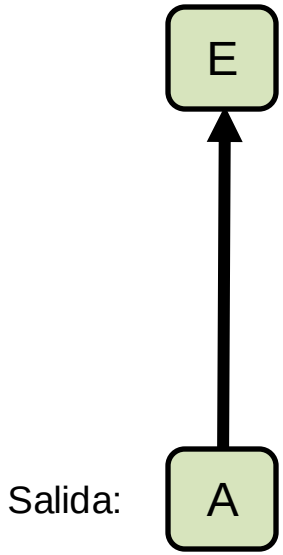
A	0
B	5
C	∞
D	4
E	13



EL CAMINO MÁS CORTO DE A HASTA E ES 13

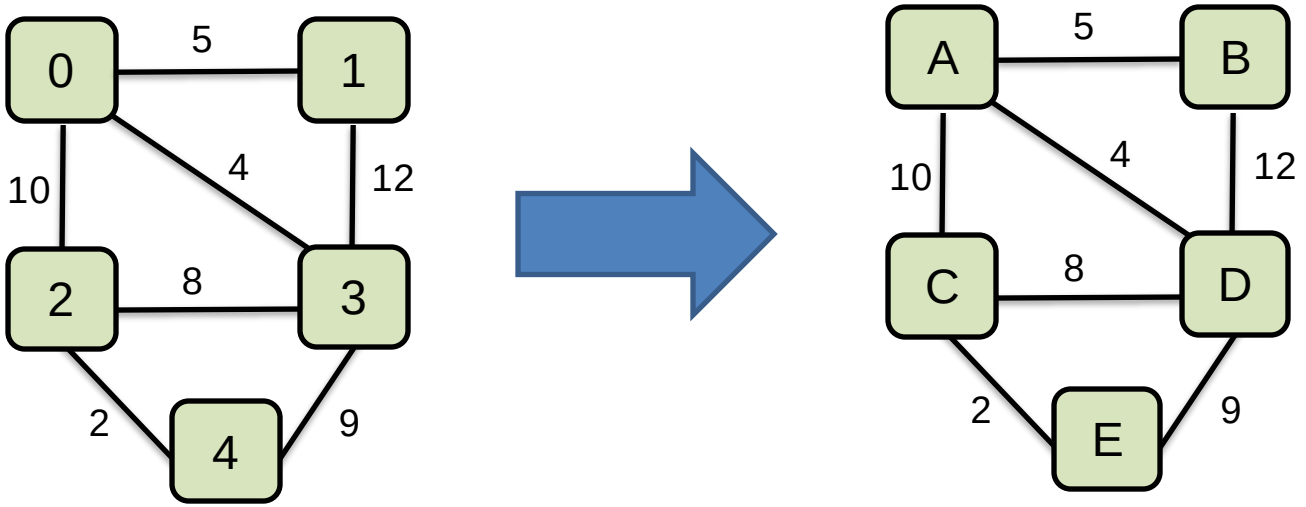
Algoritmo de Dijkstra

A	0
B	5
C	∞
D	4
E	13

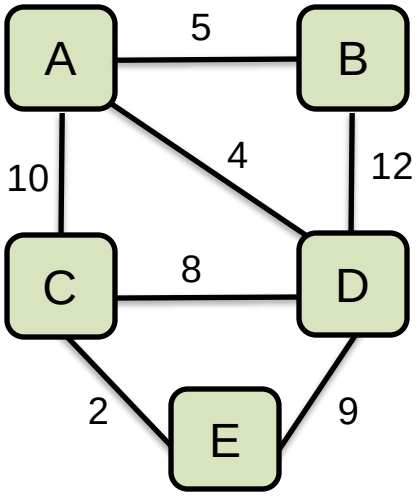


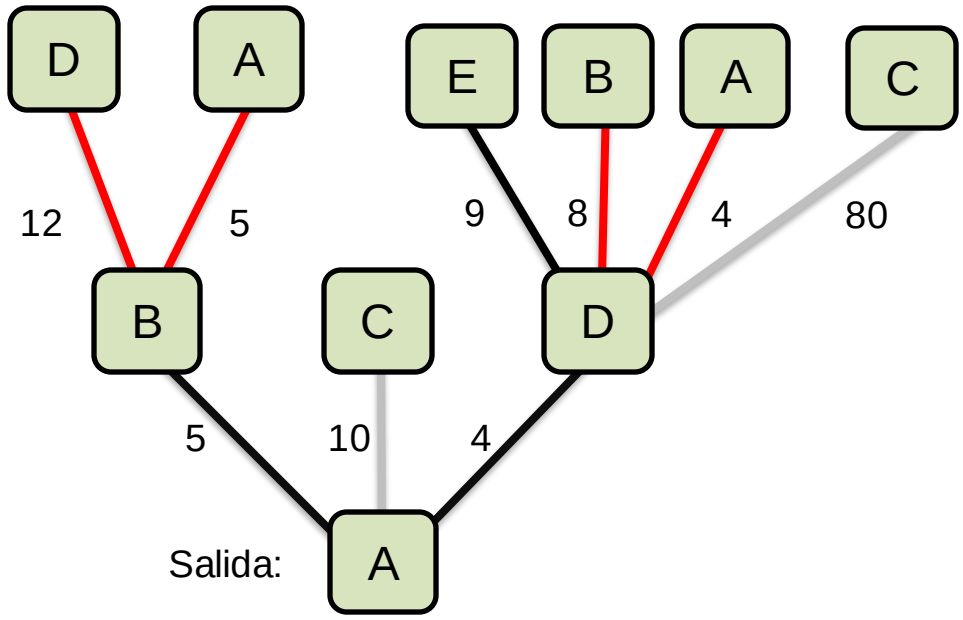
¿¿¿¿EL CAMINO MÁS CORTO DE A HASTA E ES 13????

Algoritmo de Dijkstra

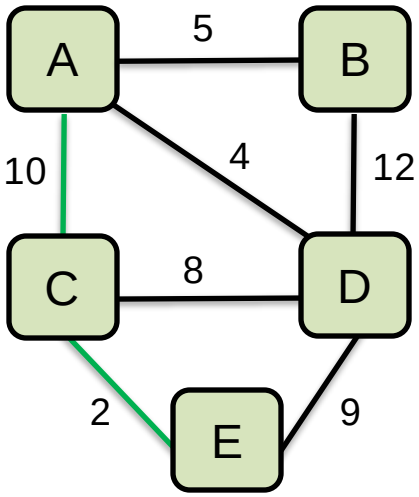


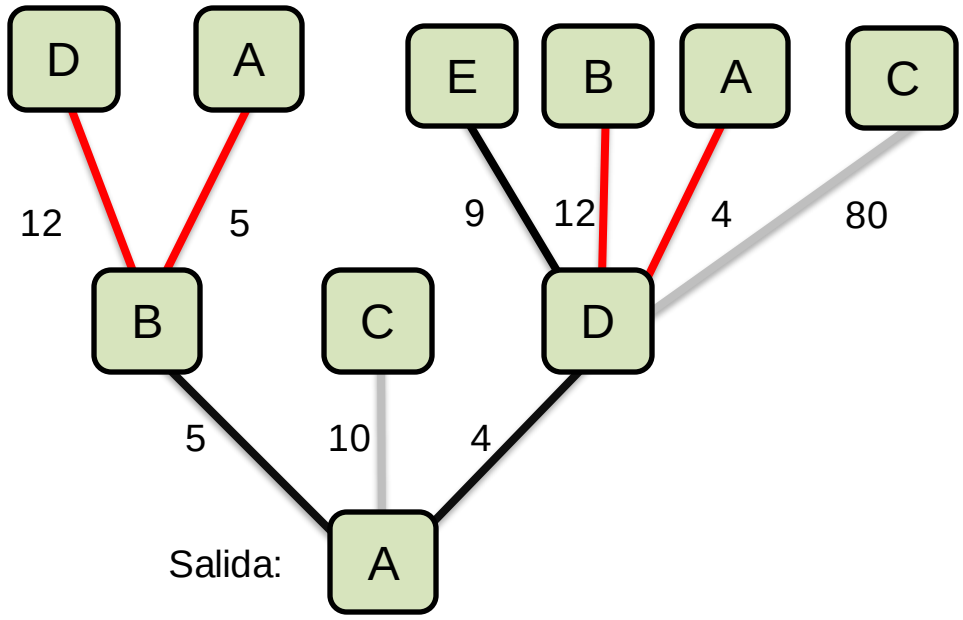
Algoritmo de Dijkstra





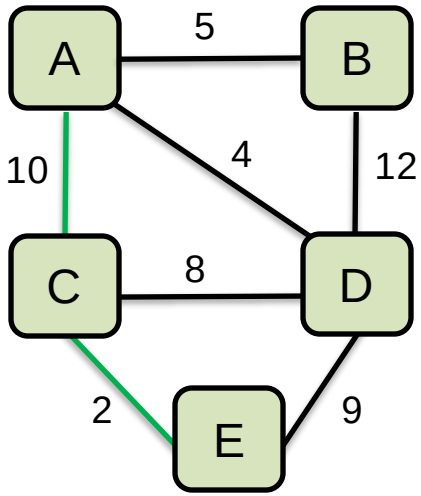
Algoritmo de Dijkstra

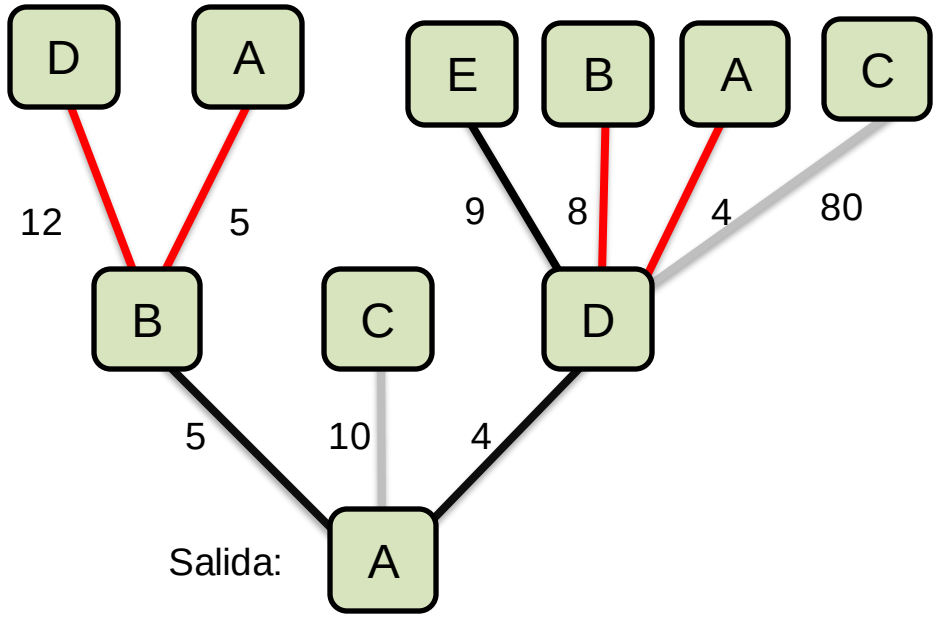




Algoritmo de Dijkstra

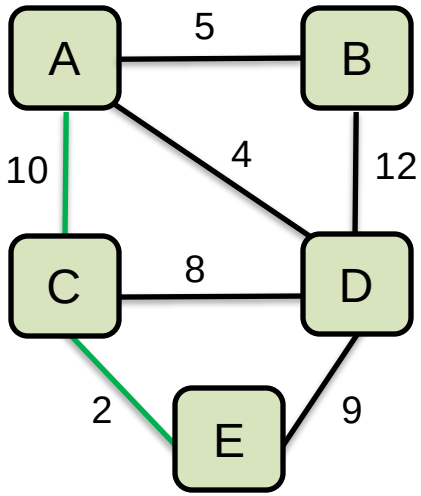
¿Dónde está el fallo?

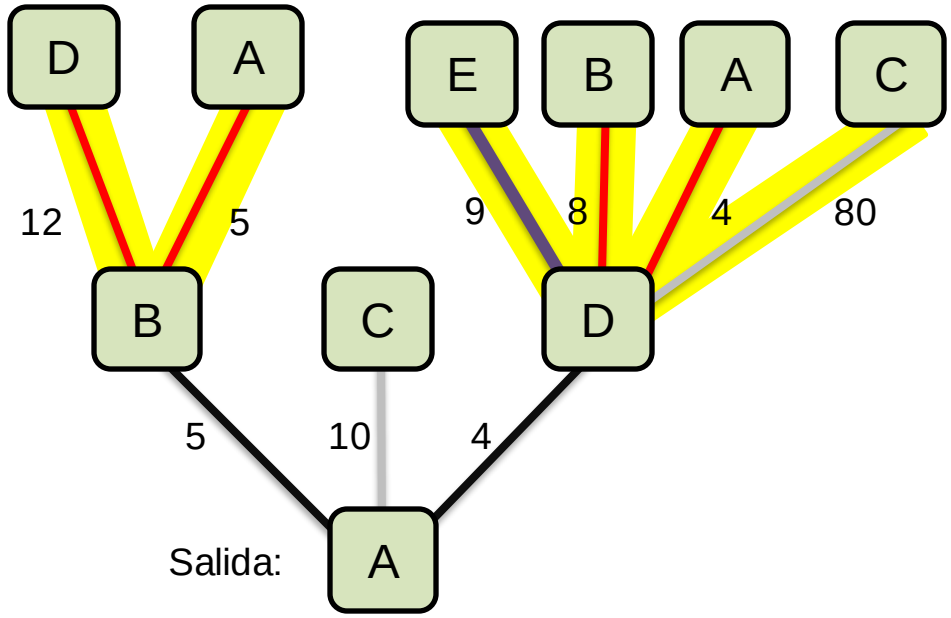




Algoritmo de Dijkstra

¿Dónde está el fallo?

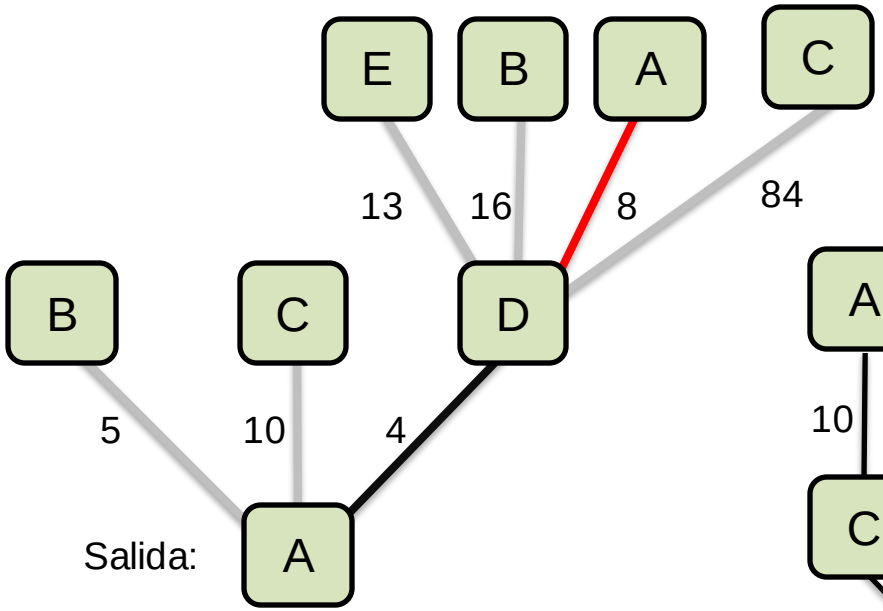


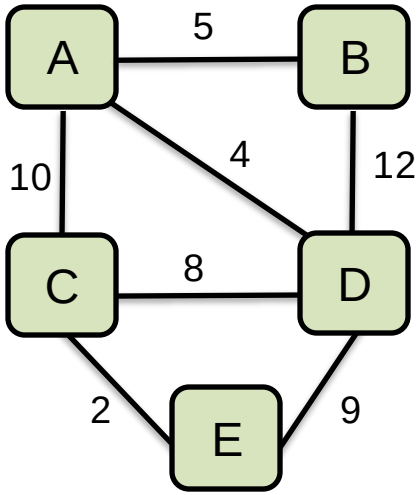


Algoritmo de Dijkstra

Objetivo: E

A	0
B	∞
C	∞
D	4
E	∞





Algoritmo de Dijkstra

¿Cómo escogemos el camino más corto de forma eficiente?

¡¡¡Iterar sobre todas las aristas disponibles en cada paso es muy costoso!!!

O(n) por iteración => **$O(n^2)$**

Algoritmo de Dijkstra

COLA DE PRIORIDAD

Guardaremos los posibles nuevos caminos con su distancia

¿¿¿Complejidad???

Algoritmo de Dijkstra

COLA DE PRIORIDAD

Guardaremos los posibles nuevos caminos con su distancia

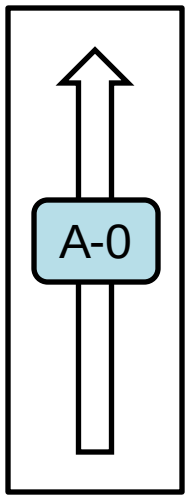
¿¿¿Complejidad???
Insertar y eliminar en una PQ es O(logn)
Por cada iteración **O(n logn)**

Algoritmo de Dijkstra

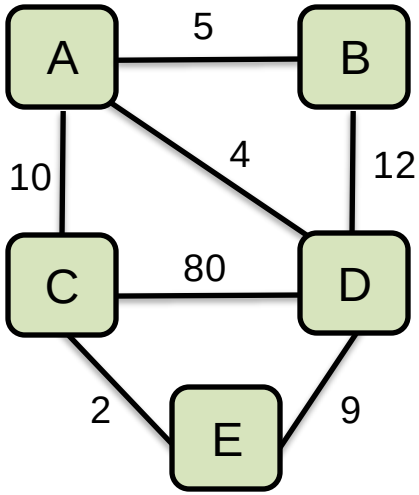
Objetivo: E

A	0
B	∞
C	∞
D	∞
E	∞

PQ



Salida: A

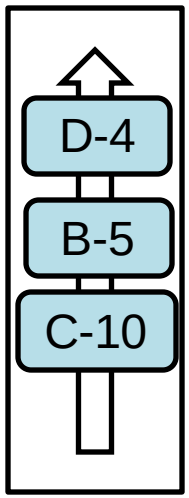


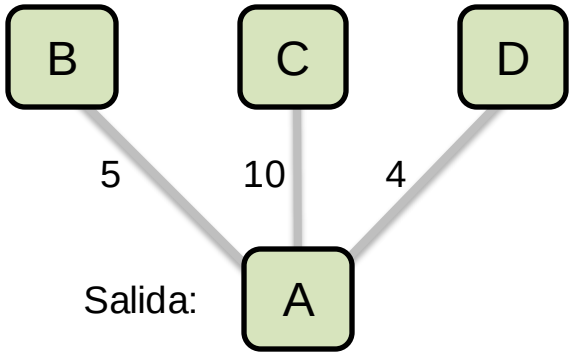
Algoritmo de Dijkstra

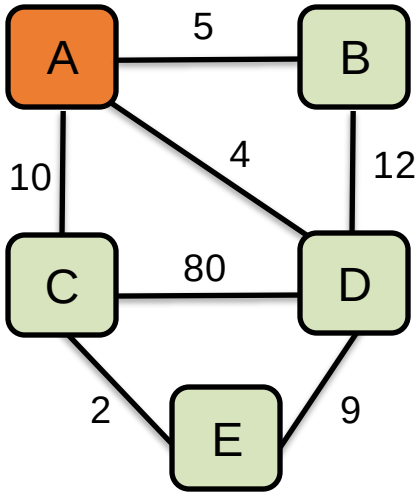
Objetivo: E

A	0
B	5
C	10
D	4
E	∞

PQ





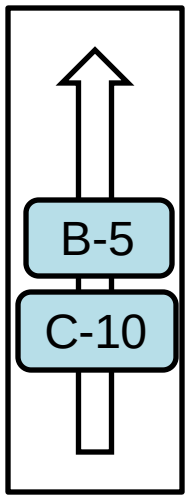


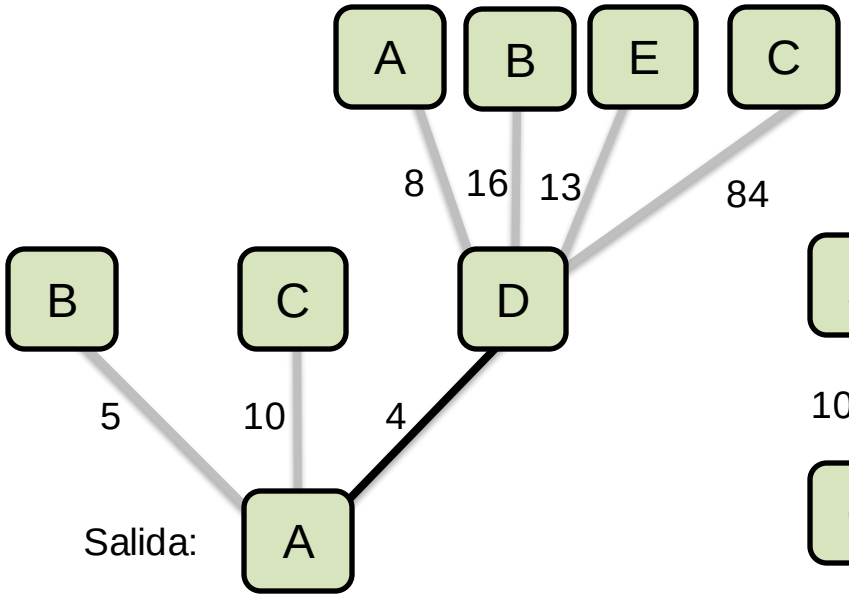
Algoritmo de Dijkstra

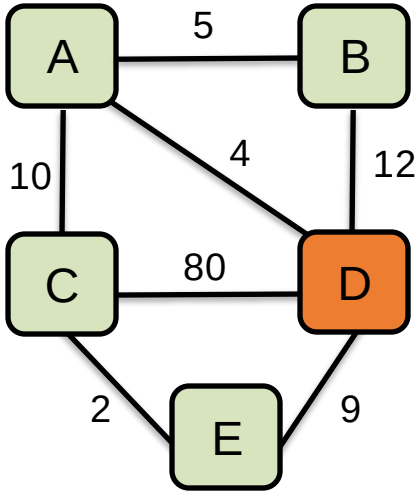
Objetivo: E

A	0
B	5
C	10
D	4
E	∞

PQ





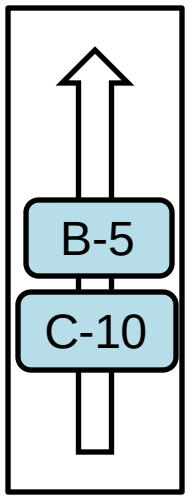


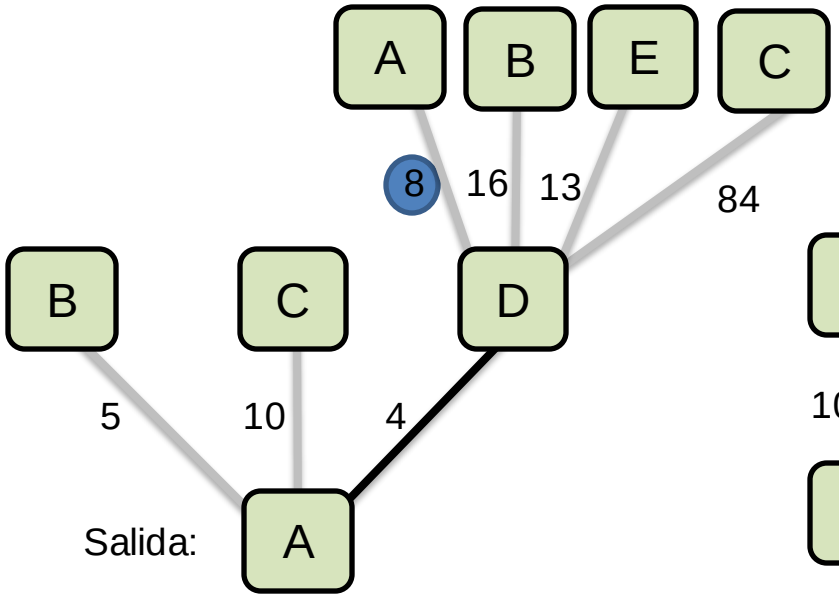
Algoritmo de Dijkstra

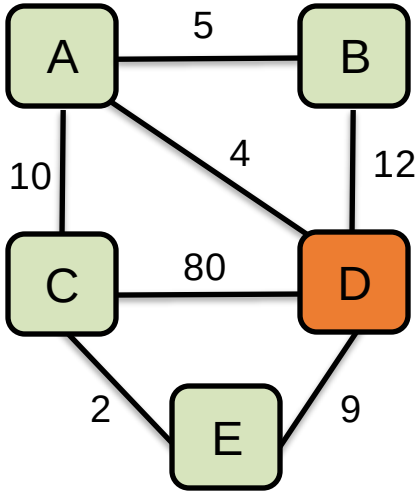
Objetivo: E

A	0
B	5
C	10
D	4
E	∞

PQ





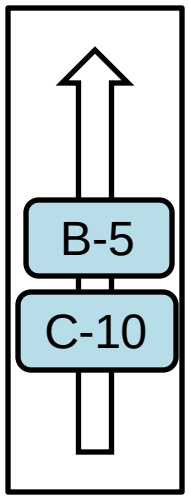


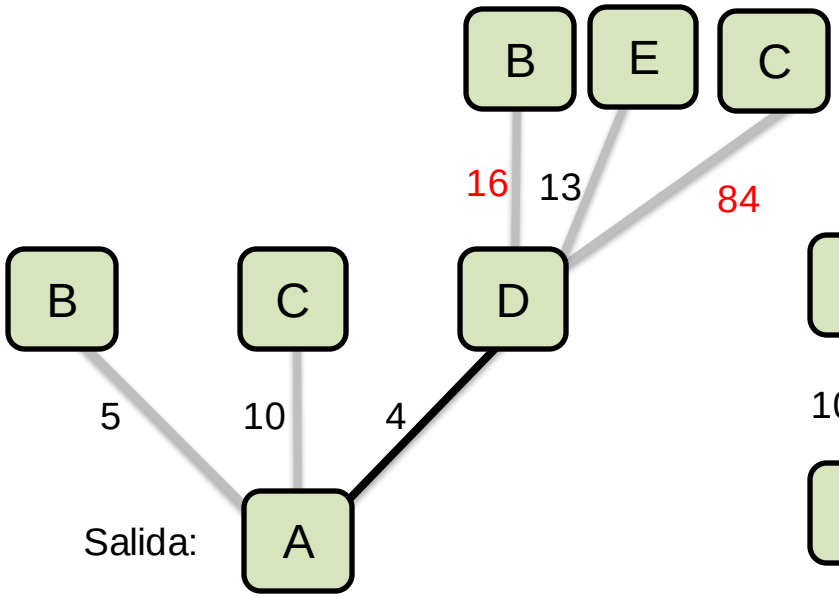
Algoritmo de Dijkstra

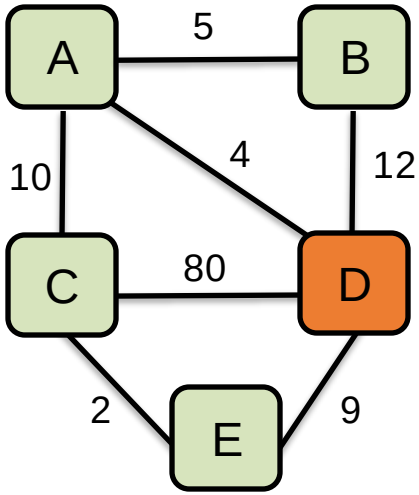
Objetivo: E

A	0
B	5
C	10
D	4
E	∞

PQ





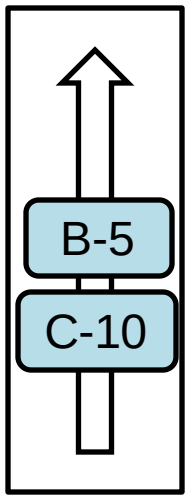


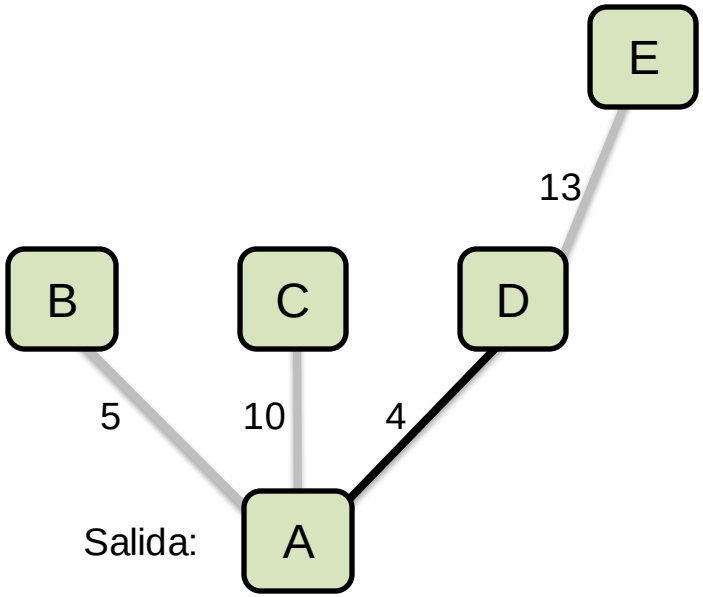
Algoritmo de Dijkstra

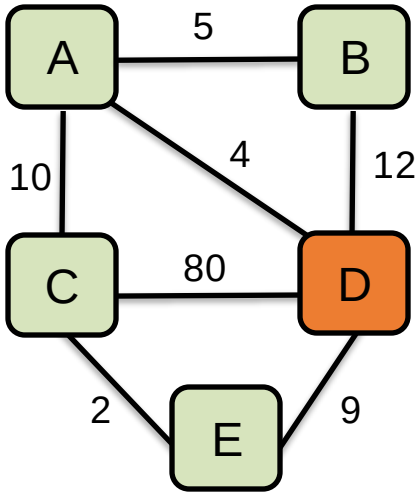
Objetivo: E

A	0
B	5
C	10
D	4
E	∞

PQ





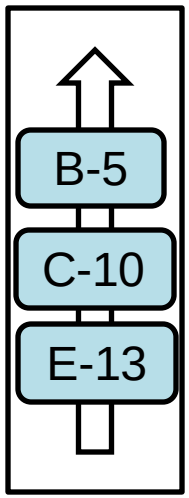


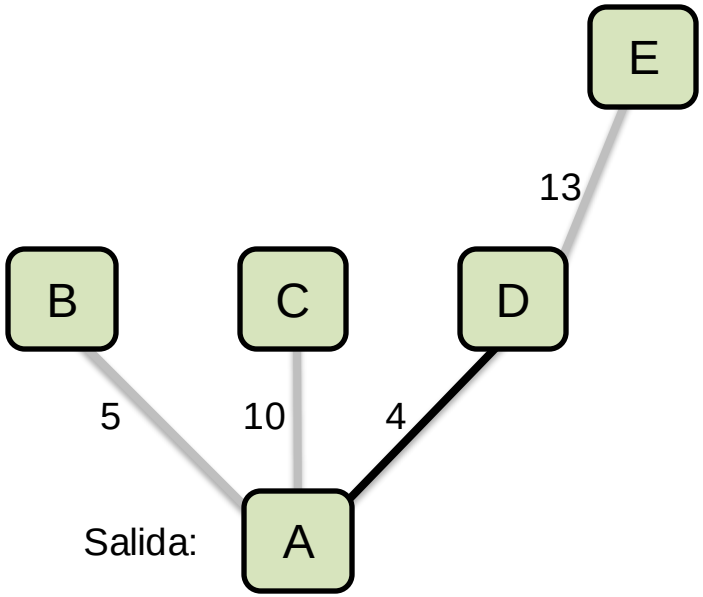
Algoritmo de Dijkstra

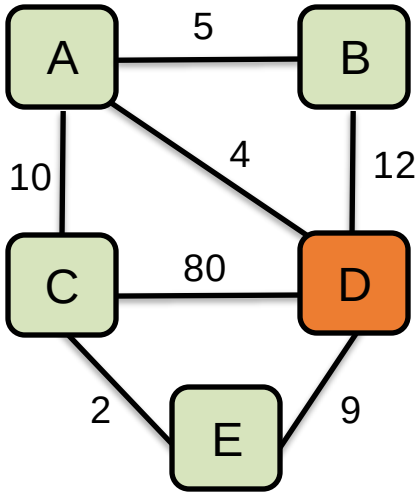
Objetivo: E

A	0
B	5
C	10
D	4
E	13

PQ



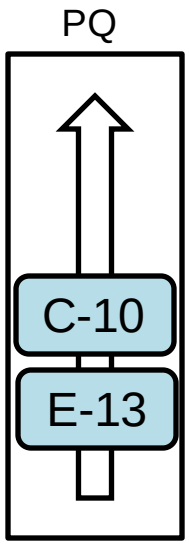


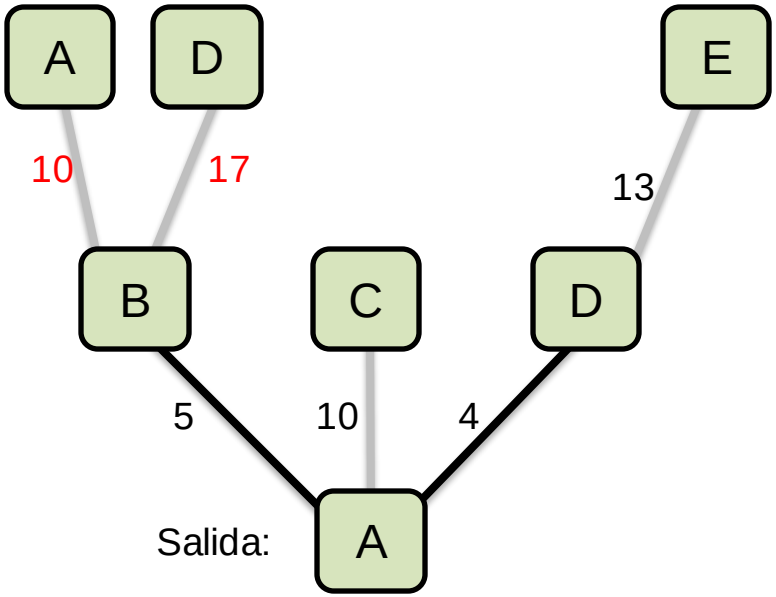


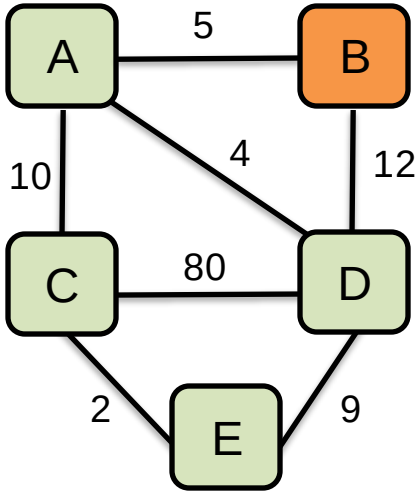
Algoritmo de Dijkstra

Objetivo: E

A	0
B	5
C	10
D	4
E	13





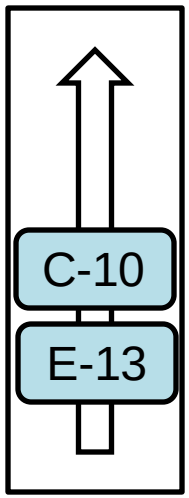


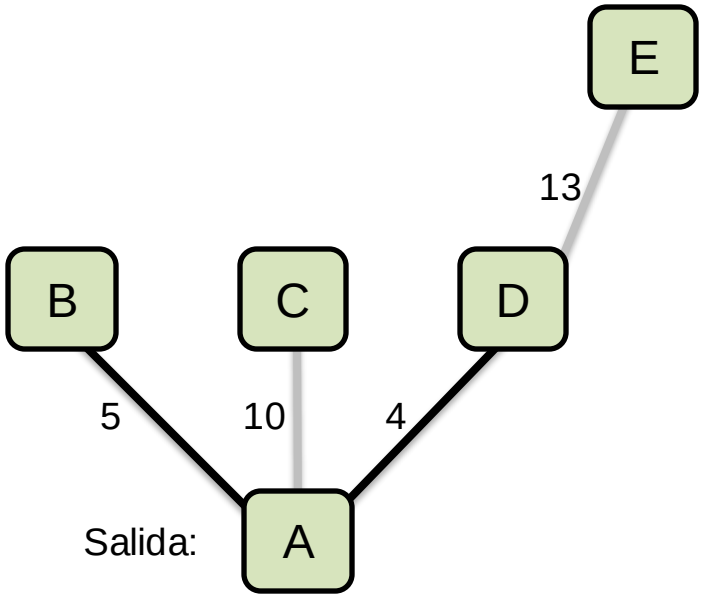
Algoritmo de Dijkstra

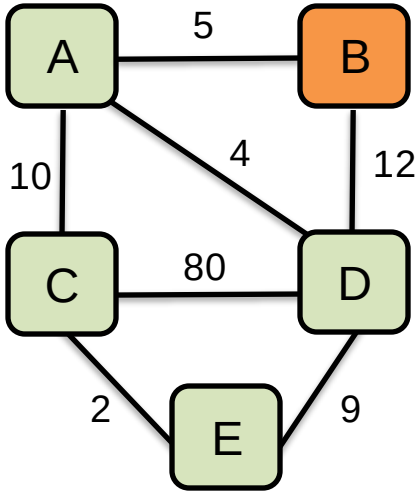
Objetivo: E

A	0
B	5
C	10
D	4
E	13

PQ





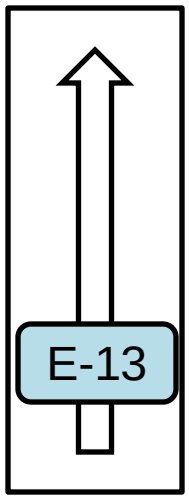


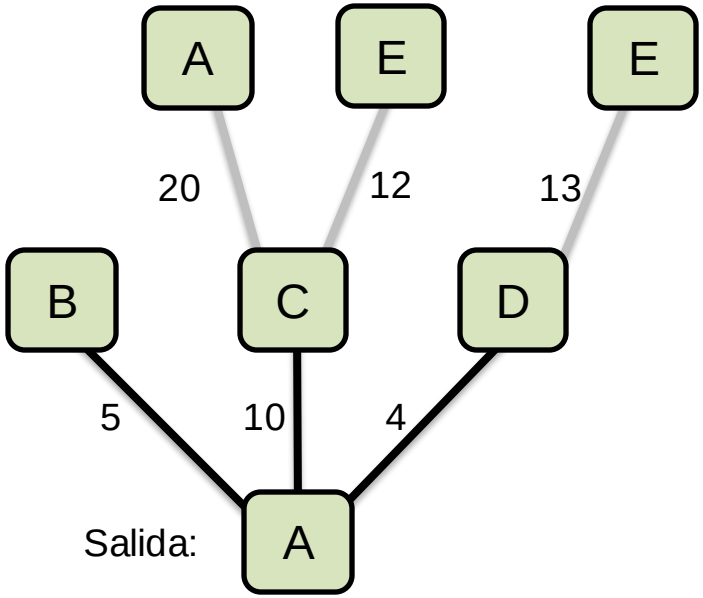
Algoritmo de Dijkstra

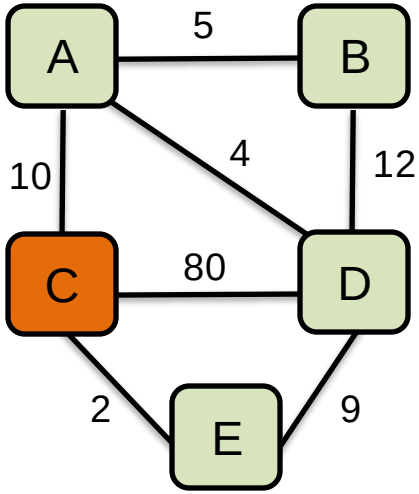
Objetivo: E

A	0
B	5
C	10
D	4
E	13

PQ





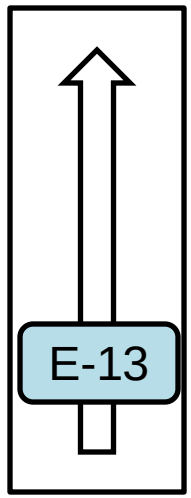


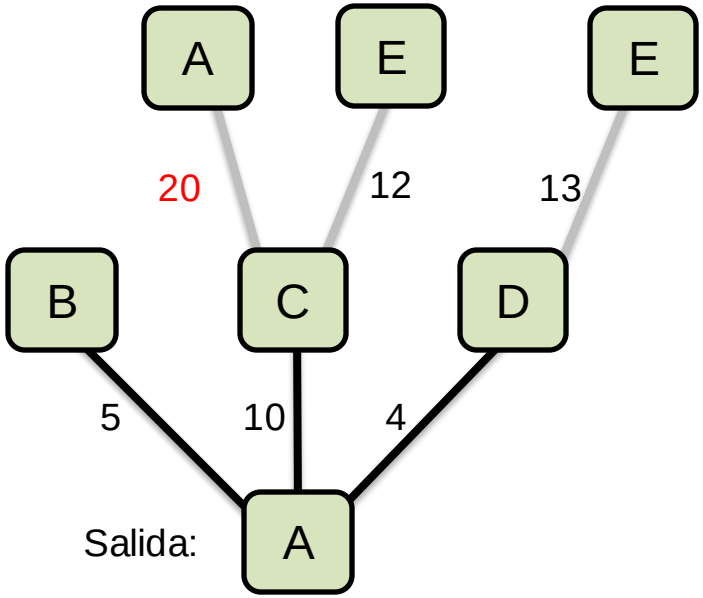
Algoritmo de Dijkstra

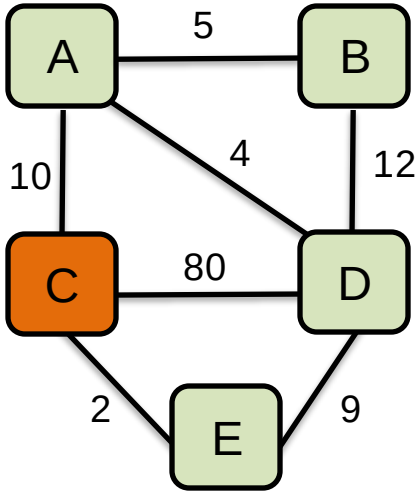
Objetivo: E

A	0
B	5
C	10
D	4
E	13

PQ





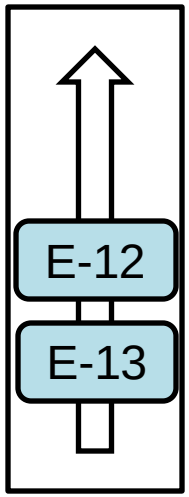


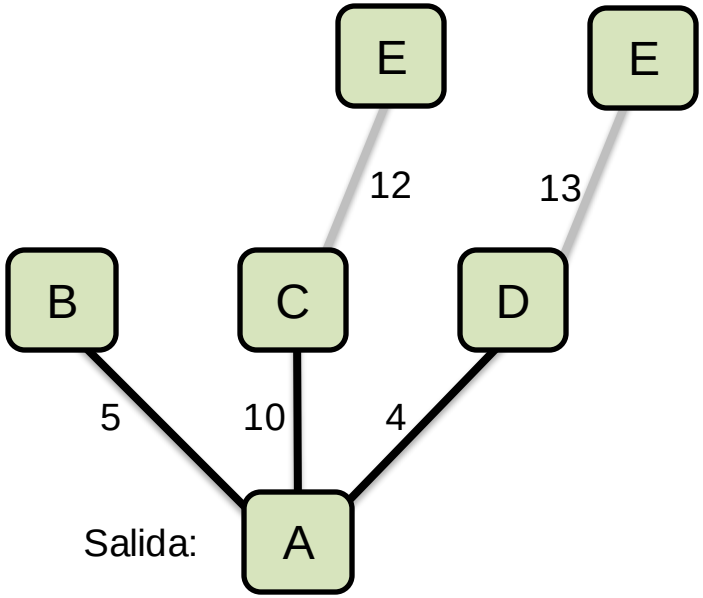
Algoritmo de Dijkstra

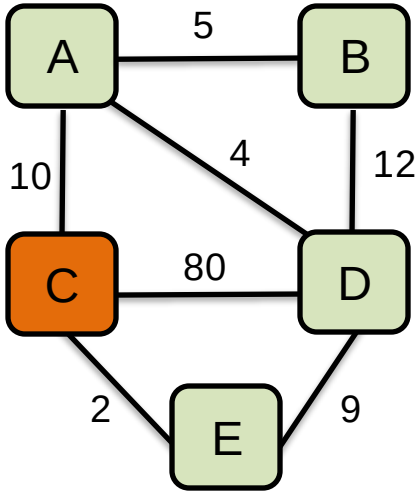
Objetivo: E

A	0
B	5
C	10
D	4
E	12

PQ





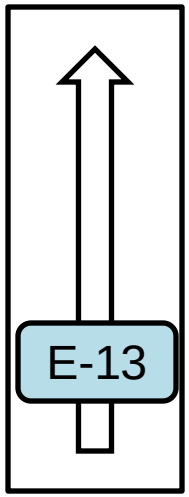


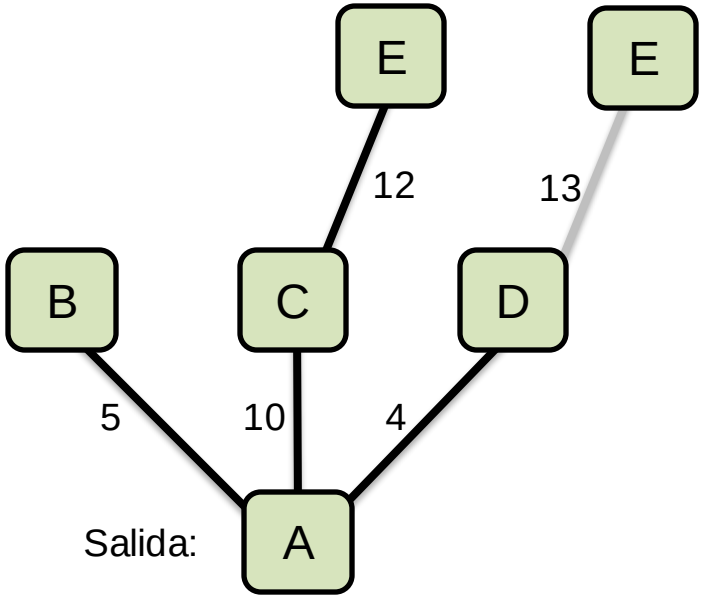
Algoritmo de Dijkstra

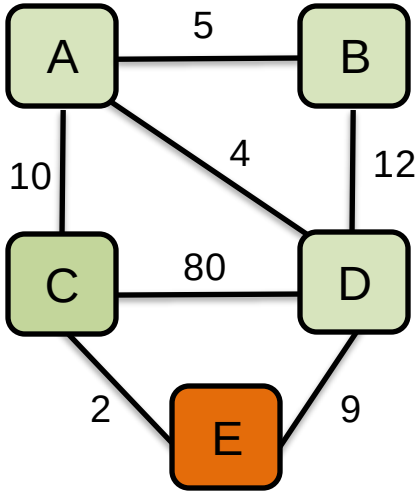
Objetivo: E

A	0
B	5
C	10
D	4
E	12

PQ





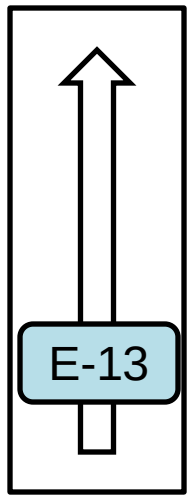


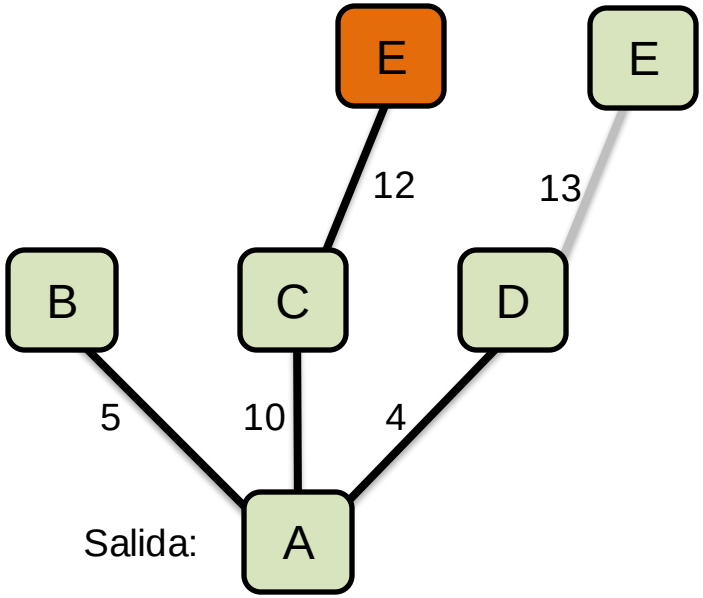
Algoritmo de Dijkstra

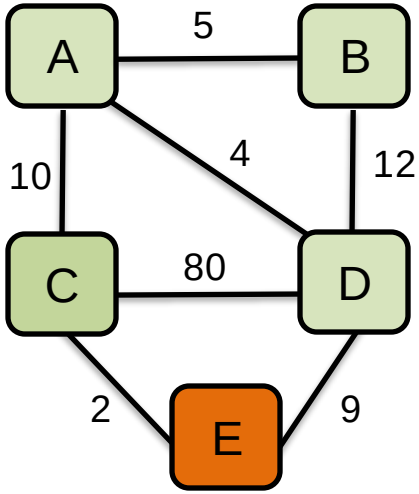
Objetivo: E

A	0
B	5
C	10
D	4
E	12

PQ







Algoritmo de Dijkstra

```
DIJKSTRA (Grafo G, nodo s)
        for u ∈ V[G] do
            distancia[u] = INFINITO
            padre[u] = NULL
            visto[u] = false
        distancia[s] = 0
        adicionar (cola, (s, distancia[s]))
        while cola no es vacía do
            u = extraer_mínimo(cola)
            visto[u] = true
            for todos v ∈ adyacencia[u] do
                if ¬ visto[v]
                    if distancia[v] > distancia[u] + peso (u, v) do
                        distancia[v] = distancia[u] + peso (u, v)
                        padre[v] = u
                        adicionar(cola,(v, distancia[v]))
```

Consejo:
Coged un código que funcione y al dosier!!!

https://github.com/stevenhalim/cpbook-code/tree/master/ch4/sssp

Algoritmos de Caminos más Cortos

Problema 2: ¿CUÁL ES EL CAMINO MÁS CORTO DEL NODO i AL RESTO?

Algoritmos de Caminos más Cortos

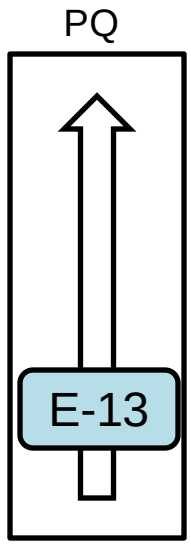
ALGORITMO DE DIJKSTRA

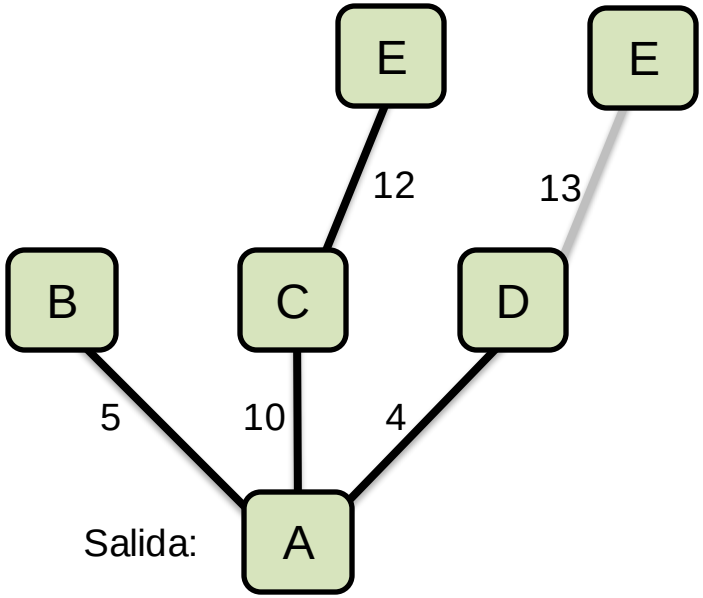
- Algoritmo Voraz

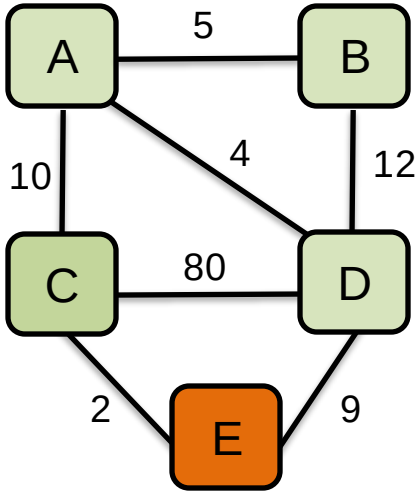
IDEA: Terminar hasta que la PQ esté vacía

Algoritmo de Dijkstra

A	0
B	5
C	10
D	4
E	12

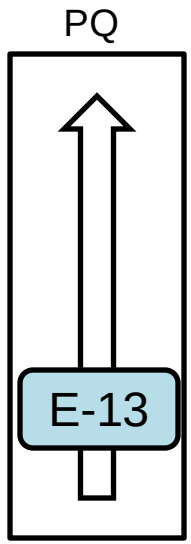


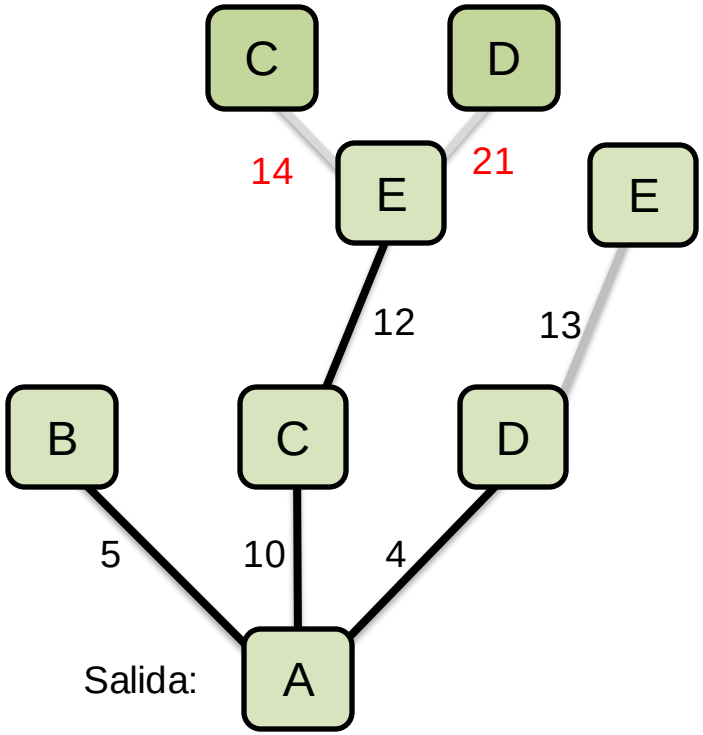


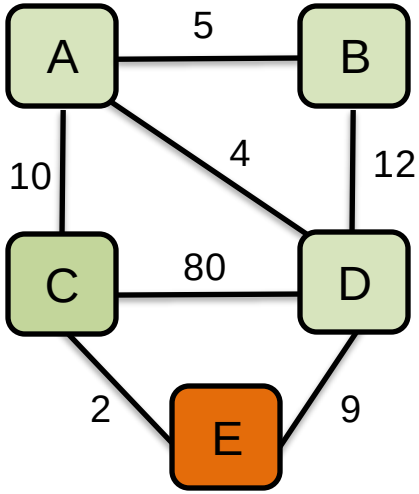


Algoritmo de Dijkstra

A	0
B	5
C	10
D	4
E	12

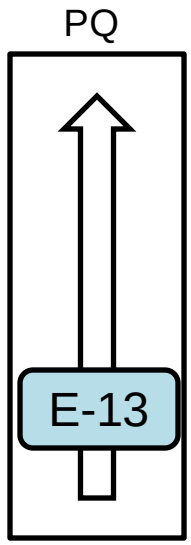


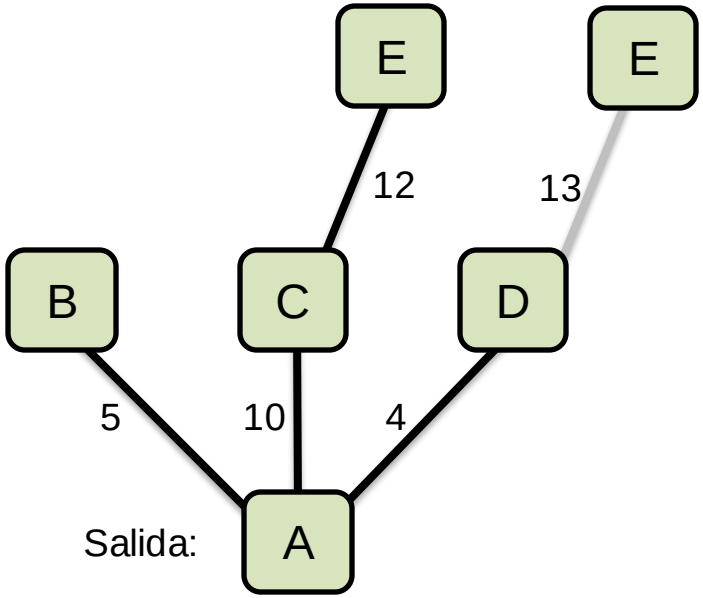


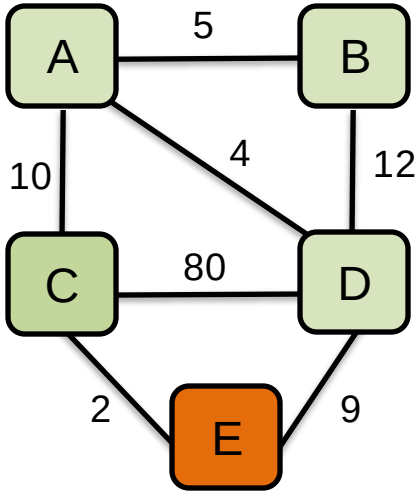


Algoritmo de Dijkstra

A	0
B	5
C	10
D	4
E	12

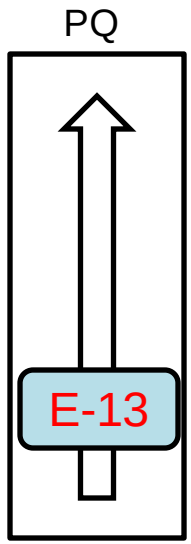


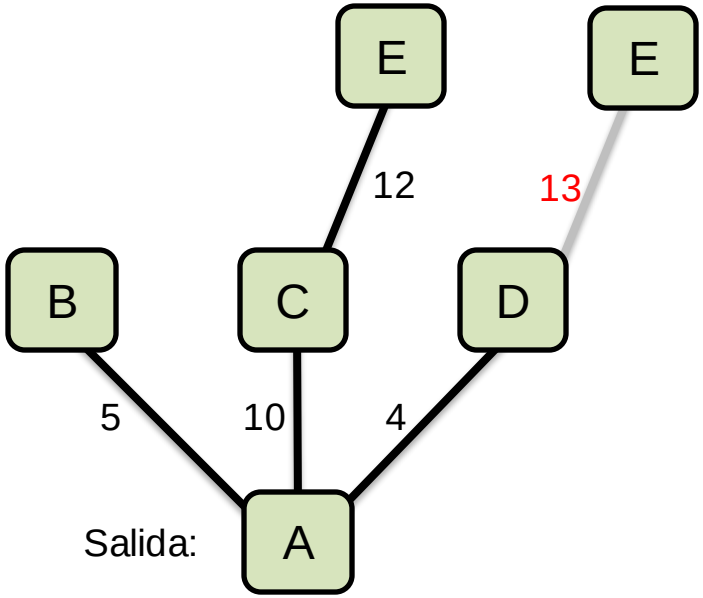


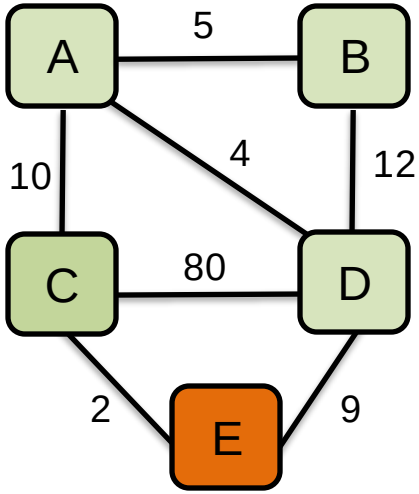


Algoritmo de Dijkstra

A	0
B	5
C	10
D	4
E	12



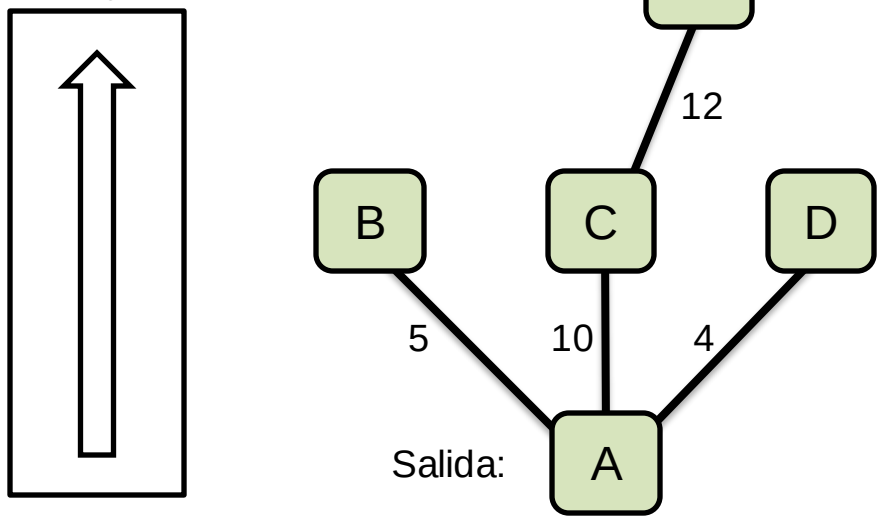


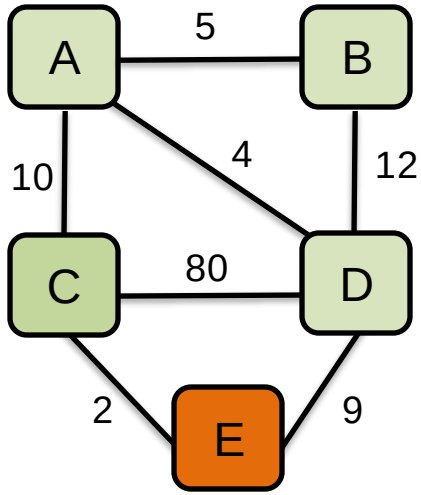


Algoritmo de Dijkstra

A	0
B	5
C	10
D	4
E	12

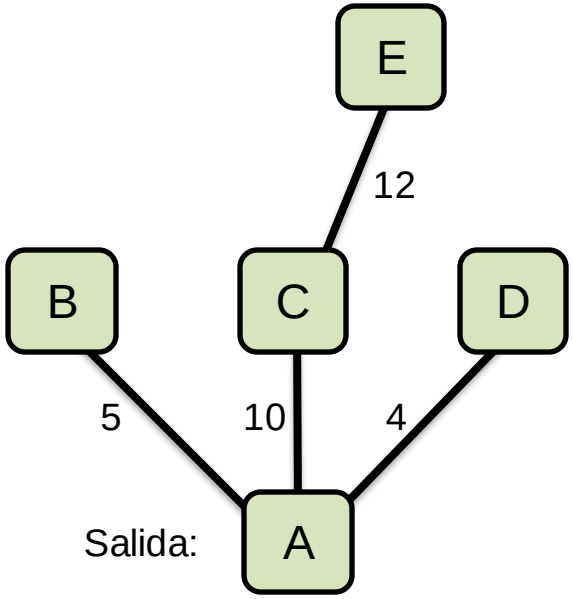
PQ





Algoritmo de Dijkstra

A	0
B	5
C	10
D	4
E	12



Algoritmos de Caminos más Cortos

Problema 3: ¿CUÁL ES EL CAMINO MÁS CORTO DESDE CUALQUIER NODO AL RESTO?

Algoritmos de Caminos más Cortos

ALGORITMO DE DIJKSTRA

- Algoritmo Voraz

IDEA: Lanzar un dijkstra desde todos los nodos...

Algoritmos de Caminos más Cortos

NO LO VOY A HACER :)

Algoritmos de Caminos más Cortos

ALGORITMO DE FLOYD-WARSHALL

- Algoritmo de Programación Dinámica

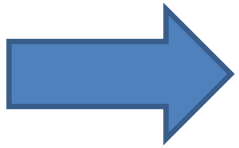
IDEA: ¿Qué es más rápido ir del nodo i al nodo j pasando por el nodo k o sin pasar?

¡¡Se implementa sobre la matriz de adyacencia!!

Algoritmo de Floyd-Warshall

```
for k from 0..N
  for i from 0..N
    for j from 0..N
      graph[i][j] = min( graph[i][j], graph[i][k] +
graph[k][j])
```

X	0	1	2	3	4
0	0	5	10	4	∞
1	5	0	∞	12	∞
2	10	∞	0	80	2
3	4	12	80	0	9
4	∞	∞	2	9	0



X	0	1	2	3	4
0	0	5	10	4	12
1	5	0	15	12	18
2	10	15	0	80	2
3	4	12	80	0	9
4	12	18	2	9	0

Algoritmo de Floyd-Warshall

¿¿¿¿Complejidad????

https://github.com/stevenhalim/cpbook-code/tree/master/ch4

```
for k from 0..N
  for i from 0..N
    for j from 0..N
      graph[i][j] = min( graph[i][j], graph[i][k] +
graph[k][j])
```

Algoritmo de Floyd-Warshall

¿¿¿¿Complejidad????

$O(n^3)$

```
for k from 0..N
  for i from 0..N
    for j from 0..N
      graph[i][j] = min( graph[i][j], graph[i][k] +
graph[k][j])
```

Algoritmo de Floyd-Warshall

¿¿¿¿Complejidad????

$O(n^3)$

```
for k from 0..N
  for i from 0..N
    for j from 0..N
      graph[i][j] = min( graph[i][j], graph[i][k] +
graph[k][j])
```

Algoritmo de Floyd-Warshall

¿¿¿¿Complejidad de lanzar n Dijkstras????

Floyd-Warshall : $O(n^3)$

Algoritmo de Floyd-Warshall

¿¿¿¿Complejidad de lanzar n Dijkstras????

N Dijkstras : $O(n^2 \log n)$

Floyd-Warshall : $O(n^3)$

Algoritmo de Floyd-Warshall

N Dijkstras : $O(n^2 \log n)$

¿¿¿¿Para qué usar Floyd-Warshall????

Floyd-Warshall : $O(n^3)$

Algoritmo de Floyd-Warshall

¡¡¡Observad los límites de los problemas!!!

Floyd-Warshall se
programa en **segundos**

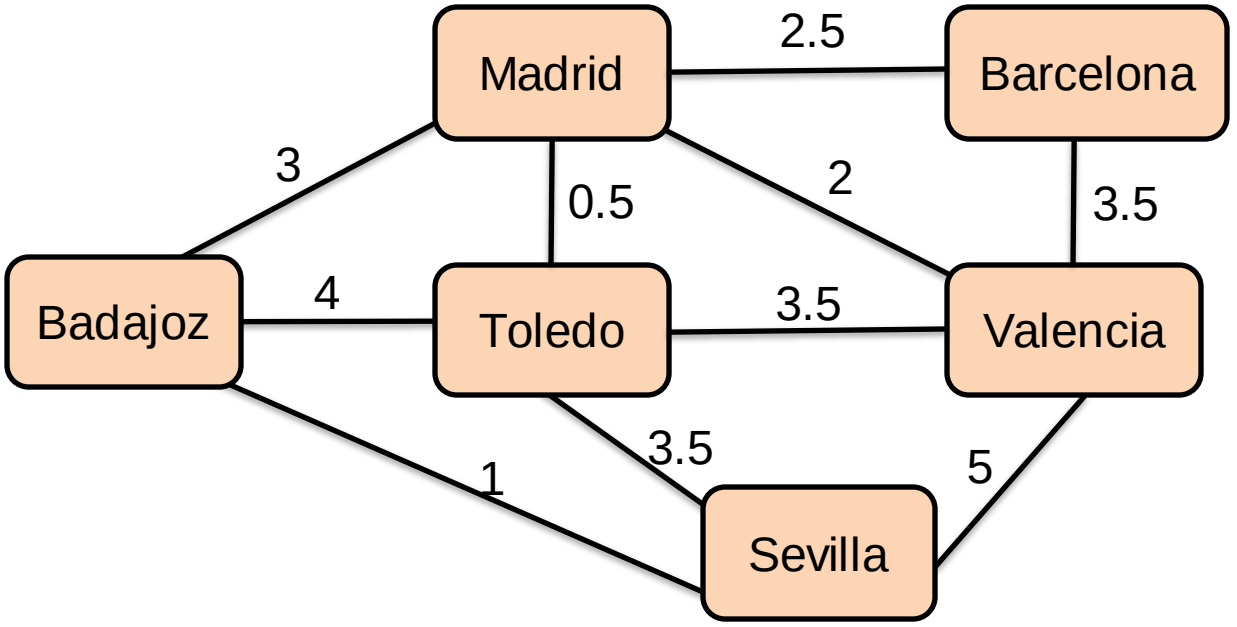
N Dijkstras : $O(n^2 \log n)$
n = 2000

Floyd-Warshall : $O(n^3)$
n = 400

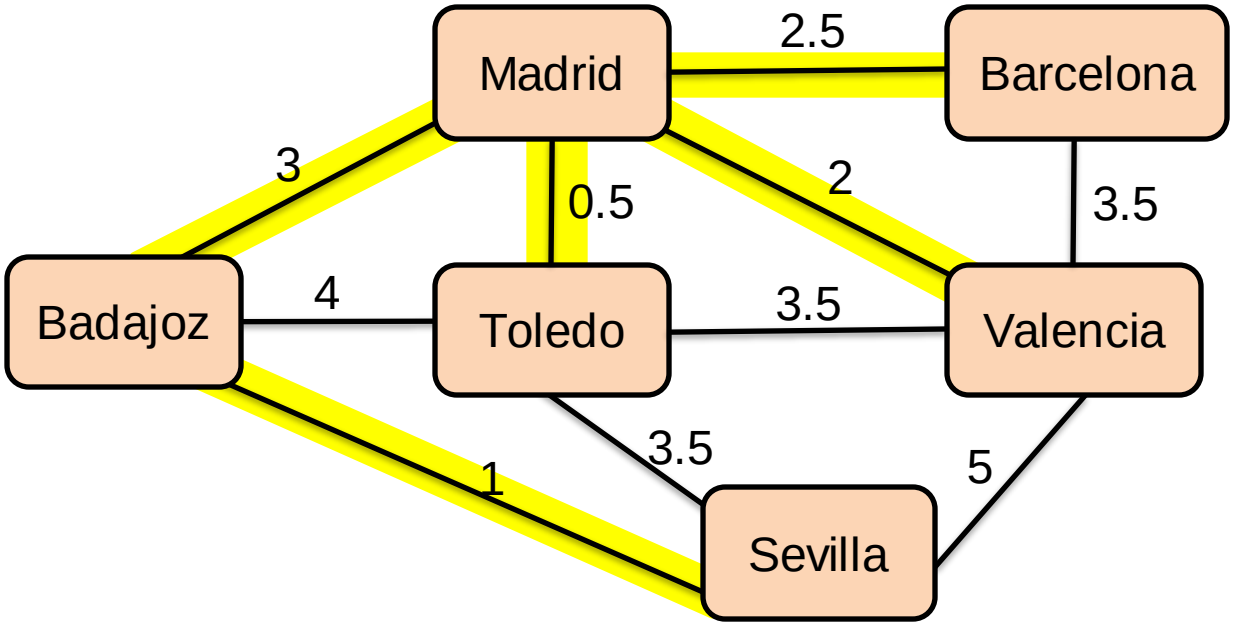
Algoritmos de Arboles de Recubrimiento

Problema 4: Encontrar el subgrafo conexo más pequeño que contiene todos los vértices.

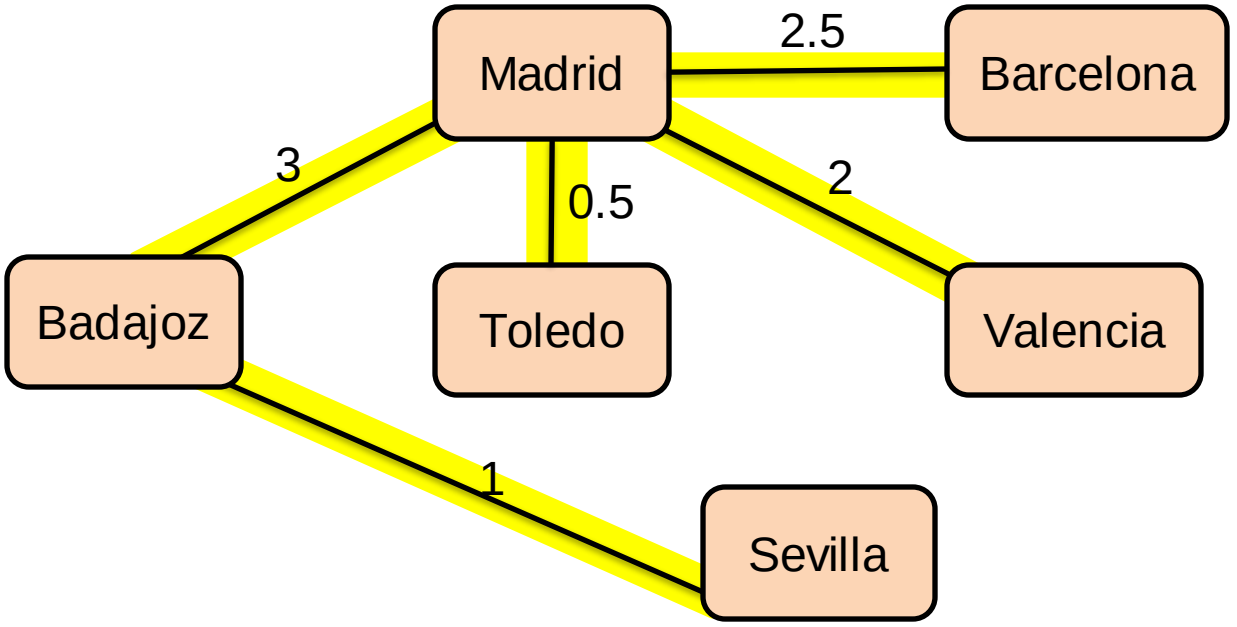
Algoritmos de Arboles de Recubrimiento



Algoritmos de Arboles de Recubrimiento

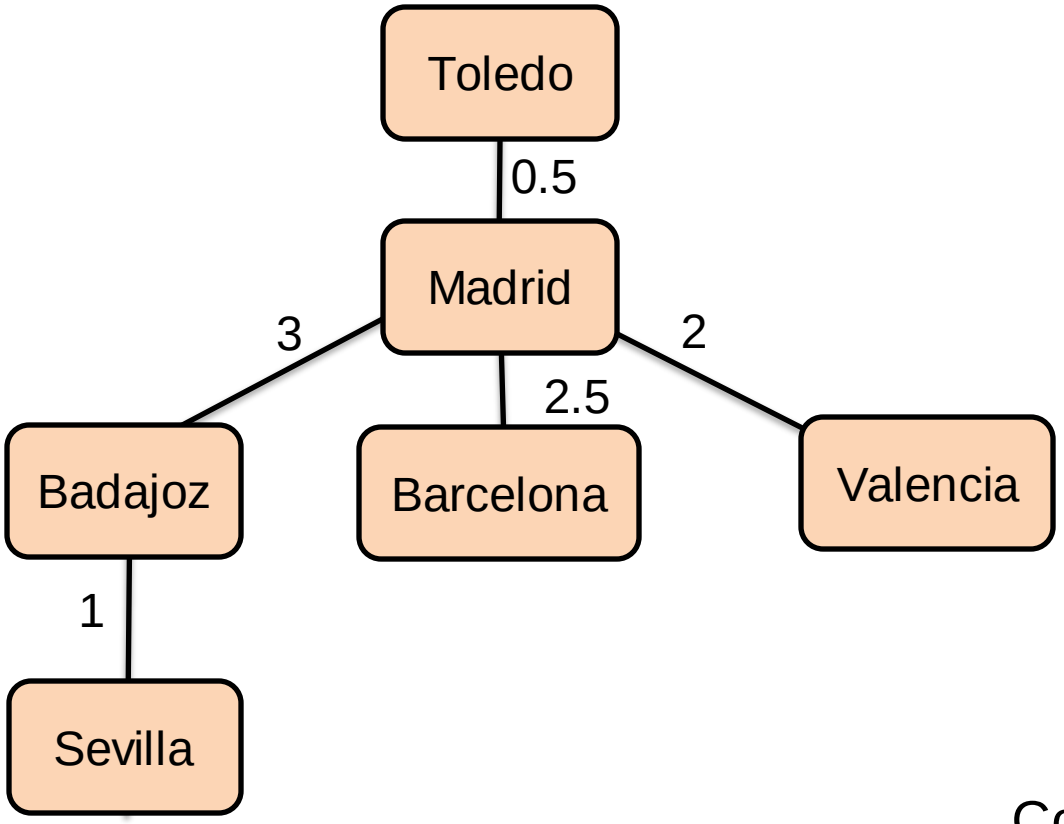


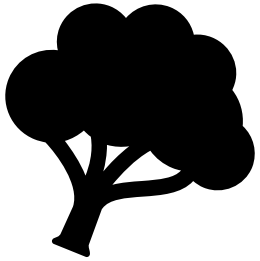
Algoritmos de Arboles de Recubrimiento



Coste Total: 9

Algoritmos de Arboles de Recubrimiento





Coste Total: 9

Algoritmos de Arboles de Recubrimiento

¿Siempre el grafo será un Árbol?

- Grafo Conexo
- Entre todos los vértices
- Acíclico

Algoritmos de Arboles de Recubrimiento

¿Siempre el grafo será un Árbol?

- Grafo Conexo
- Entre todos los vértices
- Acíclico

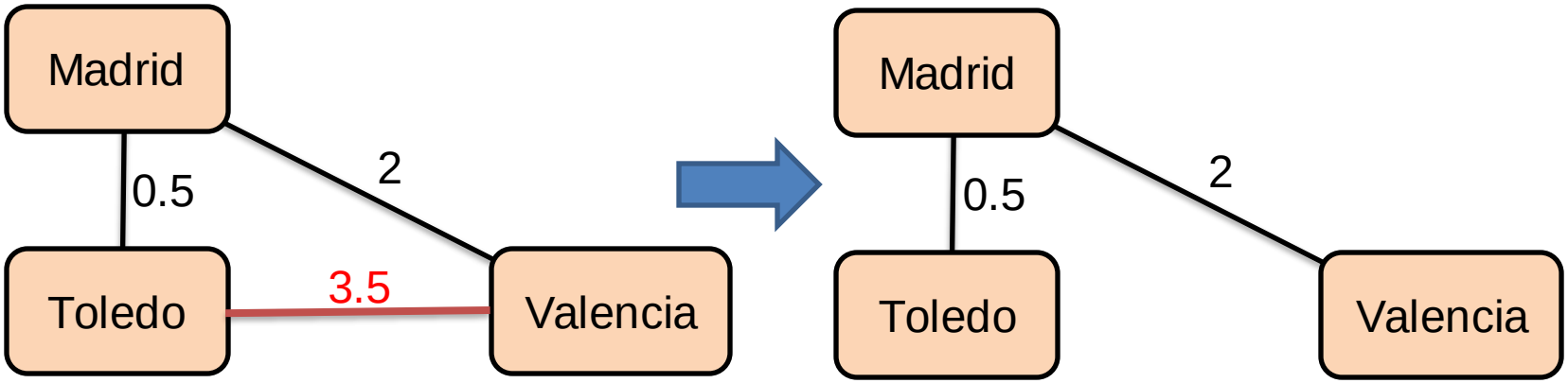
Algoritmos de Arboles de Recubrimiento

¿Siempre el grafo será un Árbol?

- Grafo Conexo
- Entre todos los vértices
- Acíclico?

Algoritmos de Arboles de Recubrimiento

Los ciclos son inútiles!!!



Algoritmos de Arboles de Recubrimiento

Dos Algoritmos:

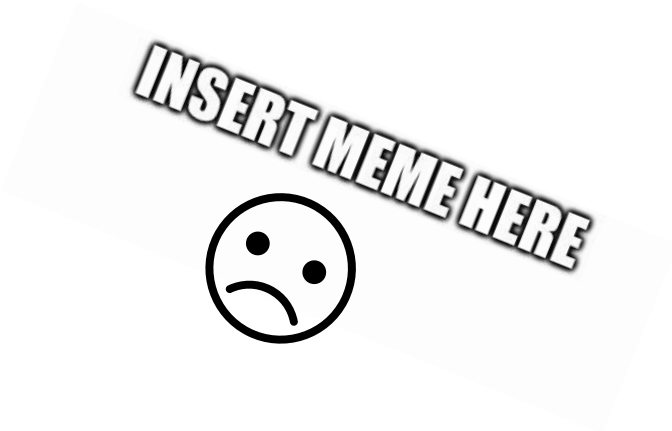
- Algoritmo de Prim
- Algoritmo de Kruskal

Algoritmos de Arboles de Recubrimiento

ALGORITMO DE KRUSKAL

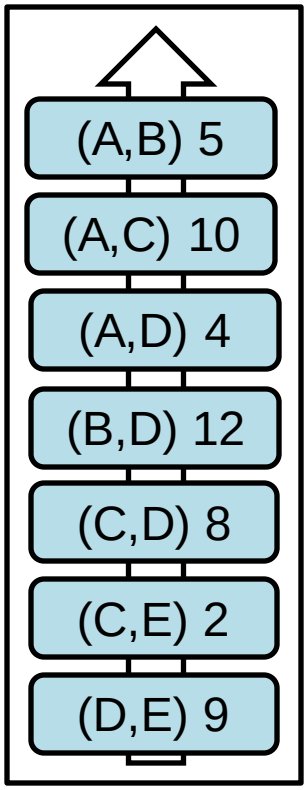
- Algoritmo Voraz

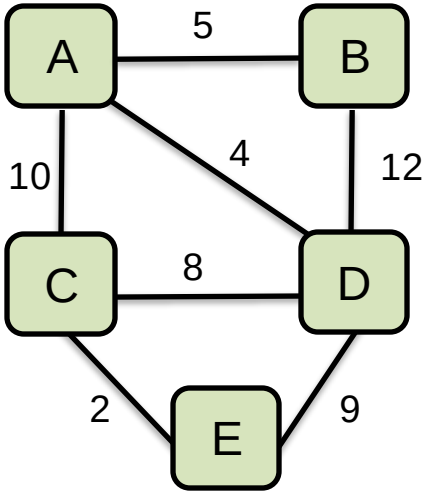
IDEA: Ir escogiendo las aristas más pequeñas que no hagan ciclos hasta que el árbol se complete.



Algoritmo de Kruskal

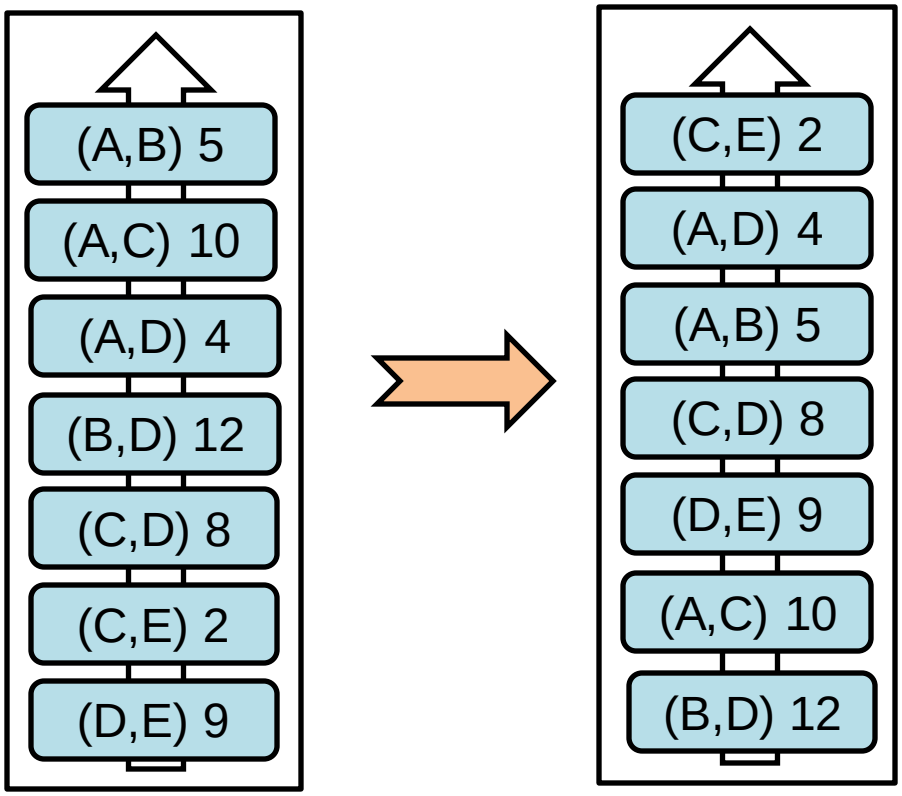
Valor: 0

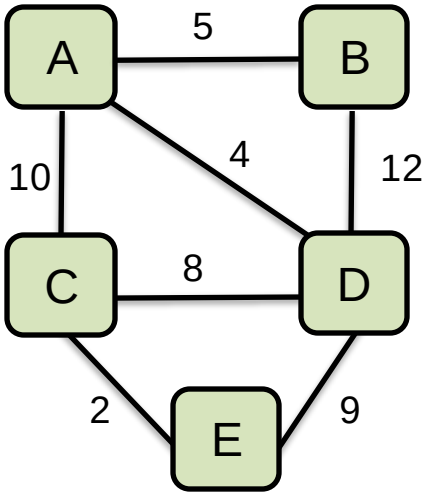




Algoritmo de Kruskal

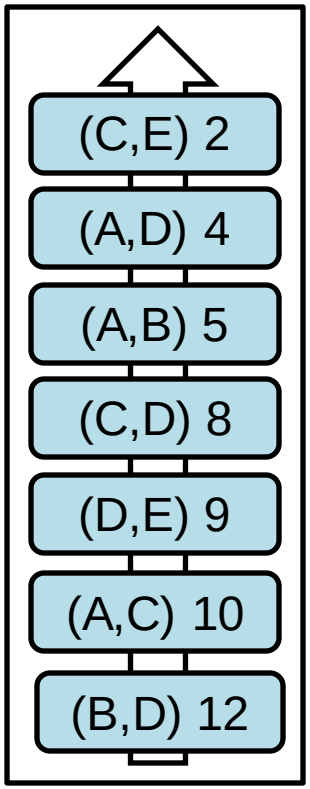
Valor: 0

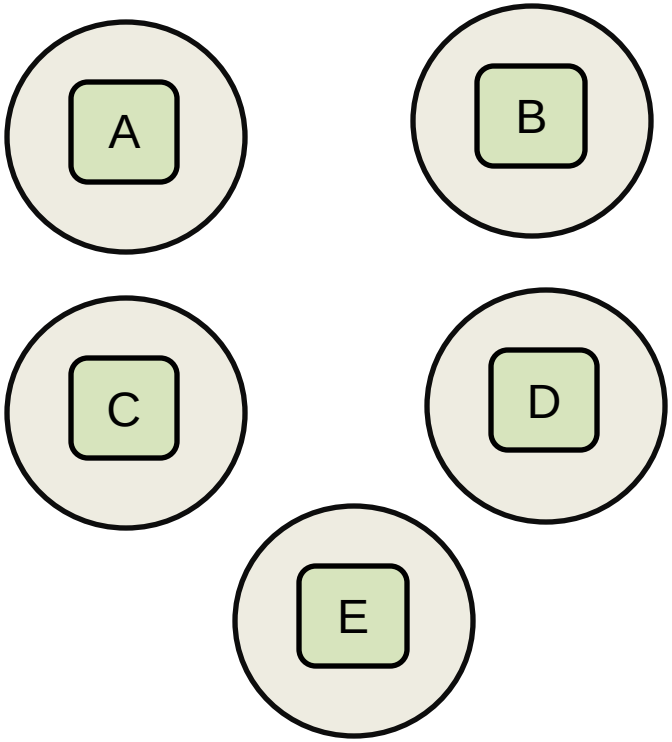


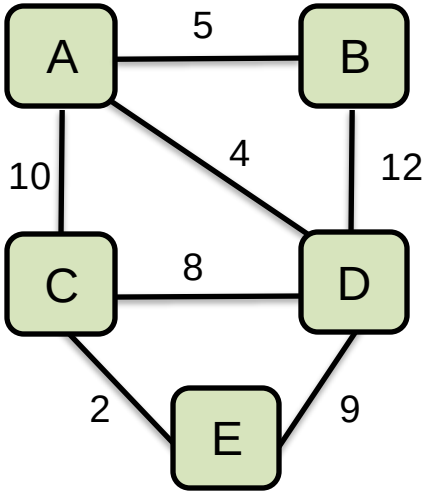


Algoritmo de Kruskal

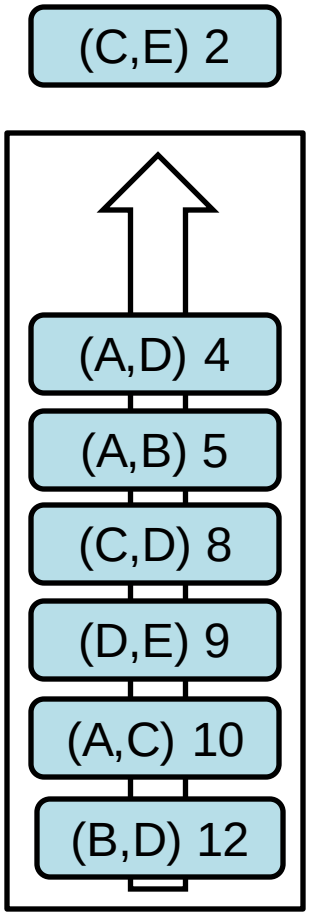
Valor: 0



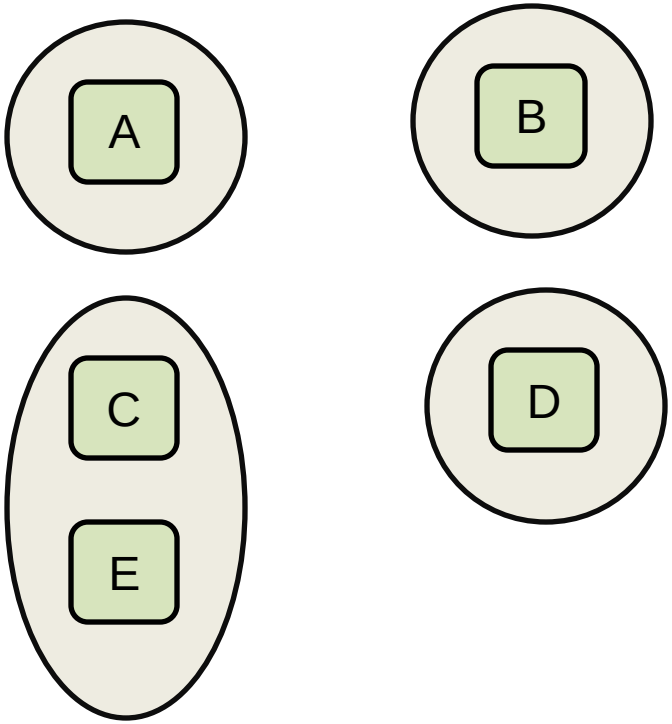


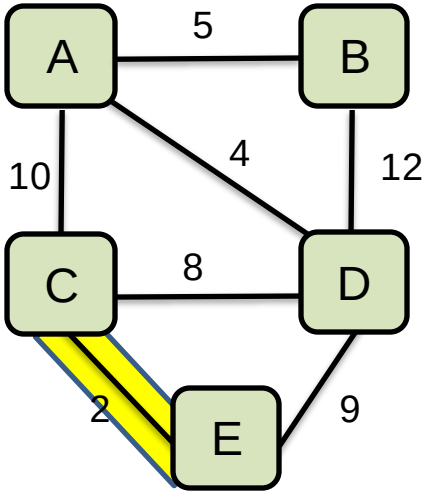


Algoritmo de Kruskal



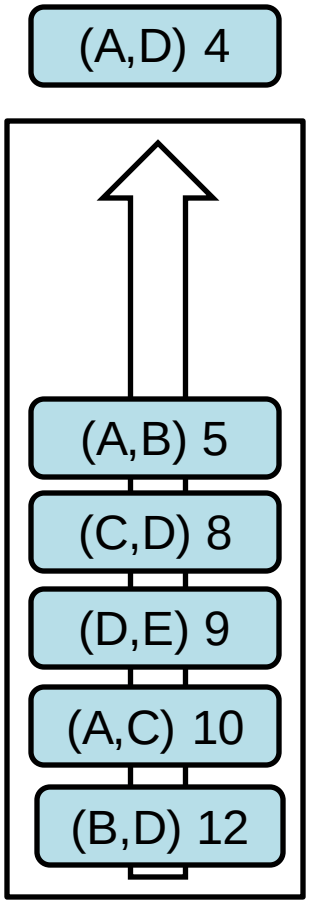
Valor: 2

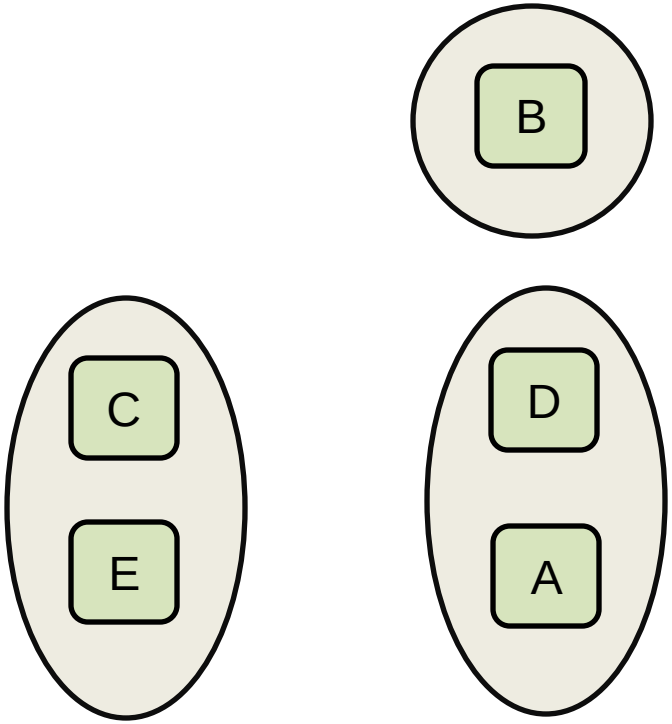


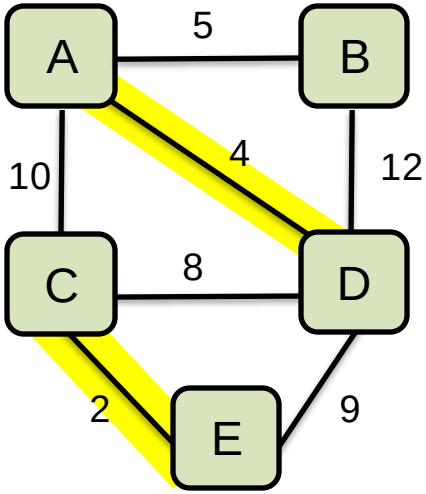


Algoritmo de Kruskal

Valor: 6

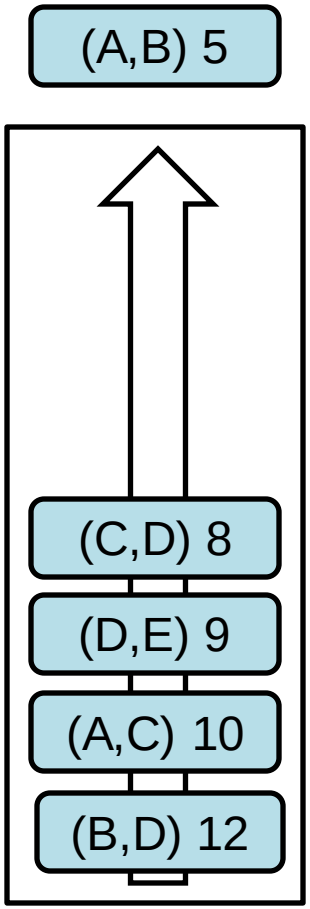


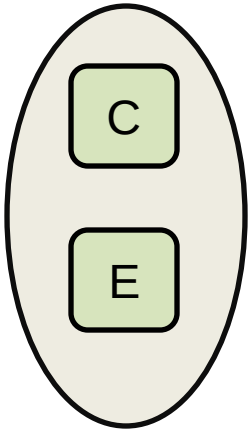


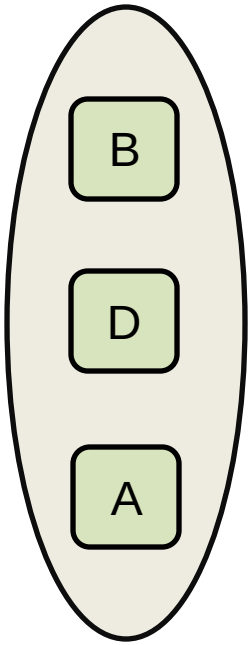


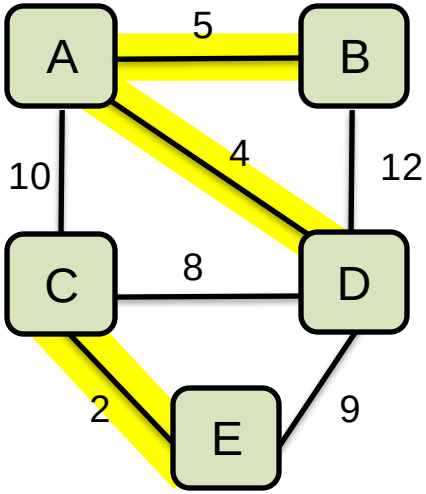
Algoritmo de Kruskal

Valor: 11



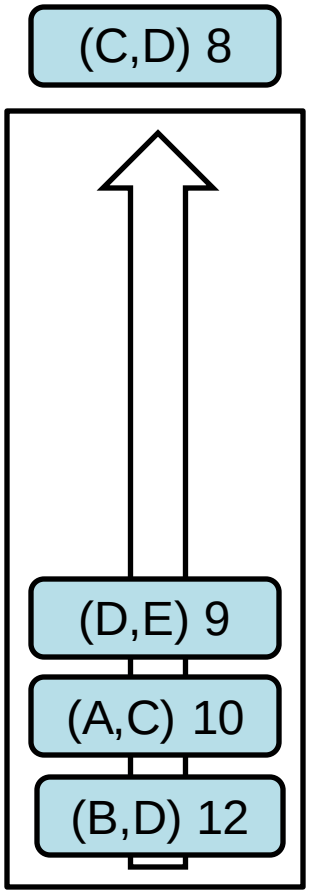


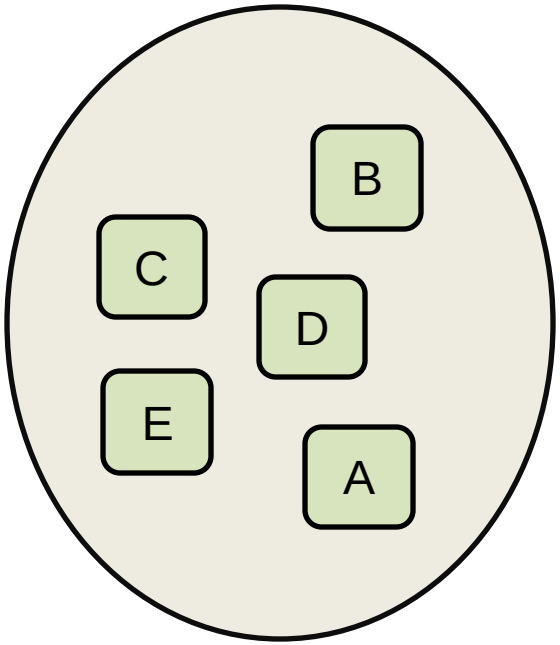


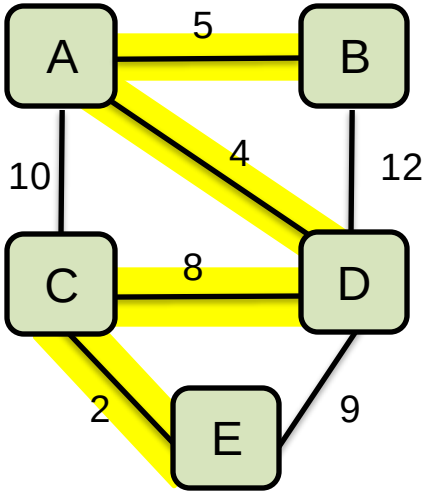


Algoritmo de Kruskal

Valor: 19

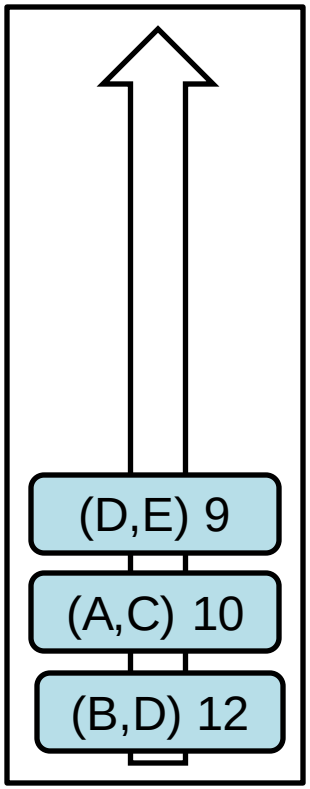


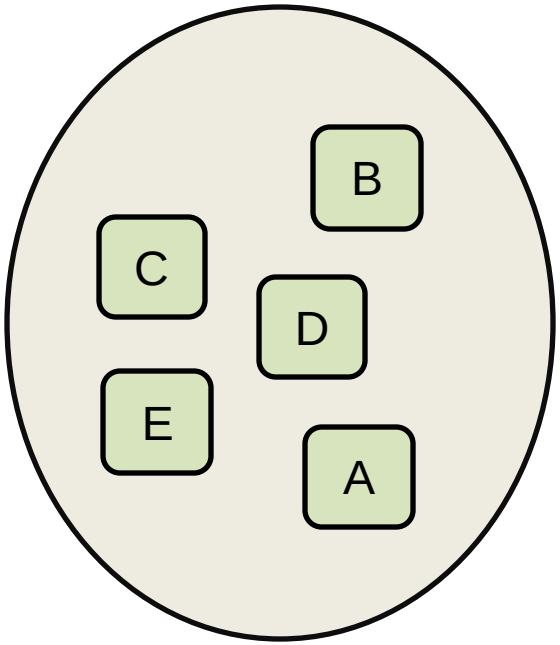


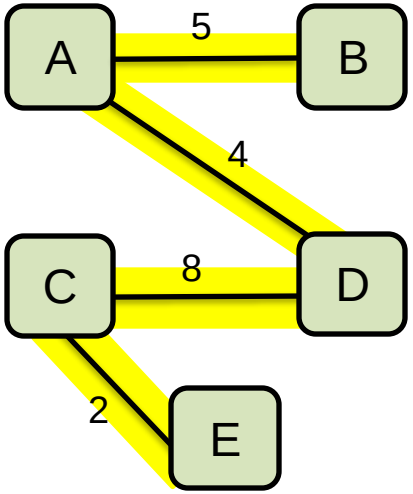


Algoritmo de Kruskal

Valor: 19

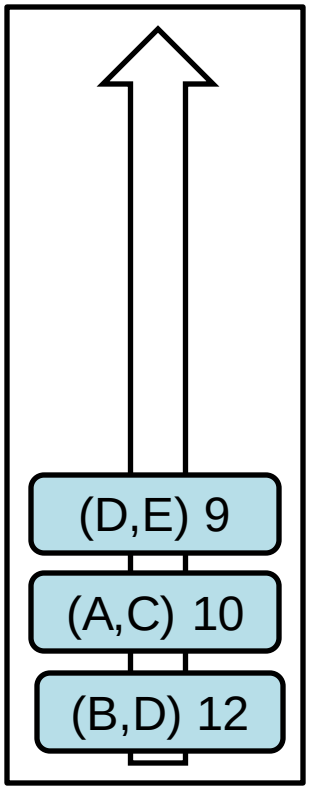


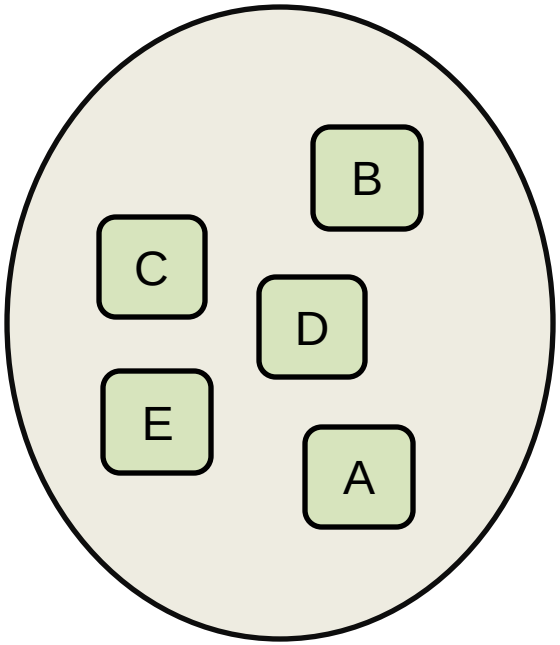


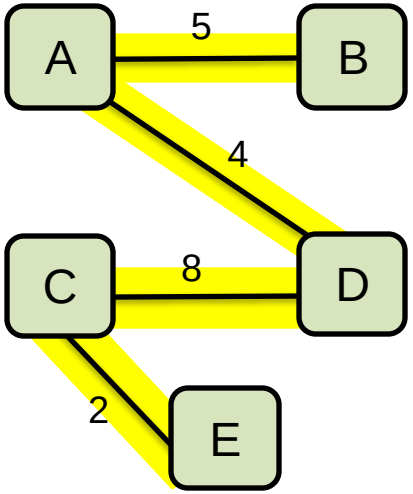


Algoritmo de Kruskal

Valor: 19







Algoritmo de Kruskal

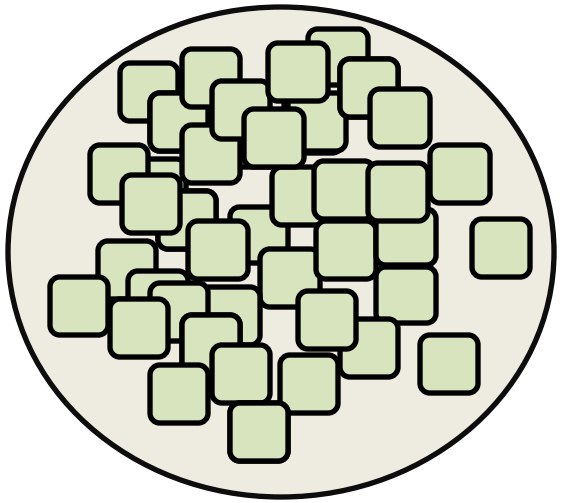
¿Cómo hacemos los conjuntos?

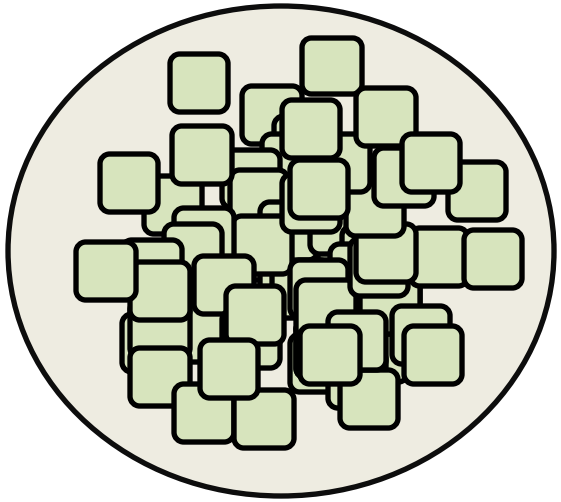
- Sets??

Algoritmo de Kruskal

¿Cómo hacemos los conjuntos?

- Sets??

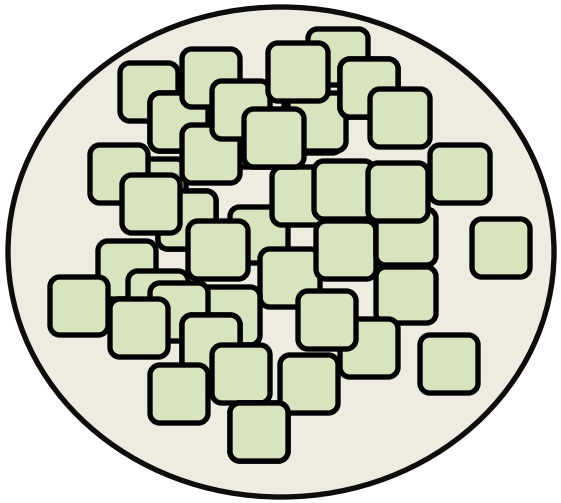




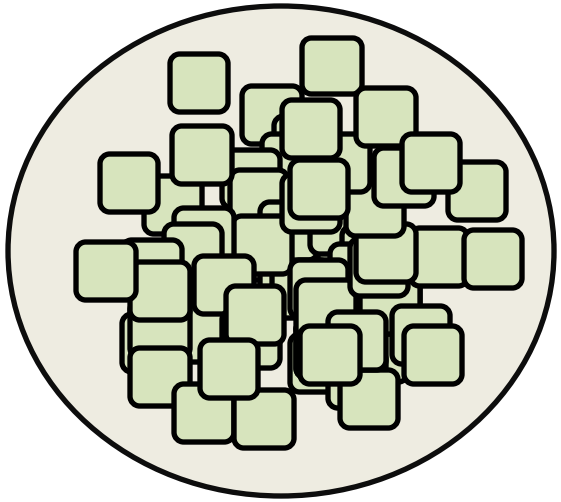
Algoritmo de Kruskal

¿Cómo hacemos los conjuntos?

- Sets??



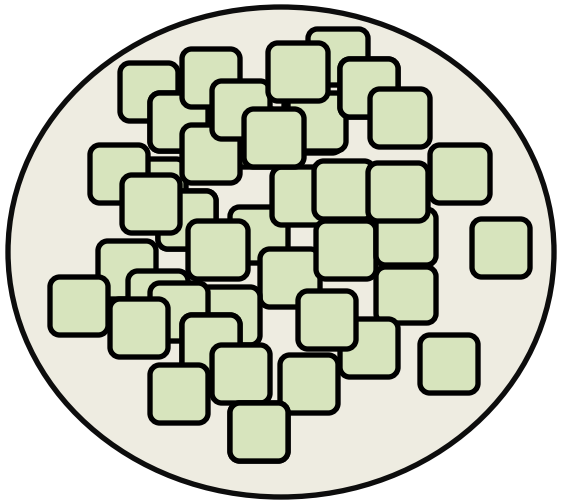
Tenemos que eliminar cada nodo de un conjunto y añadirlo en el otro



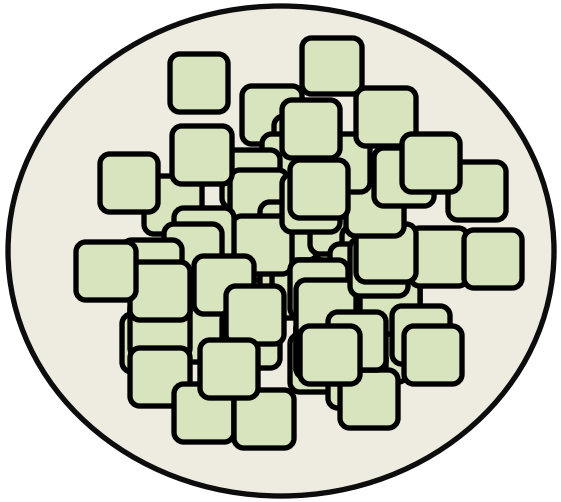
Algoritmo de Kruskal

¿Cómo hacemos los conjuntos?

- Sets??



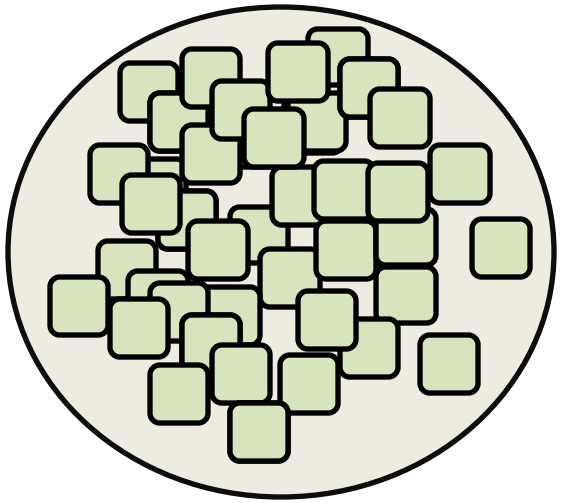
¿Complejidad?



Algoritmo de Kruskal

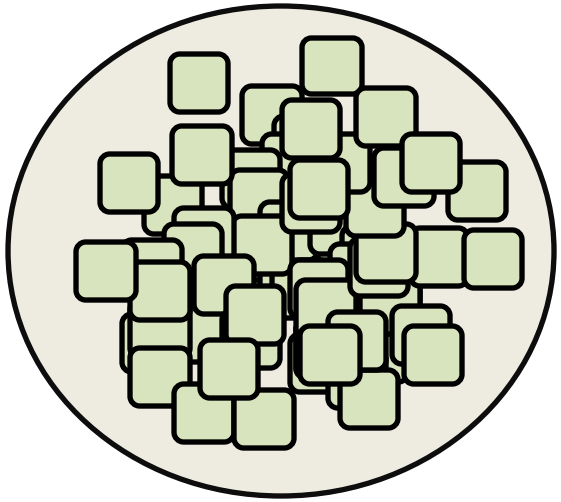
¿Cómo hacemos los conjuntos?

- Sets??



¿Complejidad?

O(n)

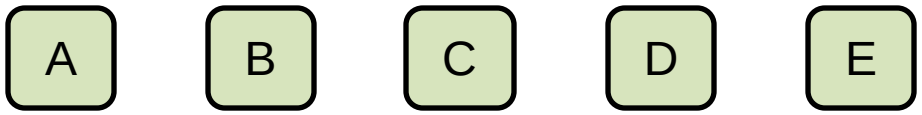


Algoritmo de Kruskal

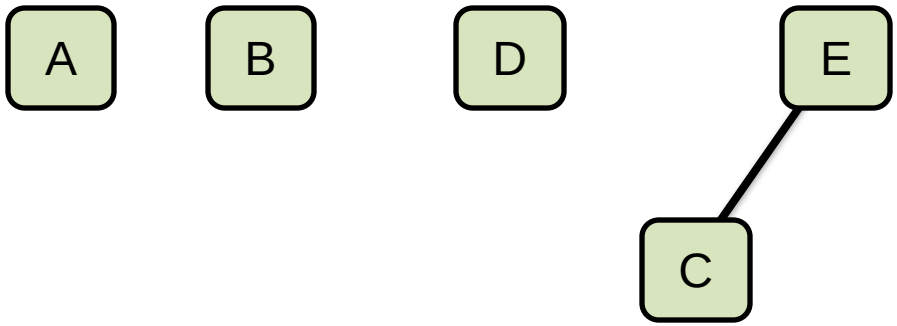
¿Cómo hacemos los conjuntos?

- Union-Find
  - Todos los conjuntos son árboles
  - Si dos nodos tienen la misma raíz, se considera que están en el mismo conjunto
  - Operaciones básicas:
    - Buscar
    - Unir
    - Igual

Algoritmo de Kruskal



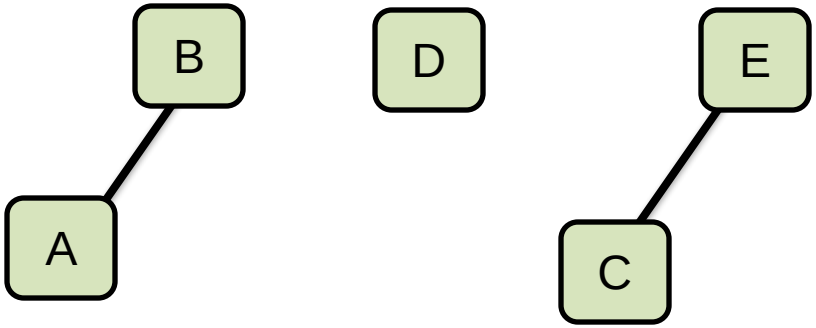
Algoritmo de Kruskal



Unir

(C,E)

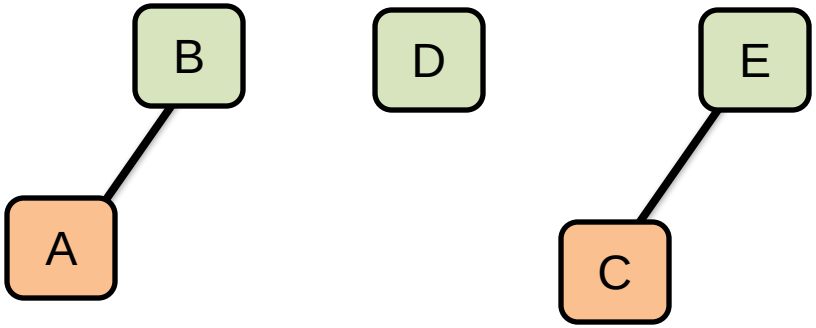
Algoritmo de Kruskal



Unir

(A,B)

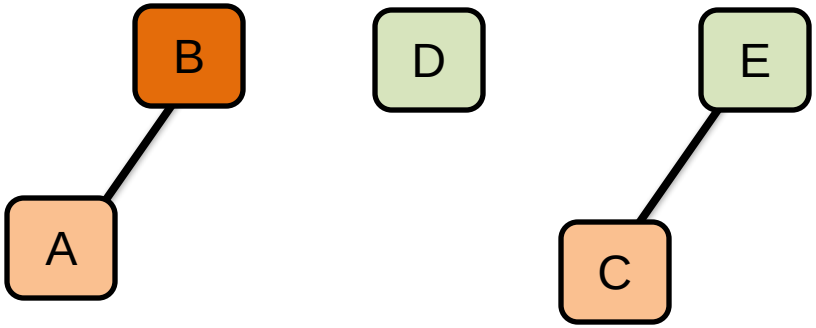
Algoritmo de Kruskal



¿Igual?

(C,A)

Algoritmo de Kruskal

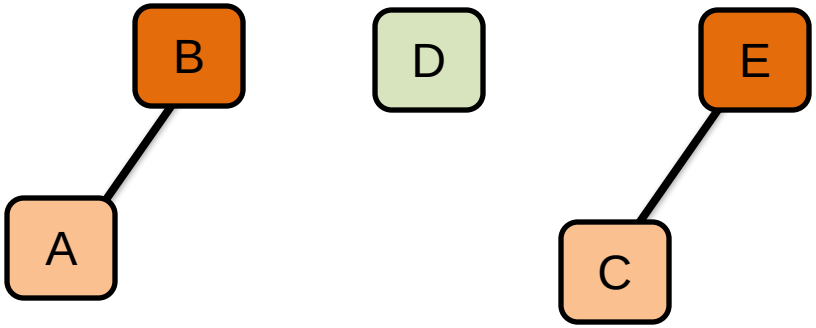


¿Igual?

(C,A)

B = ?

Algoritmo de Kruskal

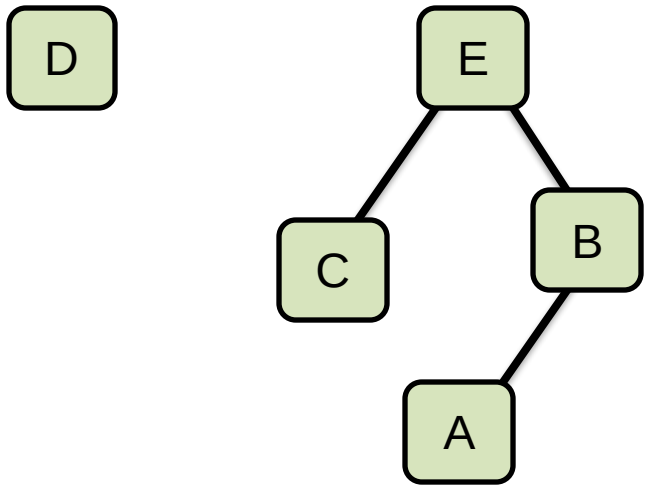


¿Igual?

(C,A)

B /= E

Algoritmo de Kruskal



Unir

(B,C)

Algoritmo de Kruskal

```
función Kruskal(G)
   Para cada v en V[G] hacer
     Nuevo conjunto C(v) = {v}.
   Nuevo heap Q que contiene todas las aristas de G,
ordenando por su peso
   Defino un árbol T = ∅
   Mientras T tenga menos de n-1 aristas y
!Q.vacío() hacer
     (u,v) ← Q.sacarMin()
     Si C(v) ≠ C(u) hacer
       Agregar arista (v,u) a  T
       Merge C(v) y C(u) en el conjunto
   Responder árbol T
```

Consejo:
Coged un código que funcione y al dosier!!!

https://github.com/stevenhalim/cpbook-code/tree/master/ch4/mst

Algoritmos de Arboles de Recubrimiento

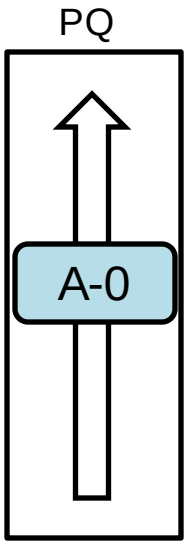
ALGORITMO DE PRIM

- Algoritmo Voraz

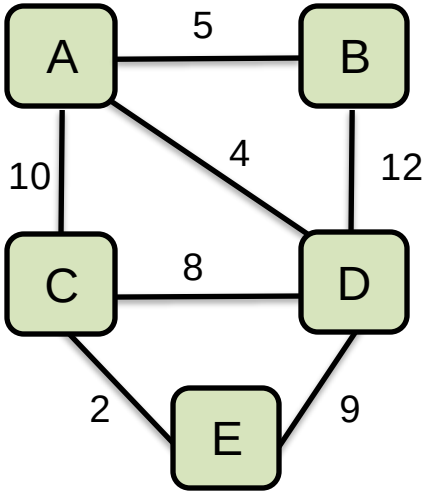
IDEA: Iniciar desde un nodo y hacer crecer el árbol por las aristas más pequeñas

Algoritmo de Prim

Valor: 0

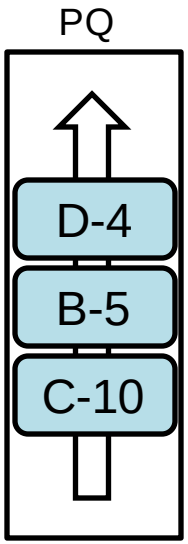


A	
B	
C	
D	
E	

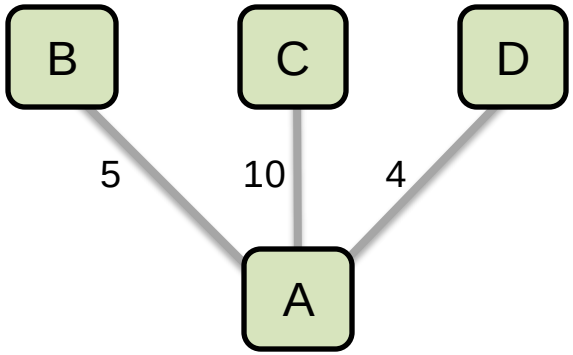


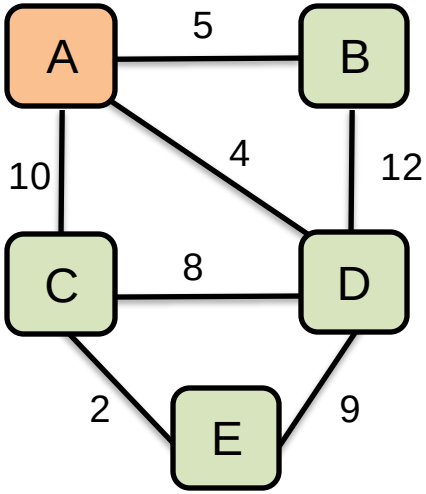
Algoritmo de Prim

Valor: 0



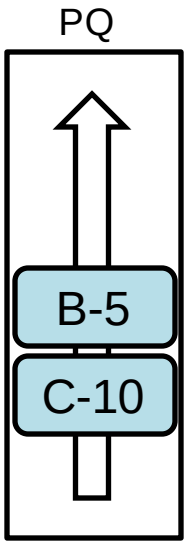
A	X
B	
C	
D	
E	



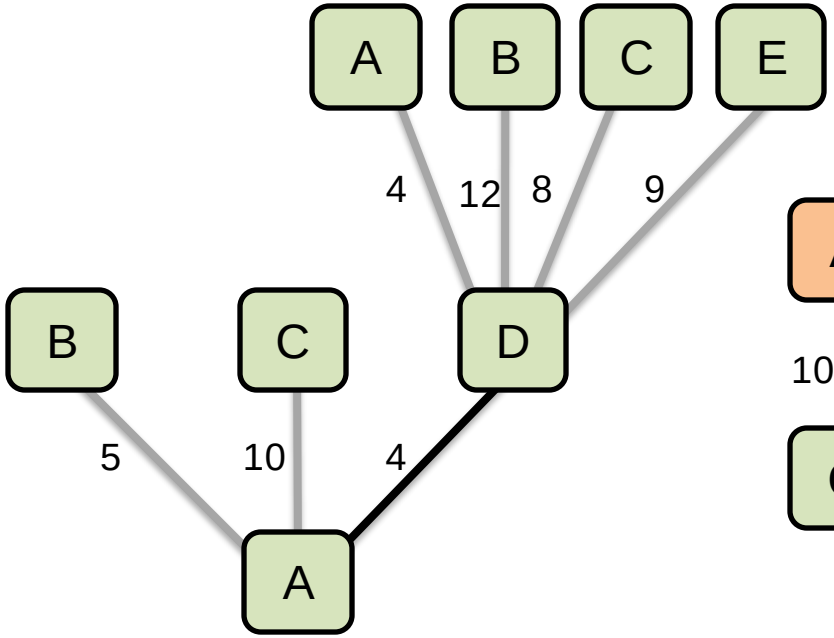


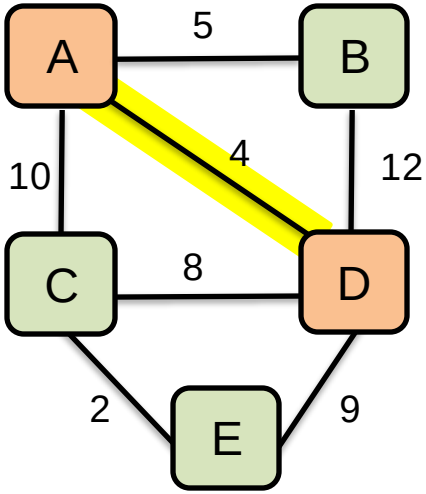
Algoritmo de Prim

Valor: 4



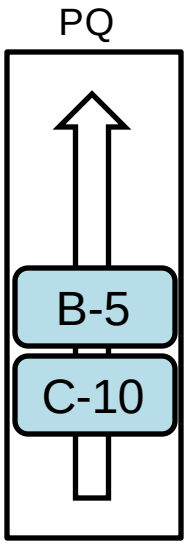
A	X
B	
C	
D	X
E	



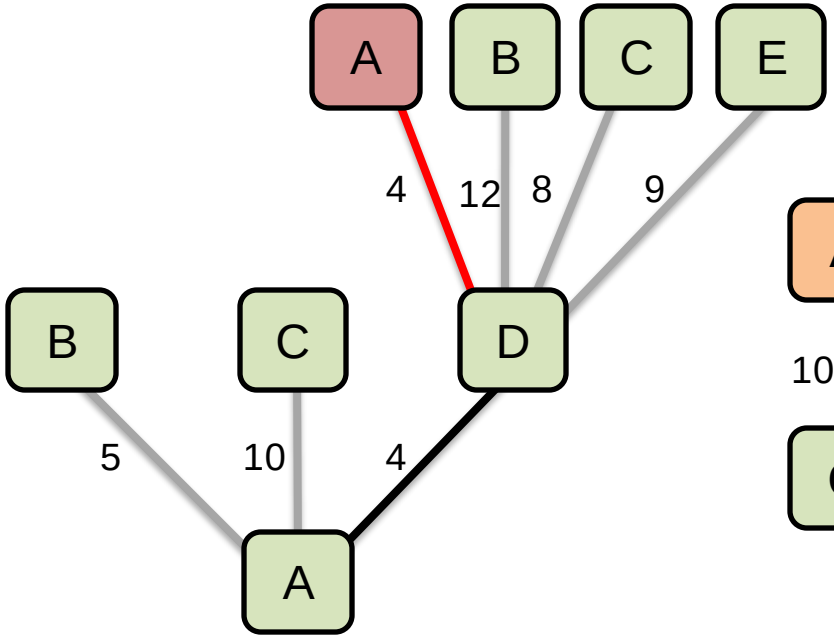


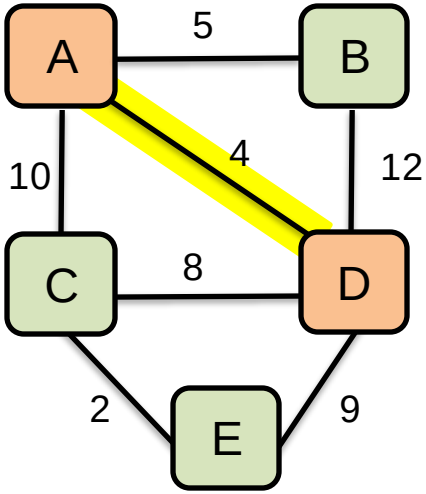
Algoritmo de Prim

Valor: 4



A	X
B	
C	
D	X
E	

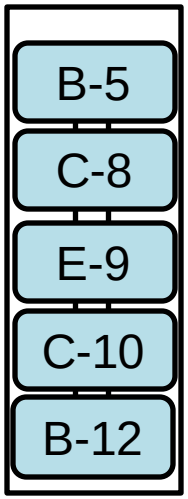




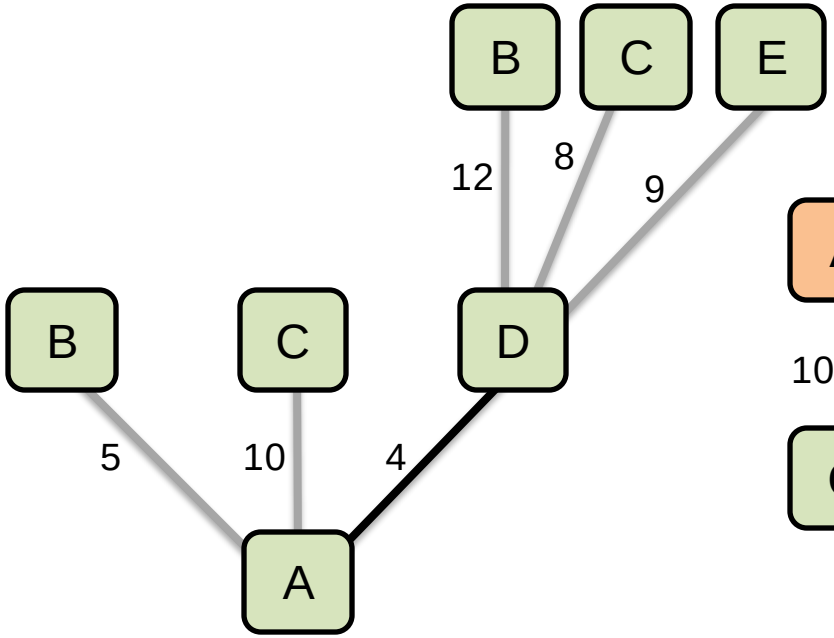
Algoritmo de Prim

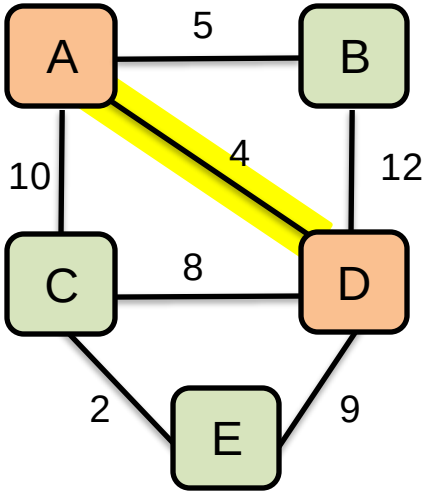
Valor: 4

PQ



A	X
B	
C	
D	X
E	

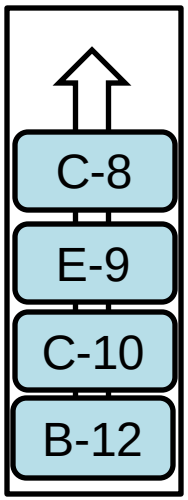




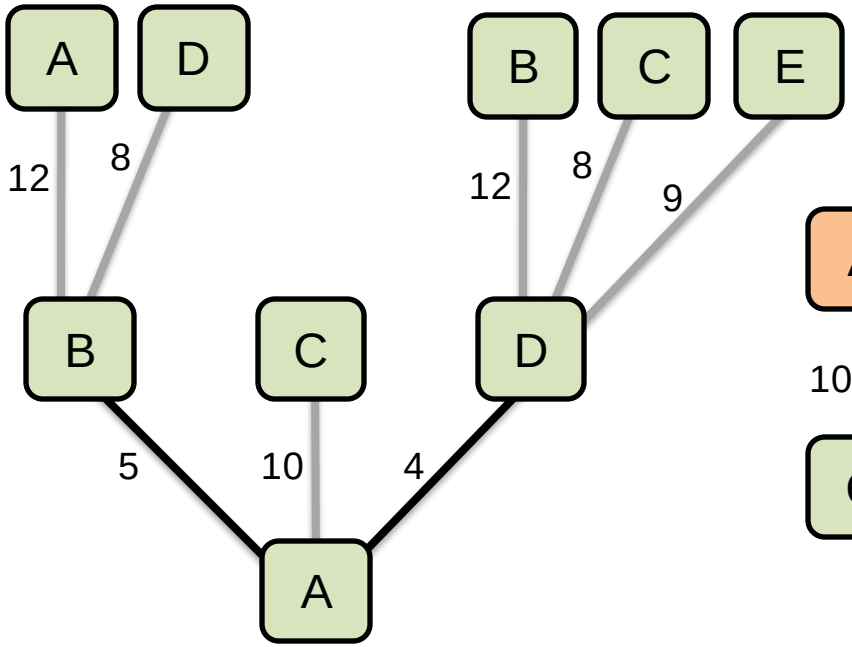
Algoritmo de Prim

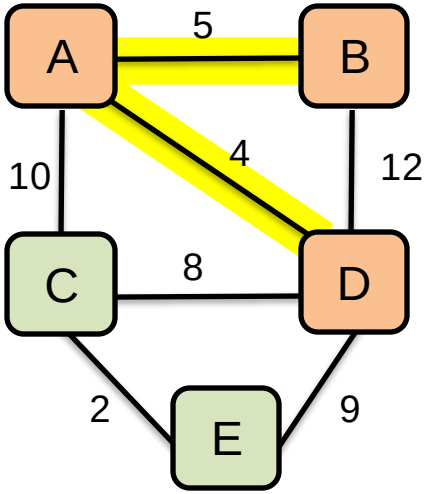
Valor: 9

PQ



A	X
B	X
C	
D	X
E	

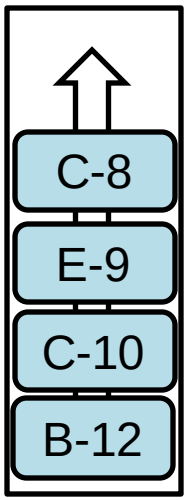




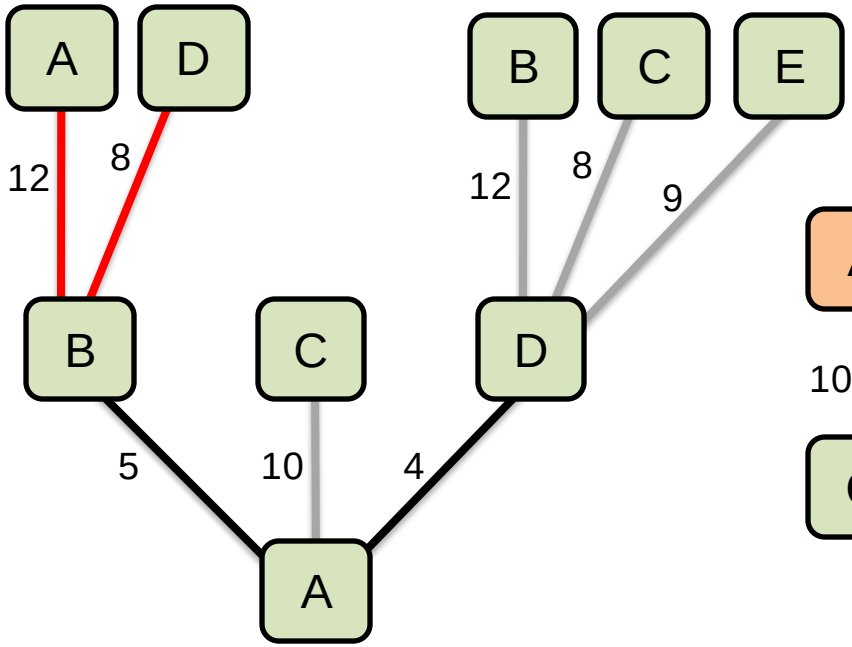
Algoritmo de Prim

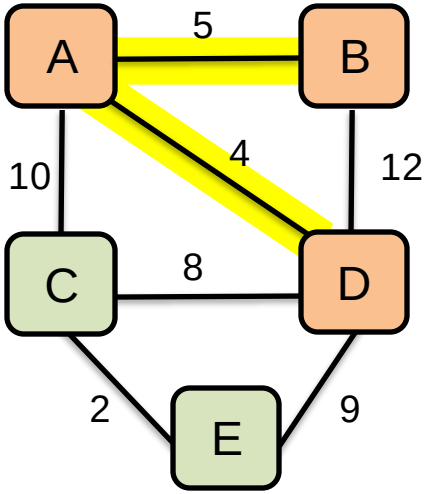
Valor: 9

PQ



A	X
B	X
C	
D	X
E	

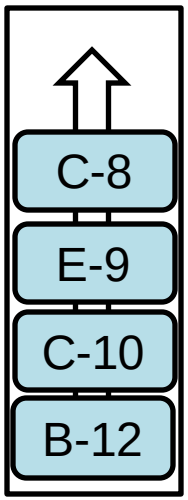




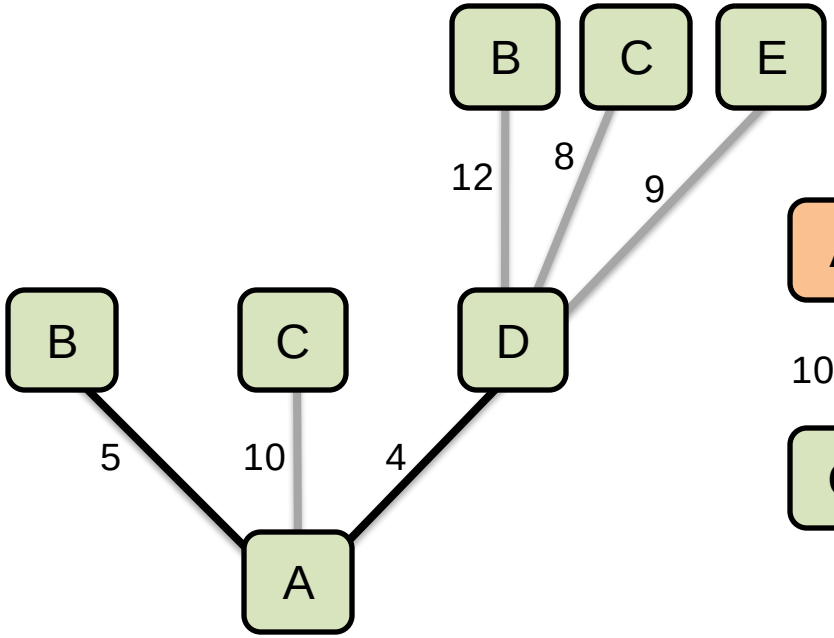
Algoritmo de Prim

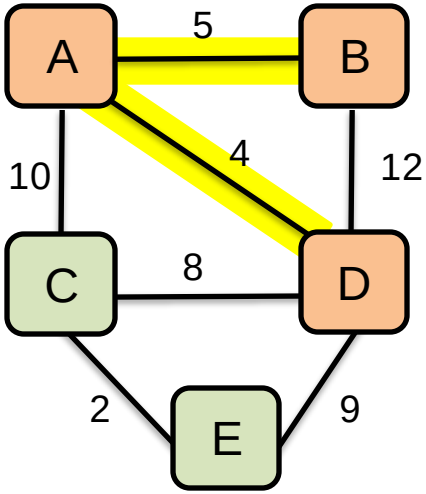
Valor: 9

PQ



A	X
B	X
C	
D	X
E	



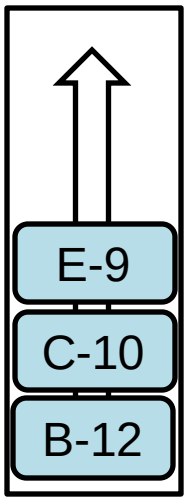


Algoritmo de Prim

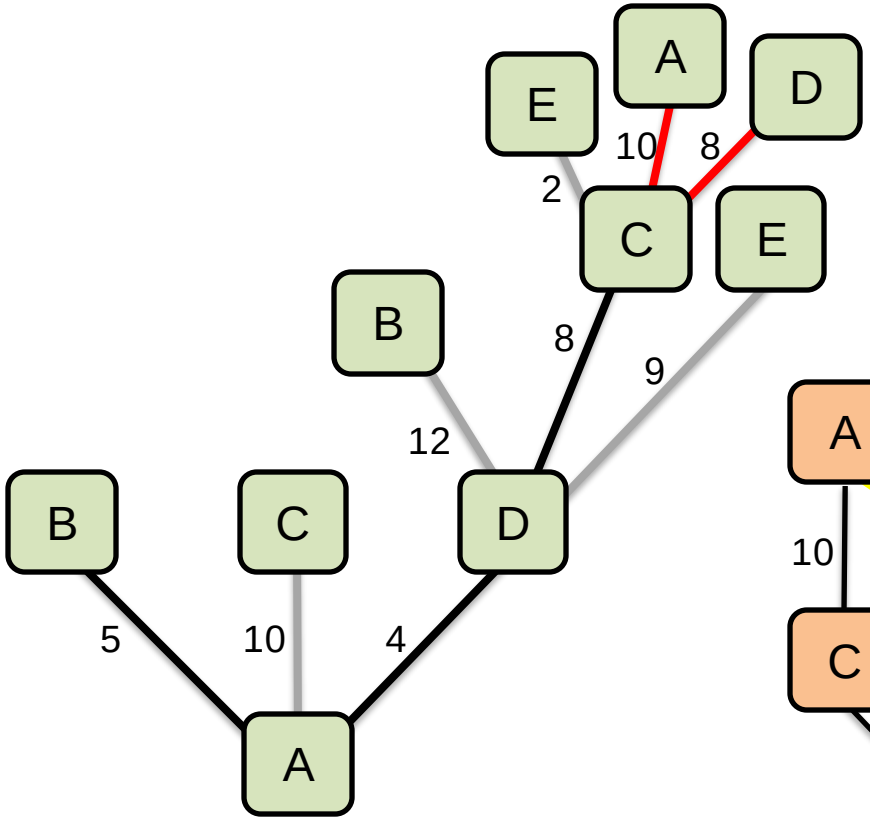
Valor:

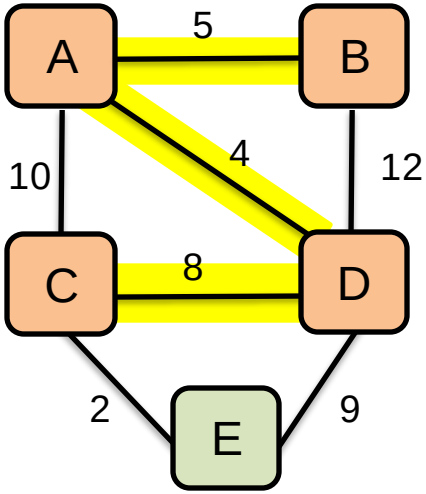
17

PQ



A	X
B	X
C	X
D	X
E	



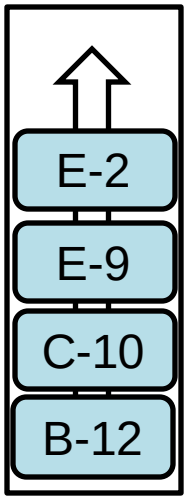


Algoritmo de Prim

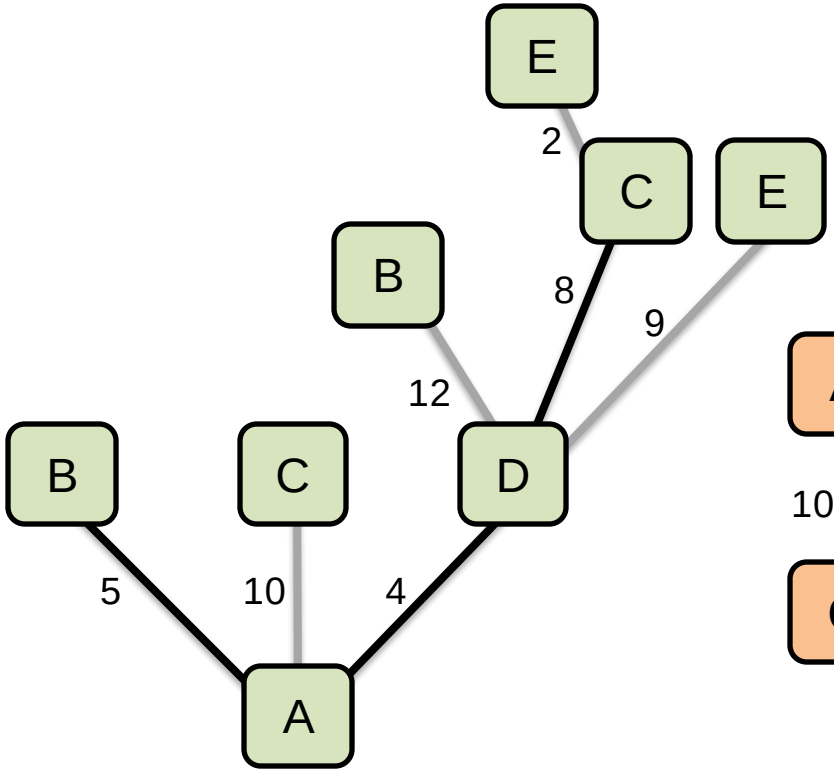
Valor:

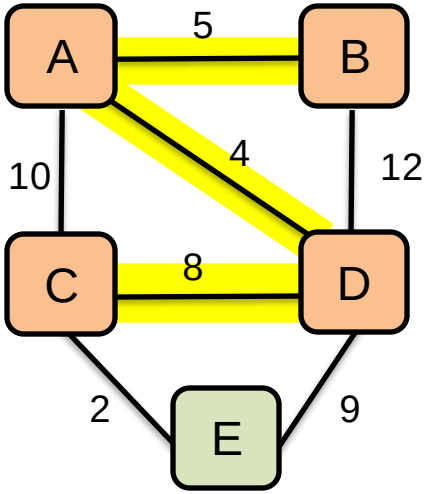
17

PQ



A	X
B	X
C	X
D	X
E	



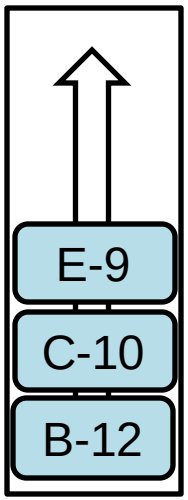


Algoritmo de Prim

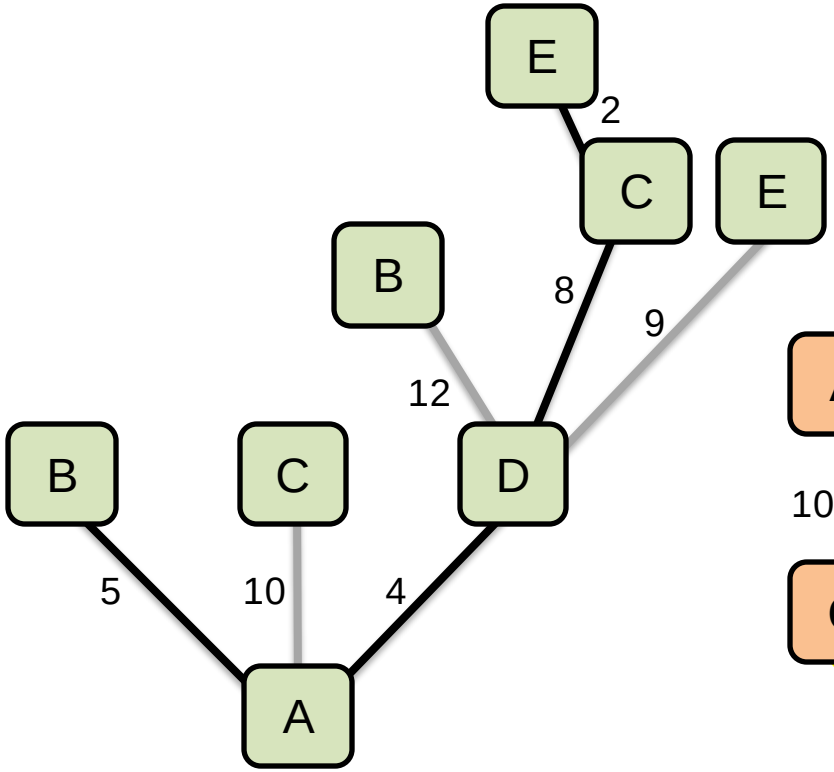
Valor:

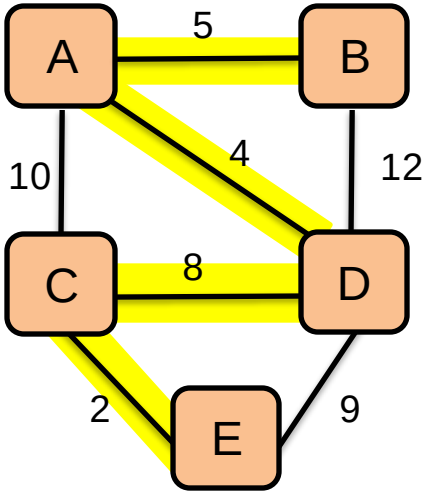
19

PQ



A	X
B	X
C	X
D	X
E	X



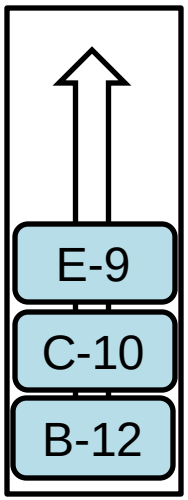


Algoritmo de Prim

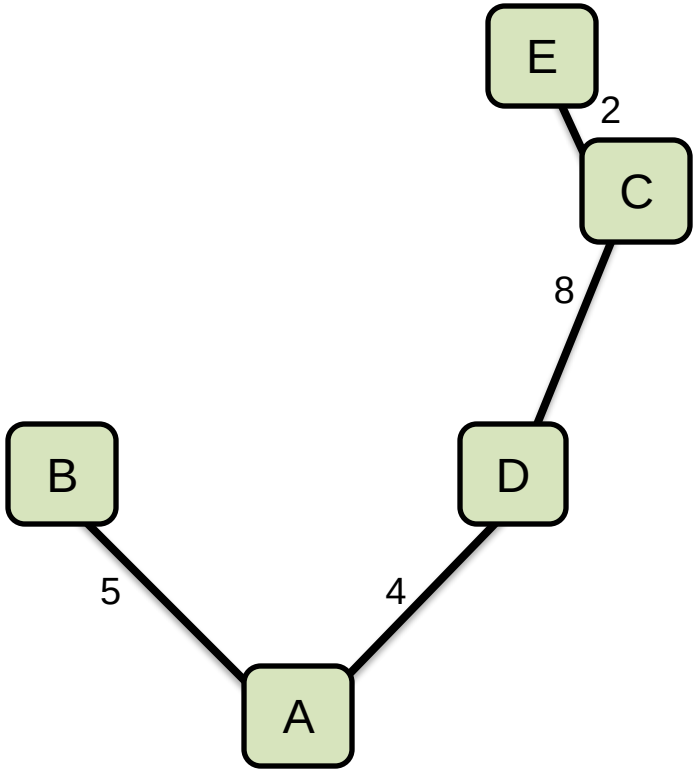
Valor:

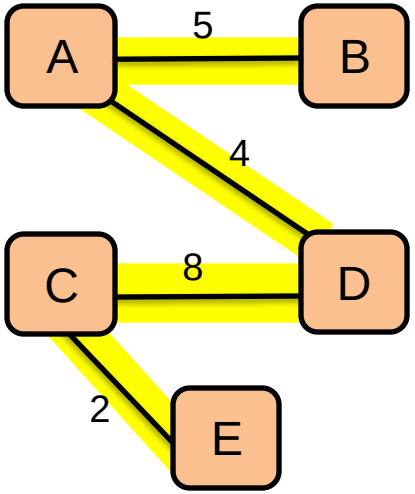
19

PQ



A	X
B	X
C	X
D	X
E	X





Algoritmo de Prim

```
Prim (Grafo G)
      for u en V[G] do
          distancia[u] = INFINITO
          padre[u] = NULL
          Añadir(cola,<u, distancia[u]>)
      distancia[u]=0
      Actualizar(cola,<u, distancia[u]>)
      while !esta_vacia(cola) do
          u = extraer_minimo(cola)
          for v adyacente a 'u' do
              if ((v ∈ cola) && (distancia[v] > peso(u, v))
                  padre[v] = u
                  distancia[v] = peso(u, v)
                  Actualizar(cola,<v, distancia[v]>)
```

Consejo:
Coged un código que funcione y al dosier!!!

https://github.com/stevenhalim/cpbook-code/tree/master/ch4/mst

Ejemplo de grafos

Acepta el reto - De aventuras por el amazonas. Problema: 281

https://onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&category=24&page=show_problem&problem=1927

https://onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&category=24&page=show_problem&problem=2169

https://onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&category=24&page=show_problem&problem=1742

https://onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&category=24&page=show_problem&problem=989

Bloque siguiente...

PROGRAMACIÓN DINÁMICA

¡Hasta la próxima semana!

Ante cualquier duda Telegram o correo:

- Isaac Lozano (isaac.lozano@urjc.es)
- Raúl Martín (raul.martin@urjc.es)
- **Sergio Salazar** (s.salazarc.2018@alumnos.urjc.es)
- Francisco Tórtola (f.tortola.2018@alumnos.urjc.es)
- Cristian Pérez (c.perezc.2018@alumnos.urjc.es)
- Xuqiang Liu (x.liu1.2020@alumnos.urjc.es)
- Alicia Pina (a.pinaz.2020@alumnos.urjc.es)
- Sara García (s.garciarod.2020@alumnos.urjc.es)
- Raúl Fauste (r.fauste.2020@alumnos.urjc.es)