Unidad 6: Grafos

Algoritmos y Estructuras de Datos

Concepto y Clasificación
Implementaciones
Algoritmo de Dijkstra

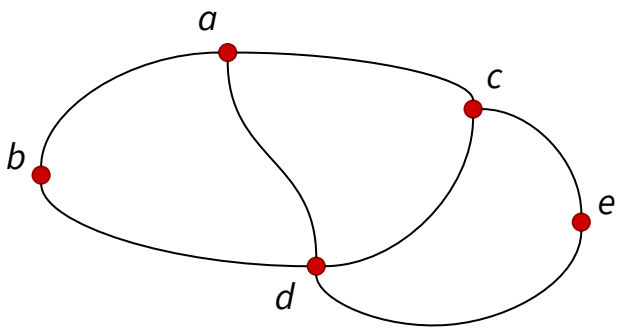
Ing. Juan Ignacio Iturriaga

Grafos

Concepto

Un grafo (*graph*) es una estructura no lineal y no jerárquica que costa **colección de vértices y aristas**, donde:

- Un vértice (*vertex*) o nodo (*node*) representa un valor.
- Un arco (*arc*) o arista (*edge*) es una conexión entre 2 vértices.

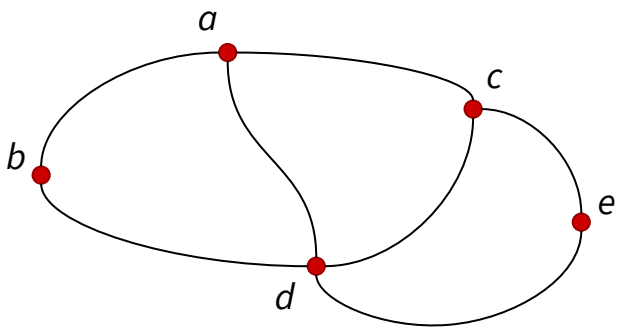


Grafos

Concepto - representación matemática

Un grafo (*graph*) es una estructura no lineal y no jerárquica que costa **colección de vértices y aristas**, donde:

- Un vértice (*vertex*) o nodo (*node*) representa un valor.
- Un arco (*arc*) o arista (*edge*) es una conexión entre 2 vértices.



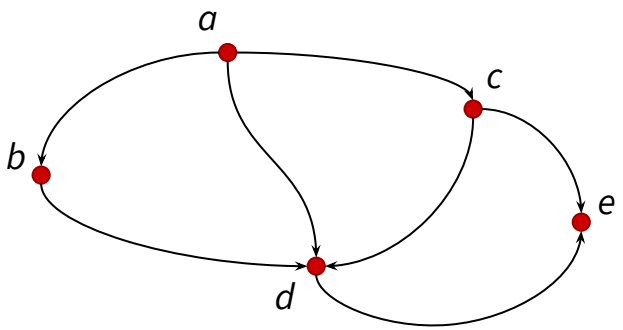
G = (V, A)
V = {a, b, c, d, e}
A = {(a, b), (a, c), (a, d), (b, d), (c, d), (c, e), (d, e)}

Grafos

Concepto - digráfo

Un grafo (*graph*) es una estructura no lineal y no jerárquica que costa **colección de vértices y aristas**, donde:

- Un vértice (*vertex*) o nodo (*node*) representa un valor.
- Un arco (*arc*) o arista (*edge*) es una conexión entre 2 vértices.
- Si los arcos tienen una dirección, es un **grafo dirigido** o digrafo



G = (V, A)
V = {a, b, c, d, e}
A = {(a, b), (a, c), (a, d), (b, d), (c, d), (c, e), (d, e)}

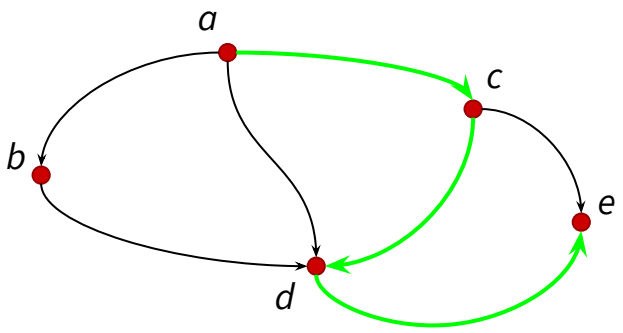
Ahora son pares ordenados, por lo tanto:
(a, b) != (b, a)

Grafos

Concepto - camino

Un grafo (*graph*) es una estructura no lineal y no jerárquica que costa **colección de vértices y aristas**, donde:

- Un vértice (*vertex*) o nodo (*node*) representa un valor.
- Un arco (*arc*) o arista (*edge*) es una conexión entre 2 vértices.
- Si los arcos tienen una dirección, es un **grafo dirigido** o digrafo
- Un **camino** es una secuencia de vértices consecutivos



C = a, c, d, e

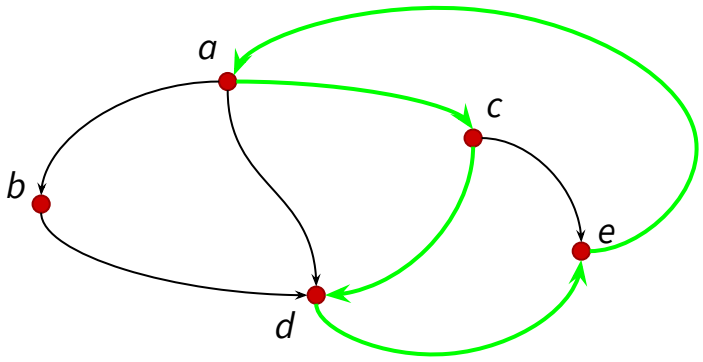
G = (V, A)
V = {a, b, c, d, e}
A = {(a, b), (a, c), (a, d), (b, d), (c, d), (c, e), (d, e)}

Grafos

Concepto - ciclo

Un grafo (*graph*) es una estructura no lineal y no jerárquica que costa **colección de vértices y aristas**, donde:

- Un vértice (*vertex*) o nodo (*node*) representa un valor.
- Un arco (*arc*) o arista (*edge*) es una conexión entre 2 vértices.
- Si los arcos tienen una dirección, es un **grafo dirigido** o digrafo
- Un **camino** es una secuencia de vértices consecutivos
- Cuando existe un camino que retorna al mismo vértice, entonces el grafo es **cíclico**



C = a, c, d, e, a

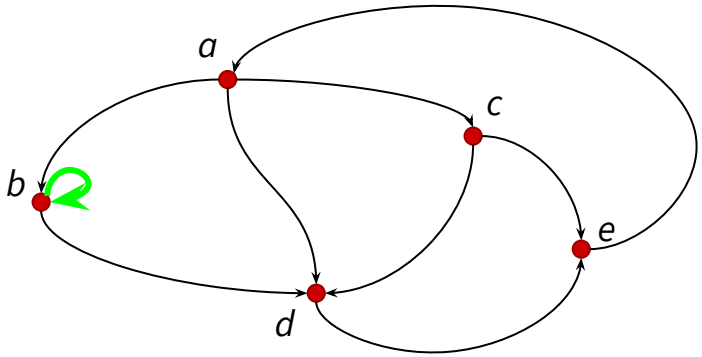
G = (V, A)
V = {a, b, c, d, e}
A = {(a, b), (a, c), (a, d), (b, d), (c, d), (c, e), (d, e), (e, a)}

Grafos

Concepto - ciclo

Un grafo (*graph*) es una estructura no lineal y no jerárquica que costa **colección de vértices y aristas**, donde:

- Un vértice (*vertex*) o nodo (*node*) representa un valor.
- Un arco (*arc*) o arista (*edge*) es una conexión entre 2 vértices.
- Si los arcos tienen una dirección, es un **grafo dirigido** o digrafo
- Un **camino** es una secuencia de vértices consecutivos
- Cuando existe un camino que retorna al mismo vértice, entonces el grafo es **cíclico**
- Un vértice también puede estar relacionado con sí mismo.



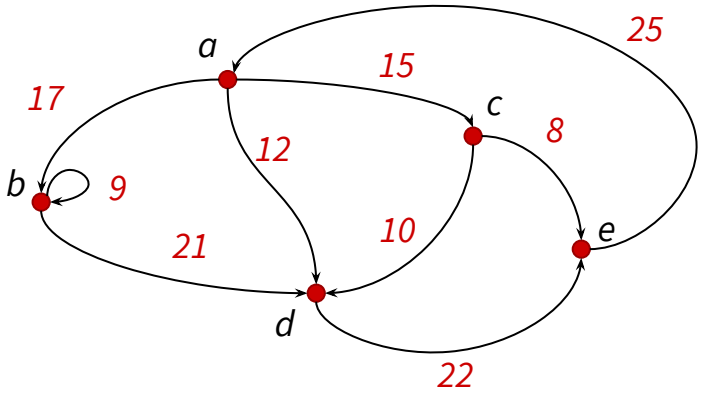
G = (V, A)
V = {a, b, c, d, e}
A = {(a, b), (a, c), (a, d), (b, d), (c, d), (c, e), (d, e), (e, a), (b, b)}

Grafos

Concepto - grafos ponderados

Un grafo (*graph*) es una estructura no lineal y no jerárquica que costa **colección de vértices y aristas**, donde:

- Un vértice (*vertex*) o nodo (*node*) representa un valor.
- Un arco (*arc*) o arista (*edge*) es una conexión entre 2 vértices.
- Si los arcos tienen una dirección, es un **grafo dirigido** o digrafo
- Un **camino** es una secuencia de vértices consecutivos
- Cuando existe un camino que retorna al mismo vértice, entonces el grafo es **cíclico**
- Un vértice también puede estar relacionado con sí mismo.
- Un grafo es ponderado cuando las aristas tienen información adicional.



G = (V, A)
V = {a, b, c, d, e}
A = {(a, b, 17), (a, c, 15), (a, d, 12), (b, d, 21),
(c, d, 10), (c, e, 8), (d, e, 22), (e, a, 25), (b, b, 9)}

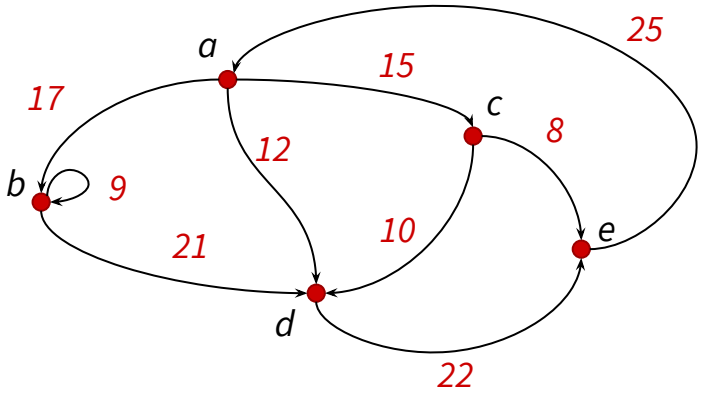
Grafos - Implementaciones

Matriz de adyacencia

Matriz de adyacencia

Adjacency Matrix

	0	1	2	3	4
a = 0	0	17	15	12	0
b = 1	0	9	0	21	0
c = 2	0	0	0	10	8
d = 3	0	0	0	0	22
e = 4	25	0	0	0	0



G = (V, A)

V = {a, b, c, d, e}

A = {(a, b, 17), (a, c, 15), (a, d, 12), (b, d, 21),
(c, d, 10), (c, e, 8), (d, e, 22), (e, a, 25), (b, b, 9)}

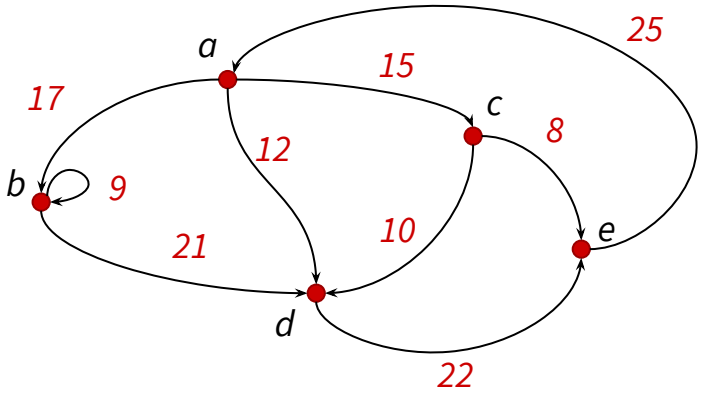
Grafos - Implementaciones

Matriz de adyacencia

Matriz de adyacencia

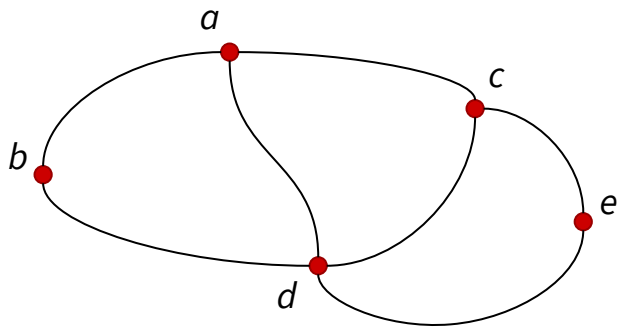
Adjacency Matrix

	0	1	2	3	4
a = 0	0	17	15	12	0
b = 1	0	9	0	21	0
c = 2	0	0	0	10	8
d = 3	0	0	0	0	22
e = 4	25	0	0	0	0



G = (V, A)
V = {a, b, c, d, e}
A = {(a, b, 17), (a, c, 15), (a, d, 12), (b, d, 21),
(c, d, 10), (c, e, 8), (d, e, 22), (e, a, 25), (b, b, 9)}

	0	1	2	3	4
a = 0	0	1	1	1	0
b = 1	1	0	0	1	0
c = 2	1	0	0	1	1
d = 3	1	1	1	0	1
e = 4	0	0	1	1	0



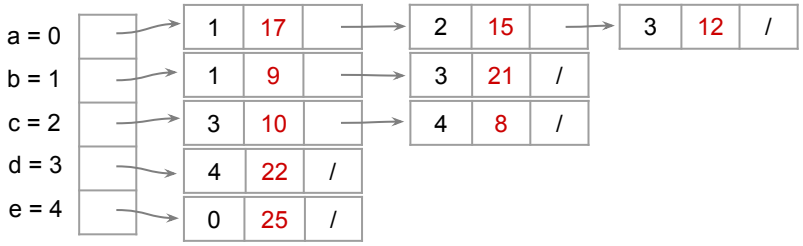
G = (V, A)
V = {a, b, c, d, e}
A = {(a, b), (a, c), (a, d), (b, d), (c, d), (c, e), (d, e)}

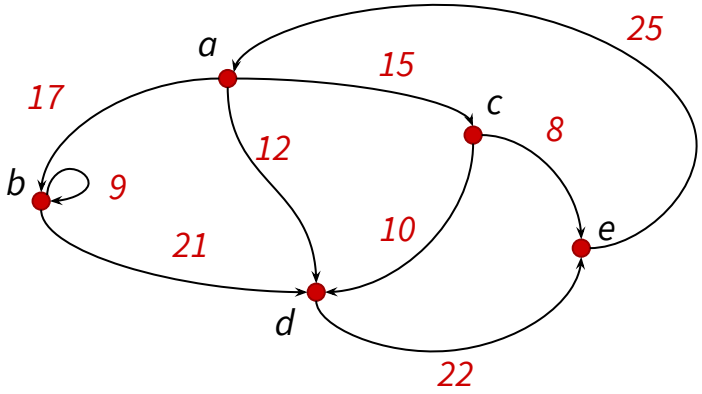
Grafos - Implementaciones

Listas de adyacencia

Listas de adyacencia

Adjacency List





G = (V, A)
V = {a, b, c, d, e}
A = {(a, b, 17), (a, c, 15), (a, d, 12), (b, d, 21),
(c, d, 10), (c, e, 8), (d, e, 22), (e, a, 25), (b, b, 9)}

Grafos - Implementaciones

Una Implementación en C (*Matriz de adyacencia)*

Definición de tipos

```
typedef struct Graph graph;
typedef struct GraphVertex vertex;

typedef struct GraphVertex {
    int index;
    char *value;
} vertex;

typedef struct Graph {
    vertex *V[MAX];
    int A[MAX][MAX];
    int size;
} graph;
```

Grafos - Implementaciones

Una Implementación en C (*Matriz de adyacencia)*

Definición de tipos

```
typedef struct Graph graph;
typedef struct GraphVertex vertex;

typedef struct GraphVertex {
    int index;
    char *value;
} vertex;

typedef struct Graph {
    vertex *V[MAX];
    int A[MAX][MAX];
    int size;
} graph;
```

Creación de vértices y grafos

```
vertex *createVertex(char *value) {
    vertex *v = (vertex
*)malloc(sizeof(vertex));
    v->index = -1;
    v->value = value;
    return v;
}

graph *createGraph() {
    graph *g = (graph
*)malloc(sizeof(graph));
    g->size = 0;
}
```

Grafos - Implementaciones

Una Implementación en C (*Matriz de adyacencia)*

Agregar vértice

```
int graphAddVertex(graph *g, vertex *v) {
    if (!g) return -1;
    if (!v) return -1;
    int result = -1;
    if ((g->size + 1) < MAX) {
        v->index = g->size;
        g->V[g->size++] = v;
        for (int i = 0; i < g->size; i++) {
            g->A[v->index][i] = 0;
            g->A[i][v->index] = 0;
        }
        result = v->index;
    }
    return result;
}

int graphAddNewVertex(graph *g, char *value) {
    vertex *v = createVertex(value);
    return graphAddVertex(g, v);
}
```

Definición de tipos

```
typedef struct Graph graph;
typedef struct GraphVertex vertex;

typedef struct GraphVertex {
    int index;
    char *value;
} vertex;

typedef struct Graph {
    vertex *V[MAX];
    int A[MAX][MAX];
    int size;
} graph;
```

Creación de vértices y grafos

```
vertex *createVertex(char *value) {
    vertex *v = (vertex
*)malloc(sizeof(vertex));
    v->index = -1;
    v->value = value;
    return v;
}

graph *createGraph() {
    graph *g = (graph
*)malloc(sizeof(graph));
    g->size = 0;
}
```

Grafos - Implementaciones

Una Implementación en C (*Matriz de adyacencia)*

Agregar vértice

```
int graphAddVertex(graph *g, vertex *v) {
    if (!g) return -1;
    if (!v) return -1;
    int result = -1;
    if ((g->size + 1) < MAX) {
        v->index = g->size;
        g->V[g->size++] = v;
        for (int i = 0; i < g->size; i++) {
            g->A[v->index][i] = 0;
            g->A[i][v->index] = 0;
        }
        result = v->index;
    }
    return result;
}

int graphAddNewVertex(graph *g, char *value) {
    vertex *v = createVertex(value);
    return graphAddVertex(g, v);
}
```

Agregar Arista

```
int graphSetArc(graph *g, int fromVertex, int toVertex,
int cost) {
    if (!g) return 0;
    if (fromVertex >= g->size) return 0;
    if (toVertex >= g->size) return 0;

    g->A[fromVertex][toVertex] = cost;
    return 1;
}
```

Definición de tipos

```
typedef struct Graph graph;
typedef struct GraphVertex vertex;

typedef struct GraphVertex {
    int index;
    char *value;
} vertex;

typedef struct Graph {
    vertex *V[MAX];
    int A[MAX][MAX];
    int size;
} graph;
```

Creación de vértices y grafos

```
vertex *createVertex(char *value) {
    vertex *v = (vertex
*)malloc(sizeof(vertex));
    v->index = -1;
    v->value = value;
    return v;
}

graph *createGraph() {
    graph *g = (graph
*)malloc(sizeof(graph));
    g->size = 0;
}
```

Grafos - Implementaciones

Una Implementación en C (*Matriz de adyacencia)*

Imprimir el grafo

```
void graphPrint(graph *g) {
    if (!g) return;

    for (int i = 0; i < g->size; i++) {
        printf("%4d - %10s : ", i, g->V[i]->value);
        for (int j = 0; j < g->size; j++) {
            printf(" %4d", g->A[i][j]);
        }
        printf("\n");
    }
}
```

Definición de tipos

```
typedef struct Graph graph;
typedef struct GraphVertex vertex;

typedef struct GraphVertex {
    int index;
    char *value;
} vertex;

typedef struct Graph {
    vertex *V[MAX];
    int A[MAX][MAX];
    int size;
} graph;
```

Creación de vértices y grafos

```
vertex *createVertex(char *value) {
    vertex *v = (vertex
*)malloc(sizeof(vertex));
    v->index = -1;
    v->value = value;
    return v;
}

graph *createGraph() {
    graph *g = (graph
*)malloc(sizeof(graph));
    g->size = 0;
}
```

Grafos - camino más corto

Algoritmo de Dijkstra

Grafos - camino más corto

Algoritmo de Dijkstra

Grafos - camino más corto

Algoritmo de Dijkstra

Grafos - camino más corto

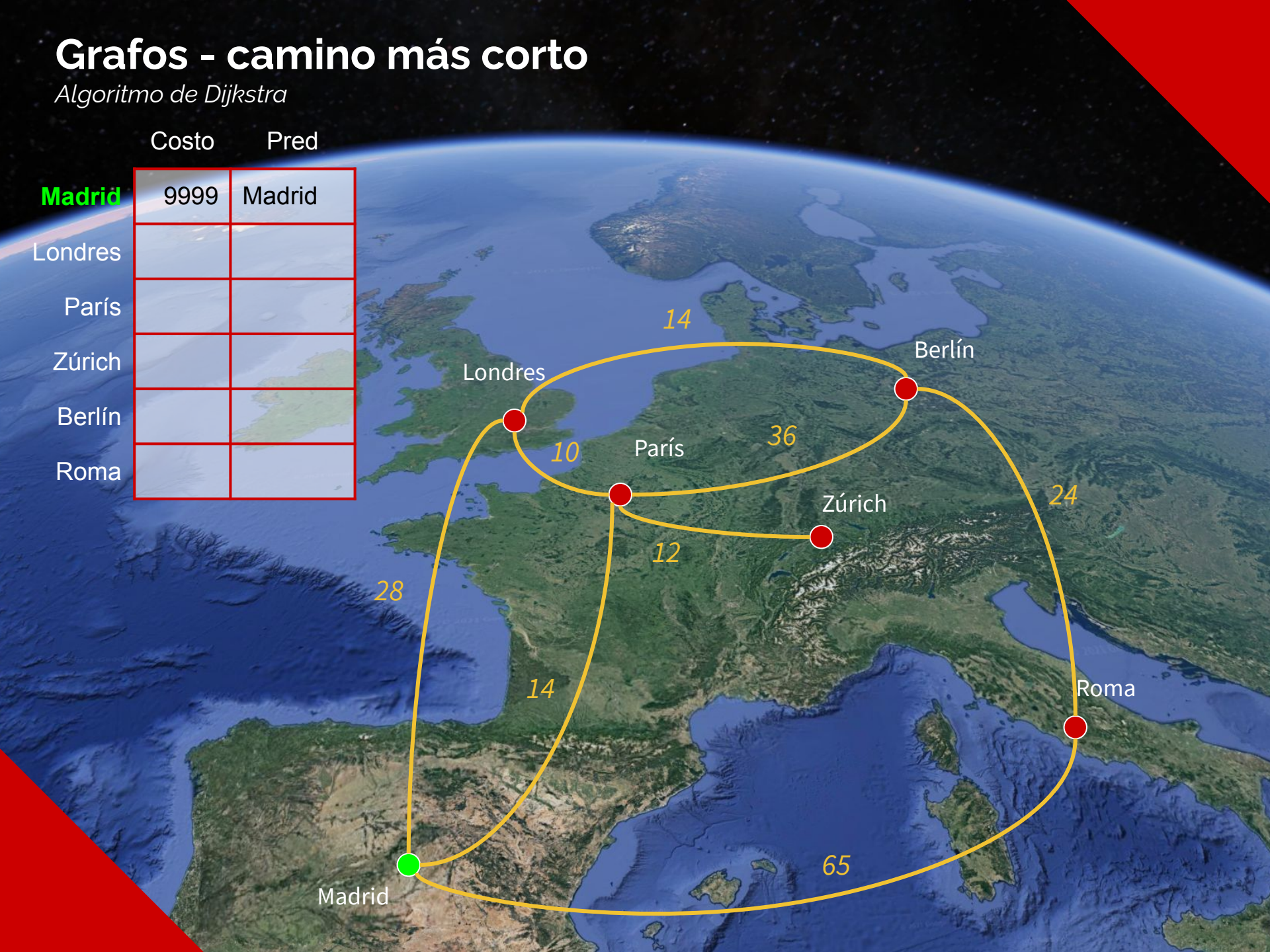
Algoritmo de Dijkstra

	Costo	Pred
Madrid		
Londres		
París		
Zúrich		
Berlín		
Roma		

Grafos - camino más corto

Algoritmo de Dijkstra

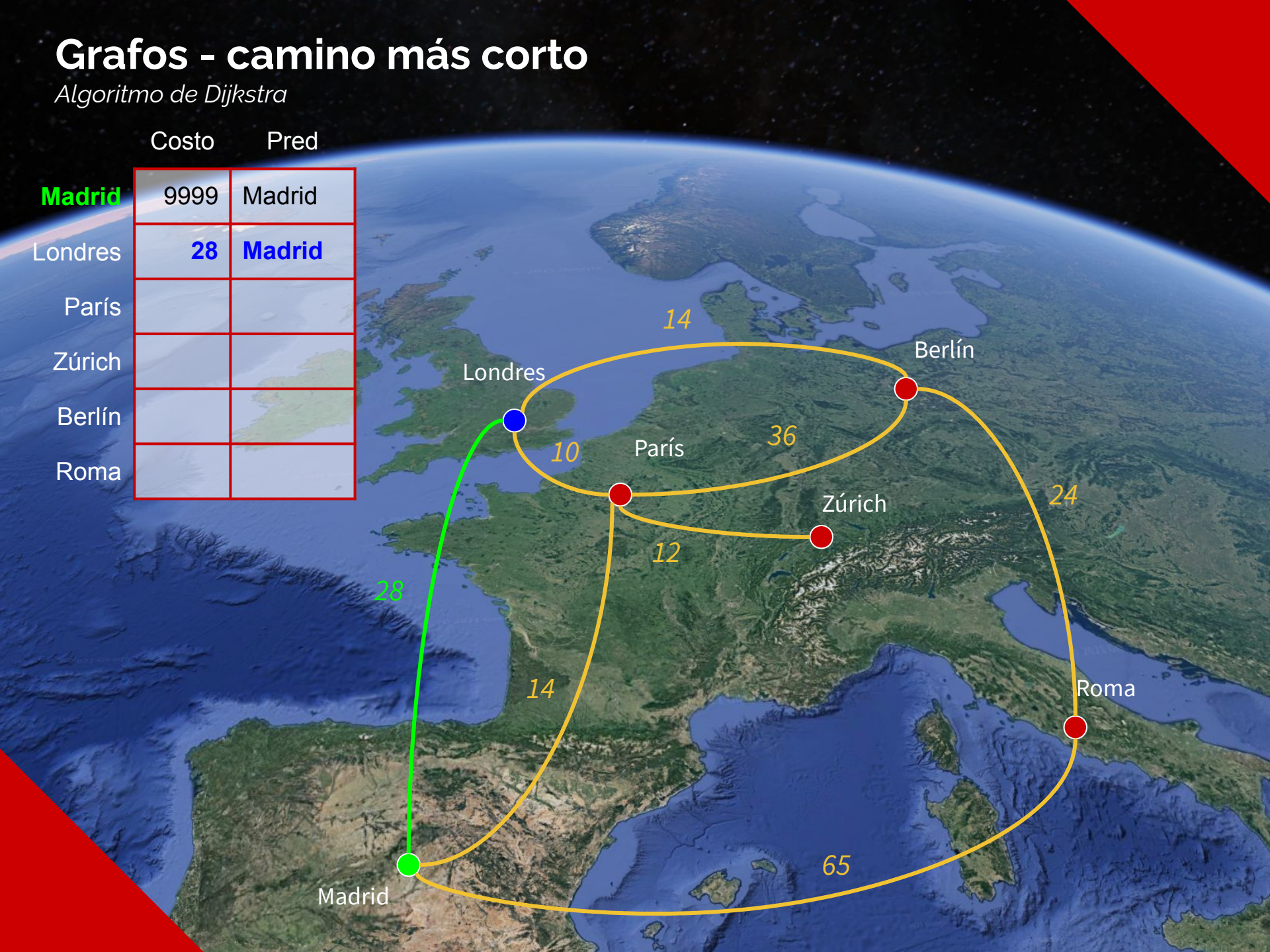
	Costo	Pred
Madrid	9999	Madrid
Londres		
París		
Zúrich		
Berlín		
Roma		



Grafos - camino más corto

Algoritmo de Dijkstra

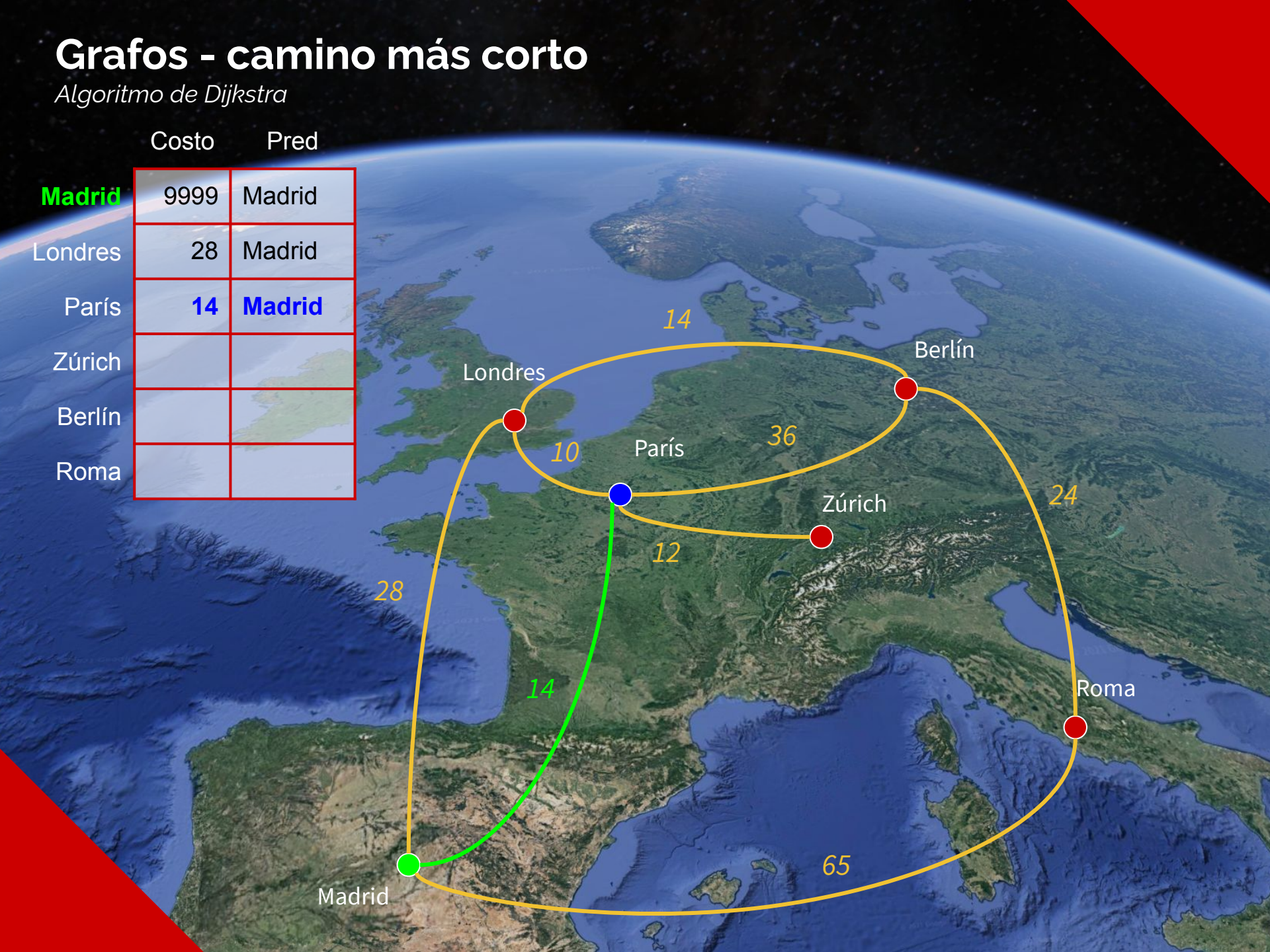
	Costo	Pred
Madrid	9999	Madrid
Londres	**28**	**Madrid**
París		
Zúrich		
Berlín		
Roma		



Grafos - camino más corto

Algoritmo de Dijkstra

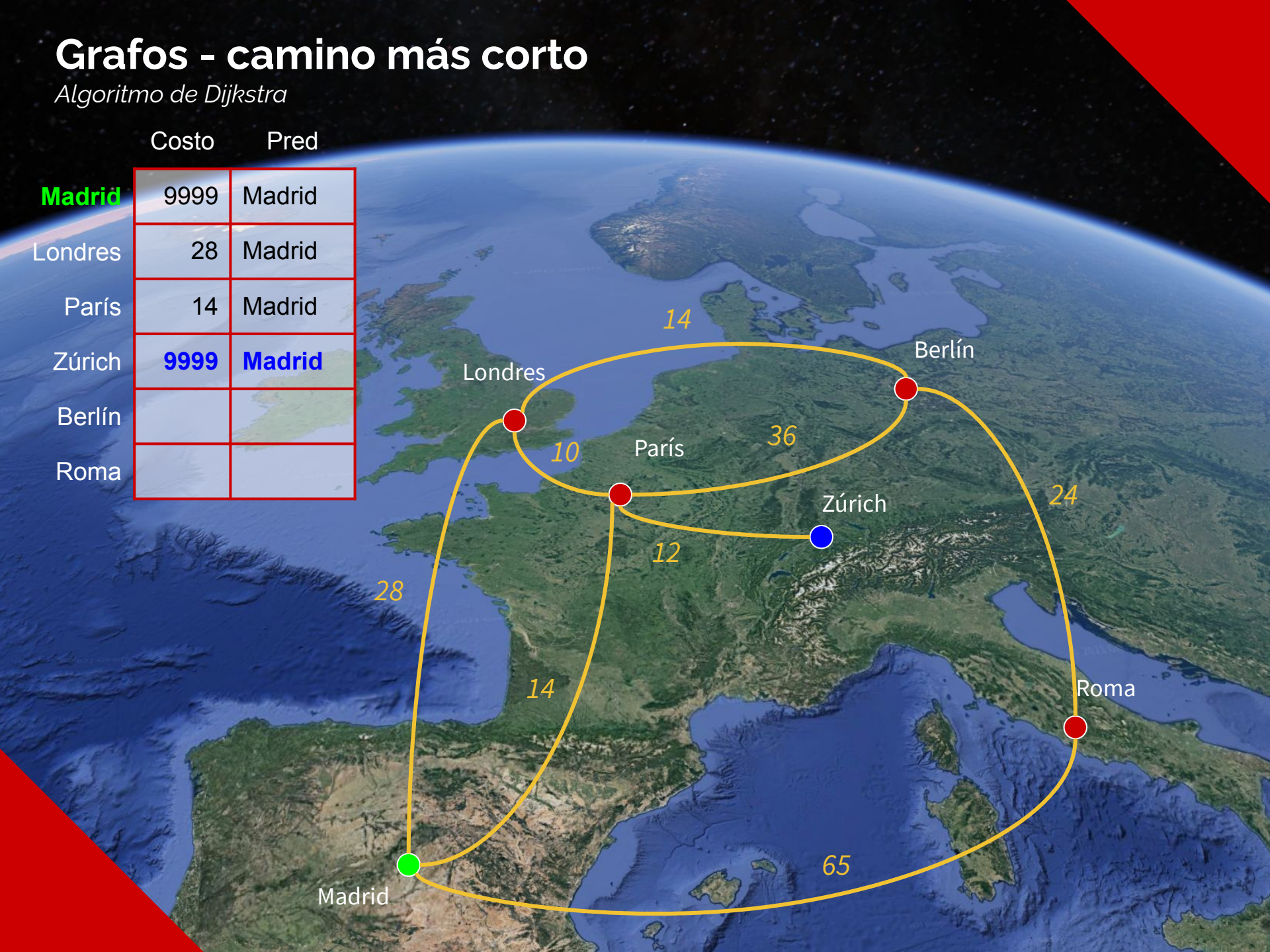
	Costo	Pred
Madrid	9999	Madrid
Londres	28	Madrid
París	**14**	**Madrid**
Zúrich		
Berlín		
Roma		



Grafos - camino más corto

Algoritmo de Dijkstra

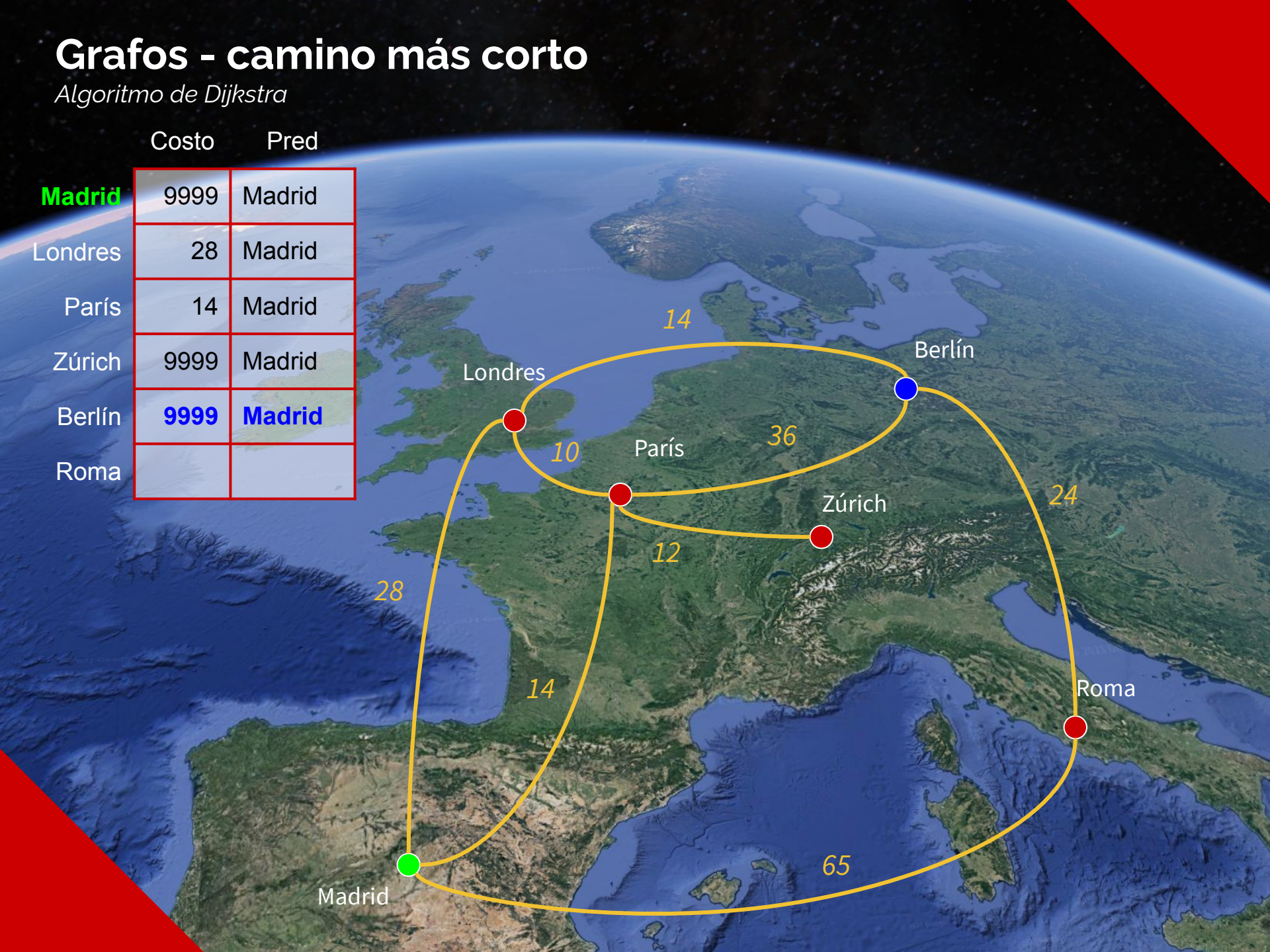
	Costo	Pred
Madrid	9999	Madrid
Londres	28	Madrid
París	14	Madrid
Zúrich	**9999**	**Madrid**
Berlín		
Roma		



Grafos - camino más corto

Algoritmo de Dijkstra

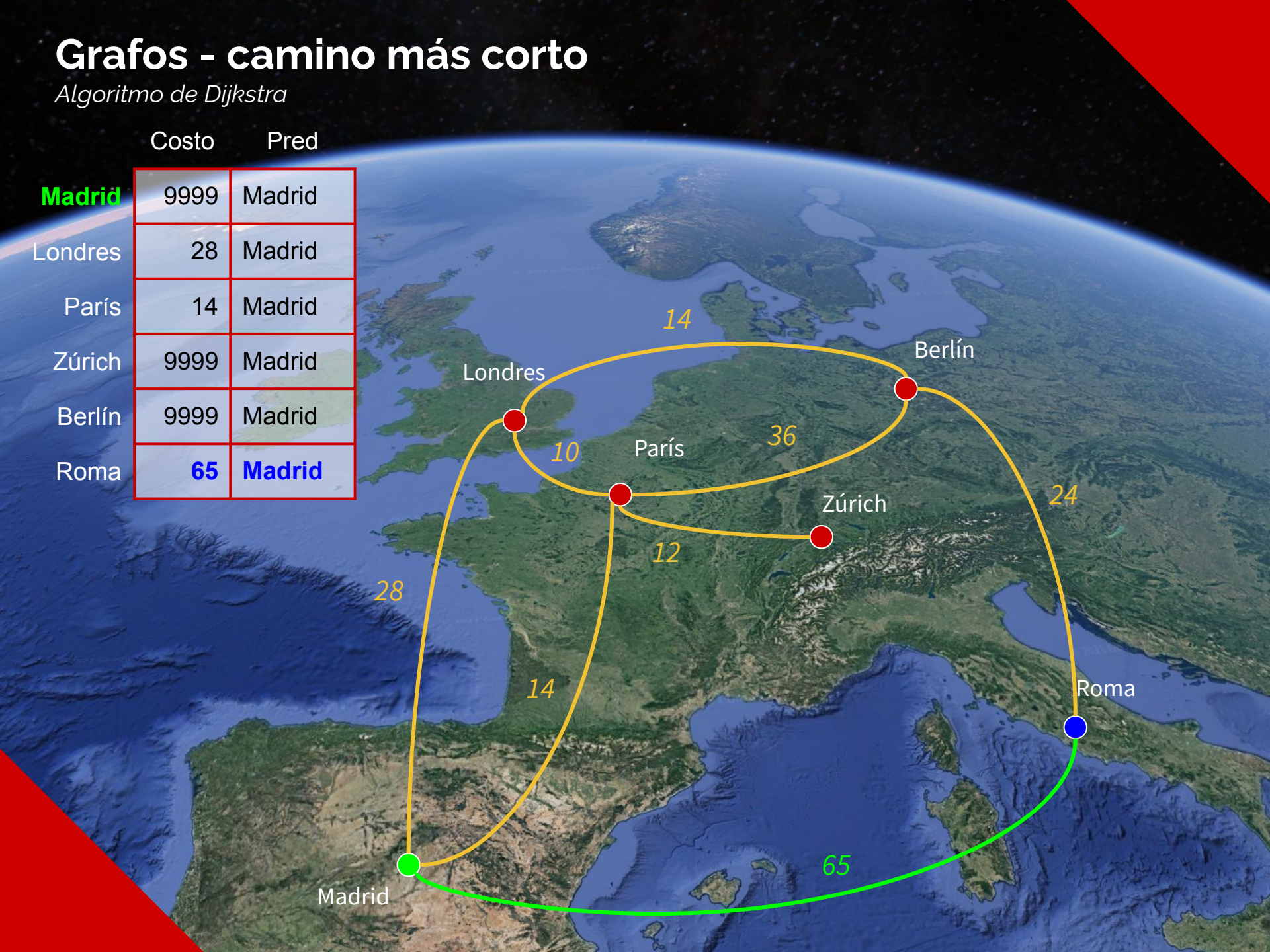
	Costo	Pred
Madrid	9999	Madrid
Londres	28	Madrid
París	14	Madrid
Zúrich	9999	Madrid
Berlín	**9999**	**Madrid**
Roma		



Grafos - camino más corto

Algoritmo de Dijkstra

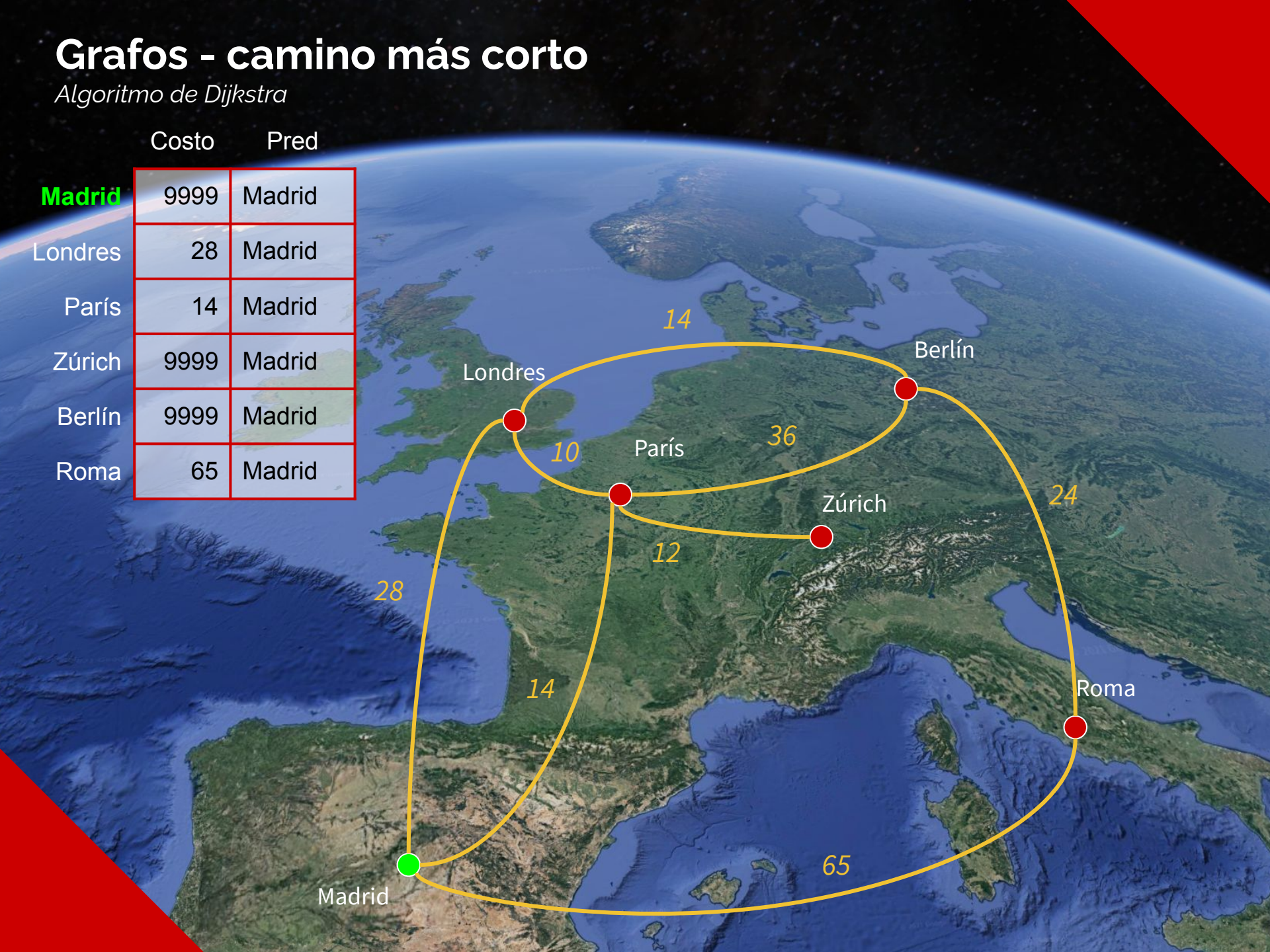
	Costo	Pred
Madrid	9999	Madrid
Londres	28	Madrid
París	14	Madrid
Zúrich	9999	Madrid
Berlín	9999	Madrid
Roma	**65**	**Madrid**



Grafos - camino más corto

Algoritmo de Dijkstra

	Costo	Pred
Madrid	9999	Madrid
Londres	28	Madrid
París	14	Madrid
Zúrich	9999	Madrid
Berlín	9999	Madrid
Roma	65	Madrid

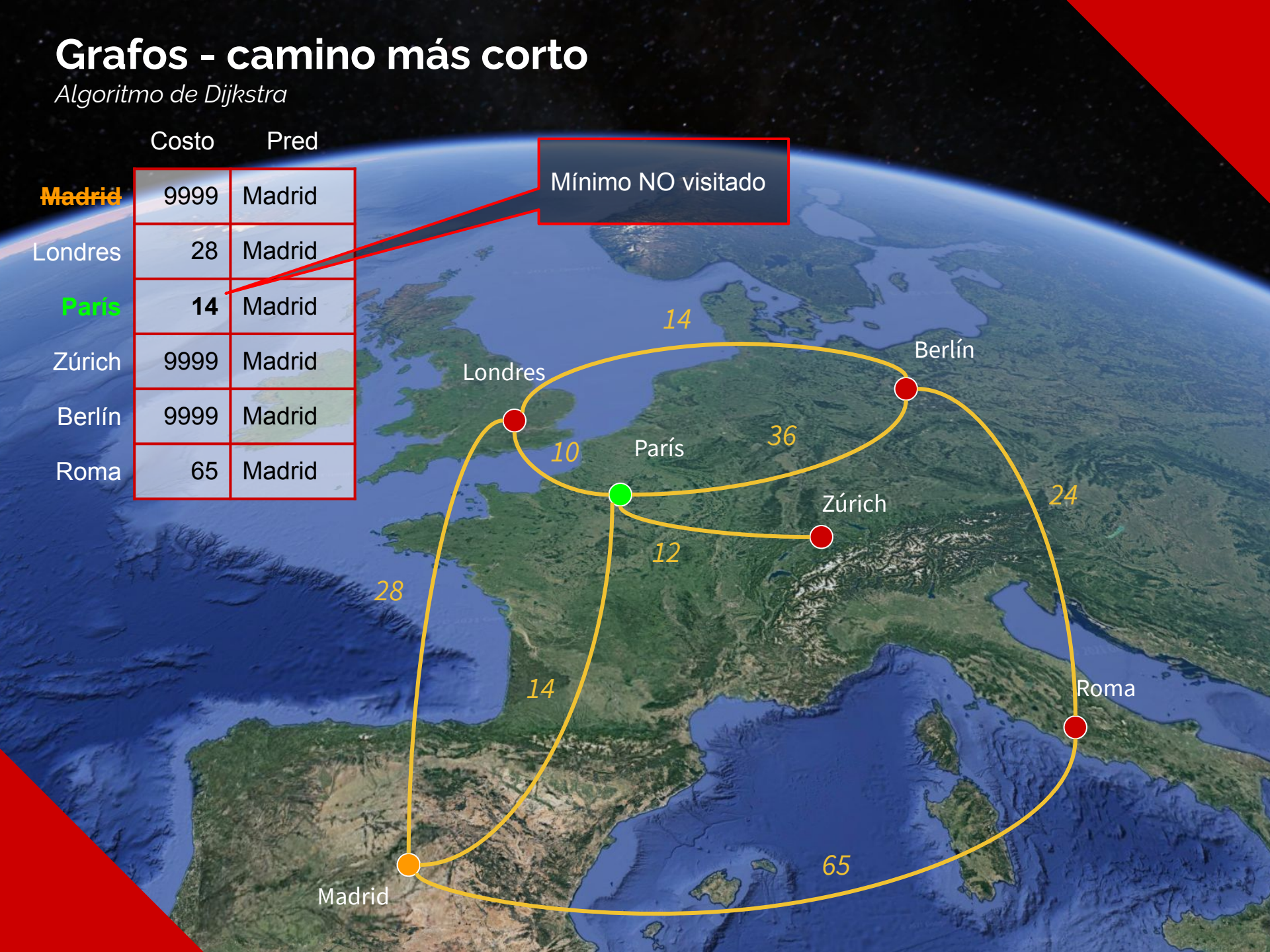


Grafos - camino más corto

Algoritmo de Dijkstra

	Costo	Pred
~~Madrid~~	9999	Madrid
Londres	28	Madrid
París	**14**	Madrid
Zúrich	9999	Madrid
Berlín	9999	Madrid
Roma	65	Madrid

Mínimo NO visitado

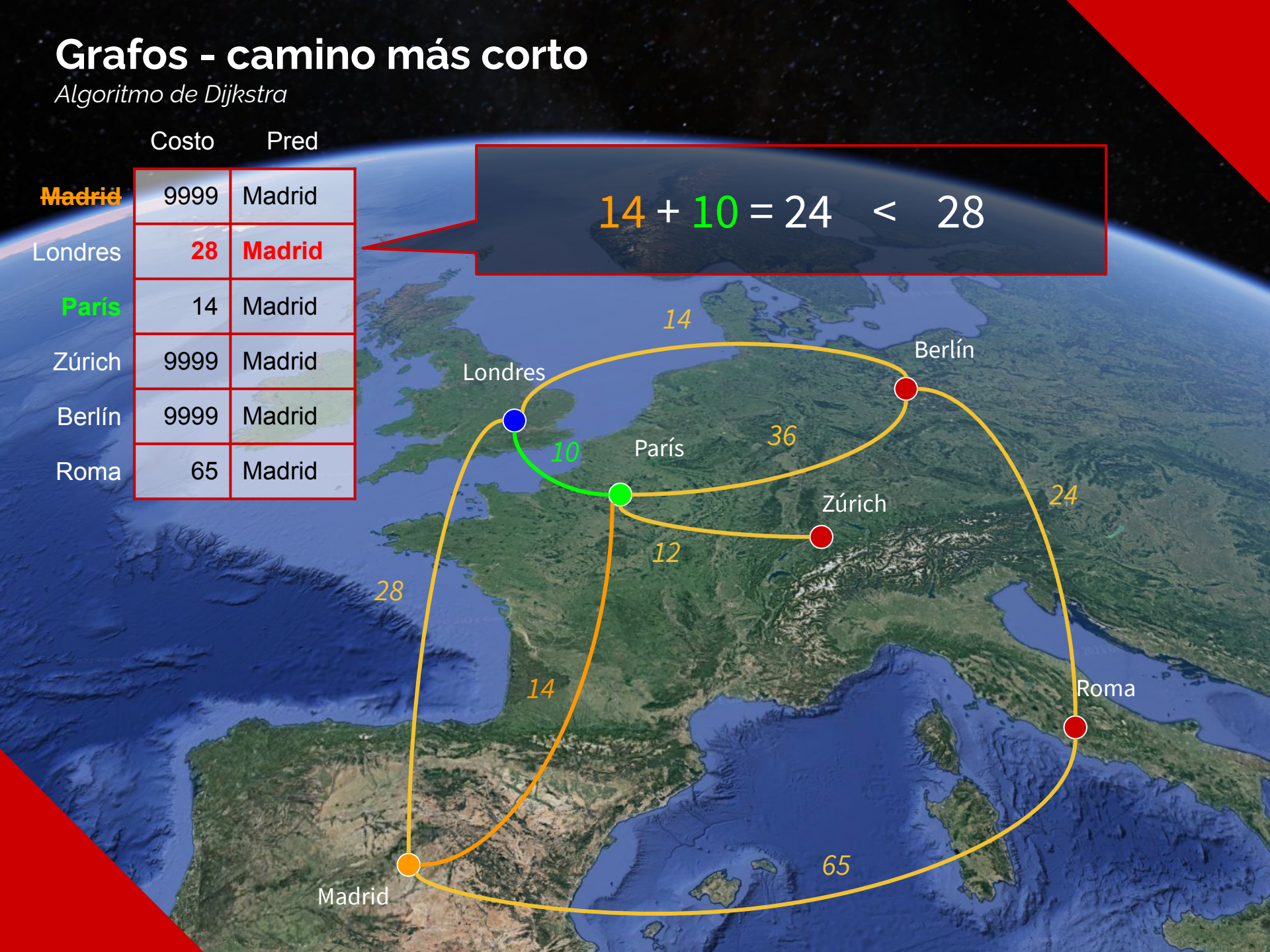


Grafos - camino más corto

Algoritmo de Dijkstra

	Costo	Pred
~~Madrid~~	9999	Madrid
Londres	28	Madrid
París	14	Madrid
Zúrich	9999	Madrid
Berlín	9999	Madrid
Roma	65	Madrid

$14 + 10 = 24 \quad < \quad 28$

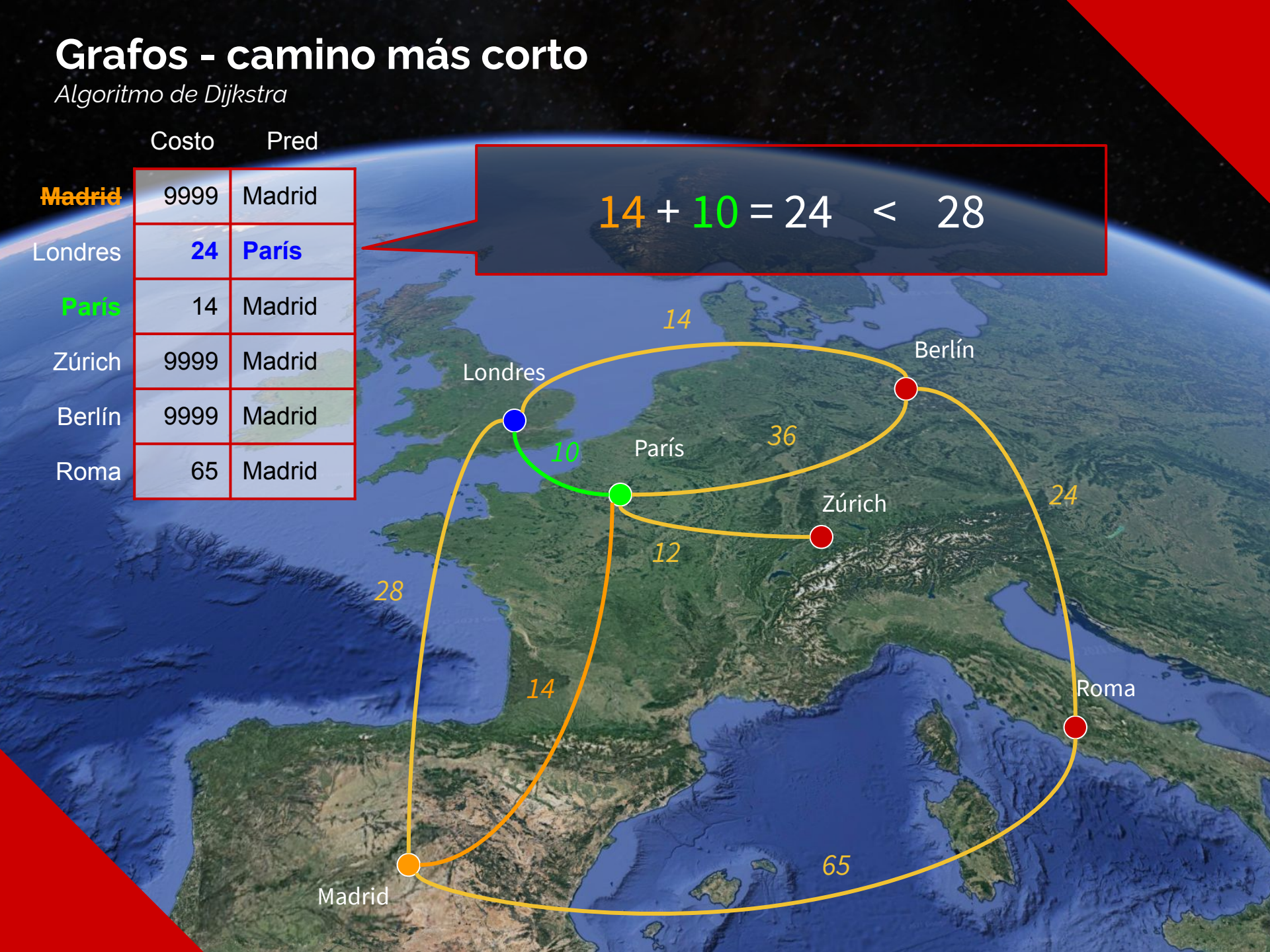


Grafos - camino más corto

Algoritmo de Dijkstra

	Costo	Pred
~~Madrid~~	9999	Madrid
Londres	**24**	**París**
París	14	Madrid
Zúrich	9999	Madrid
Berlín	9999	Madrid
Roma	65	Madrid

$14 + 10 = 24 \quad < \quad 28$

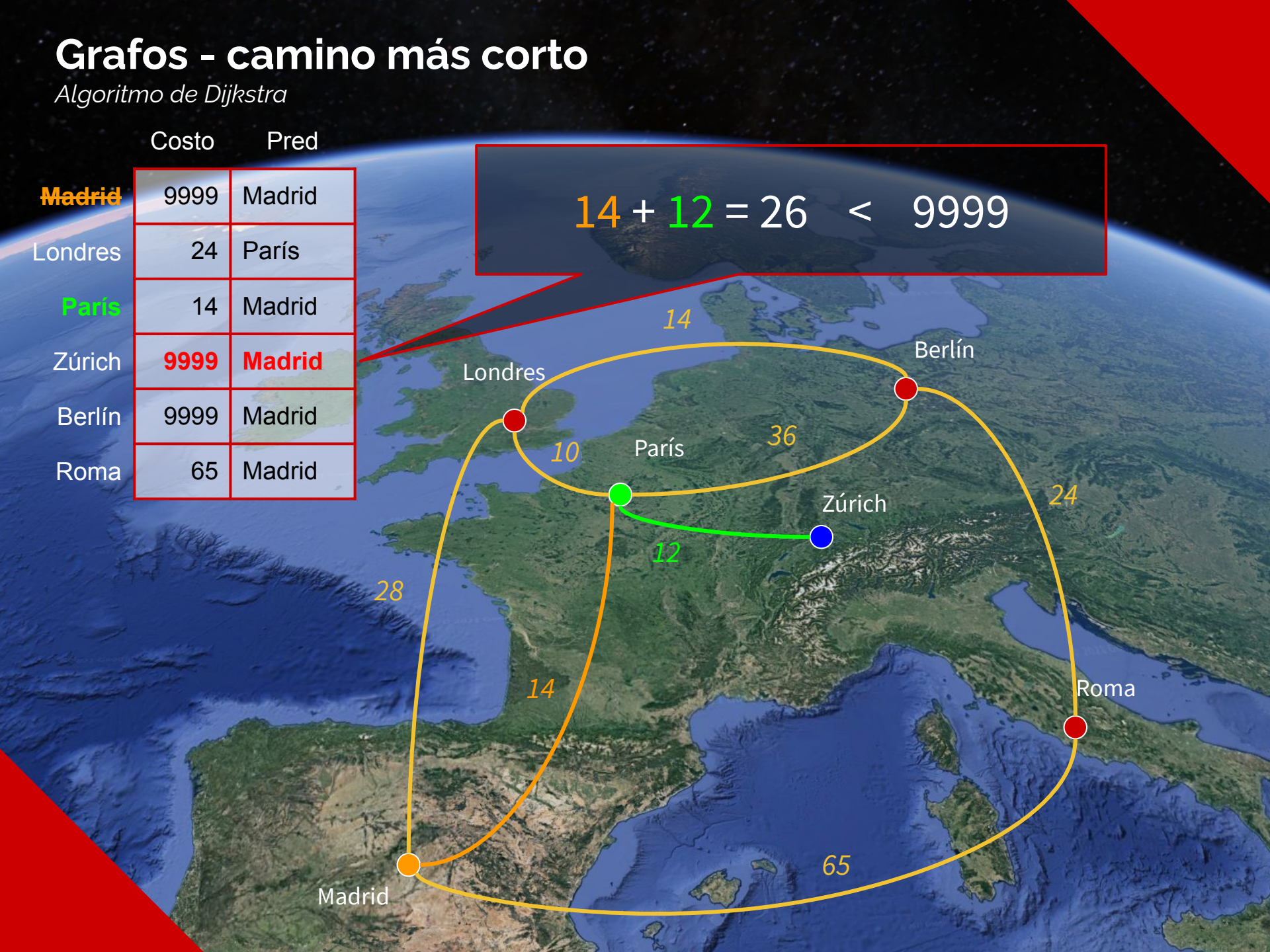


Grafos - camino más corto

Algoritmo de Dijkstra

	Costo	Pred
~~Madrid~~	9999	Madrid
Londres	24	París
París	14	Madrid
Zúrich	9999	Madrid
Berlín	9999	Madrid
Roma	65	Madrid

$14 + 12 = 26 \quad < \quad 9999$

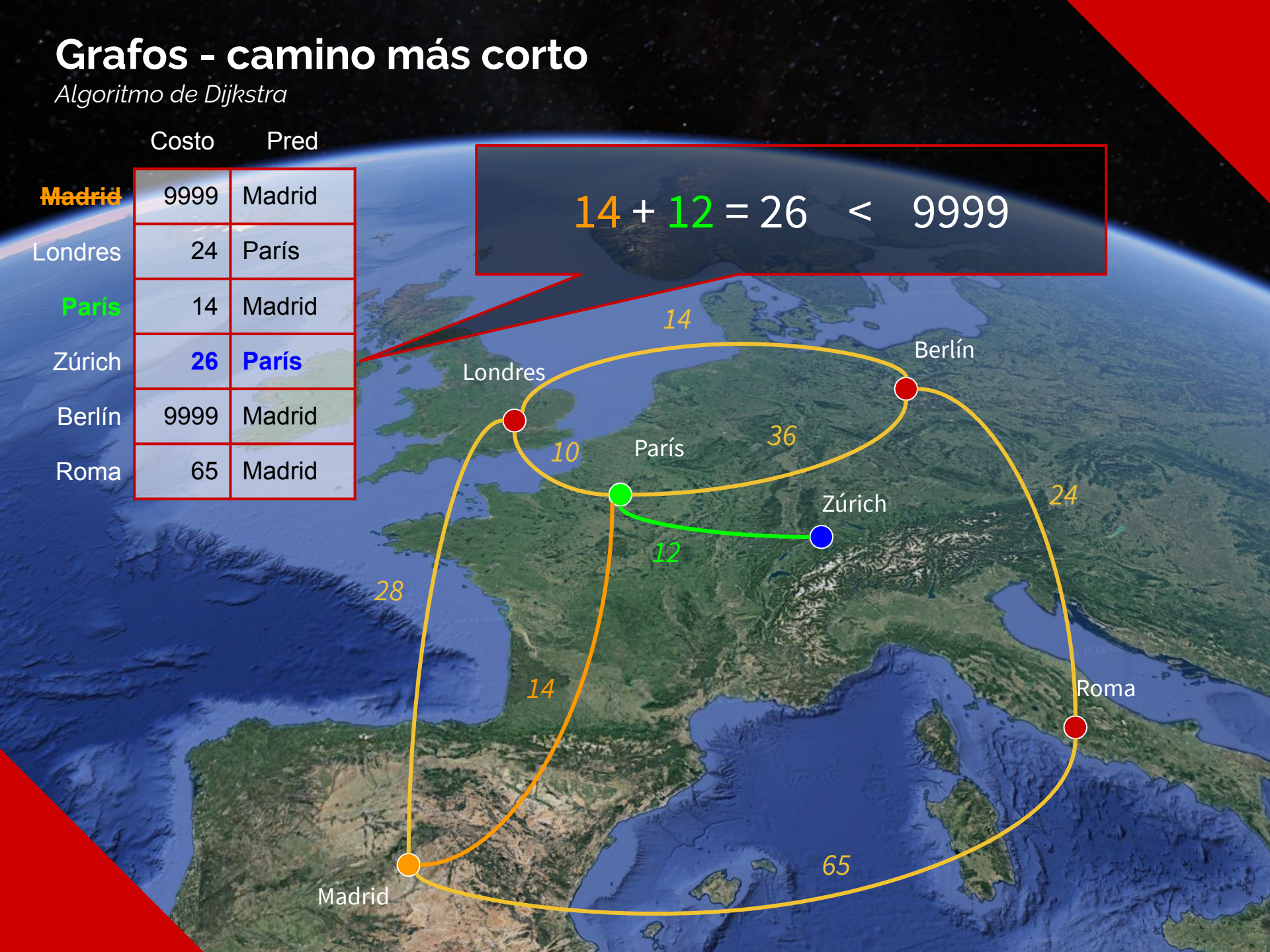


Grafos - camino más corto

Algoritmo de Dijkstra

	Costo	Pred
~~Madrid~~	9999	Madrid
Londres	24	París
París	14	Madrid
Zúrich	**26**	**París**
Berlín	9999	Madrid
Roma	65	Madrid

$14 + 12 = 26 \quad < \quad 9999$

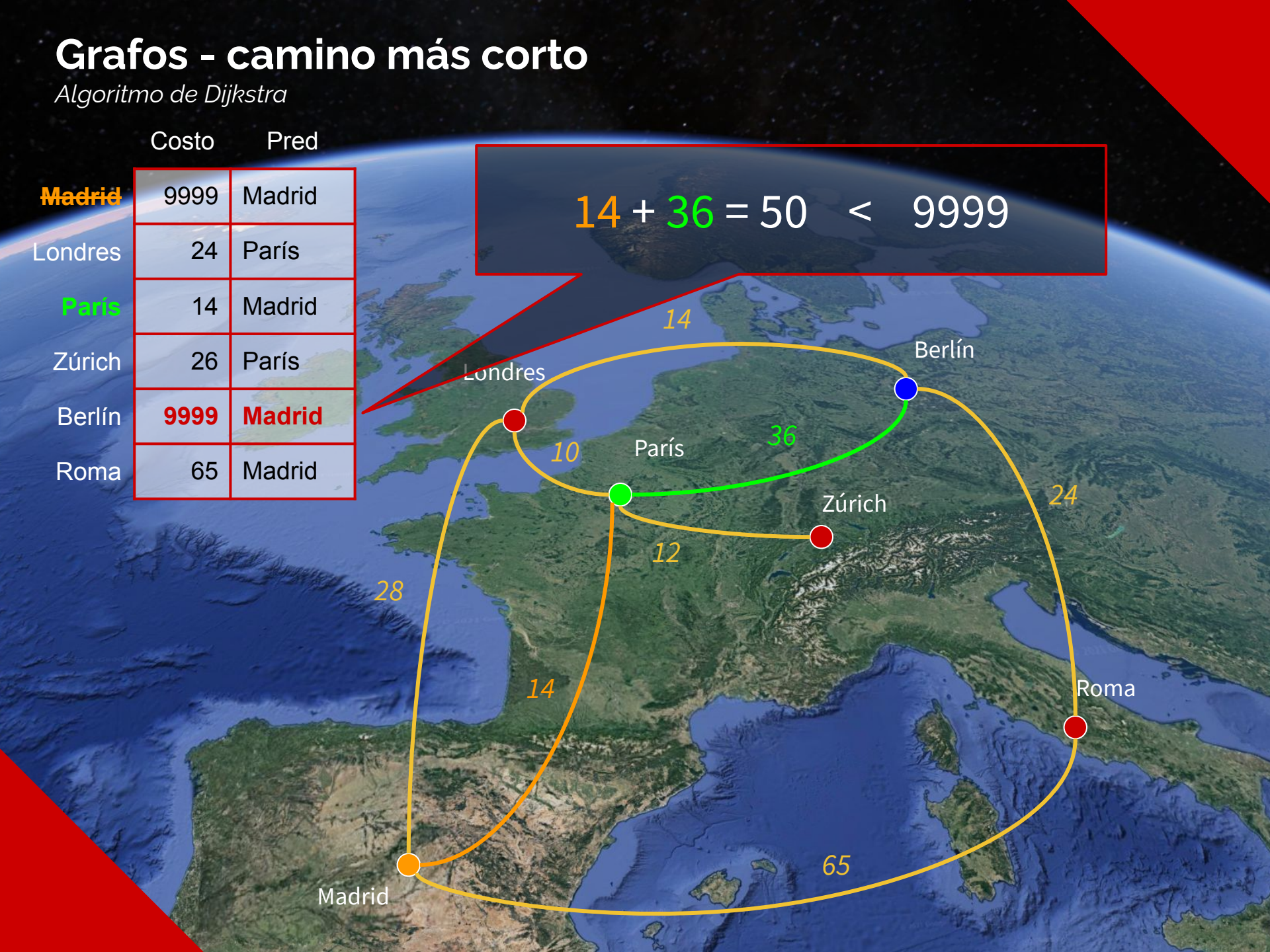


Grafos - camino más corto

Algoritmo de Dijkstra

	Costo	Pred
~~Madrid~~	9999	Madrid
Londres	24	París
París	14	Madrid
Zúrich	26	París
Berlín	**9999**	**Madrid**
Roma	65	Madrid

$$14 + 36 = 50 \quad < \quad 9999$$

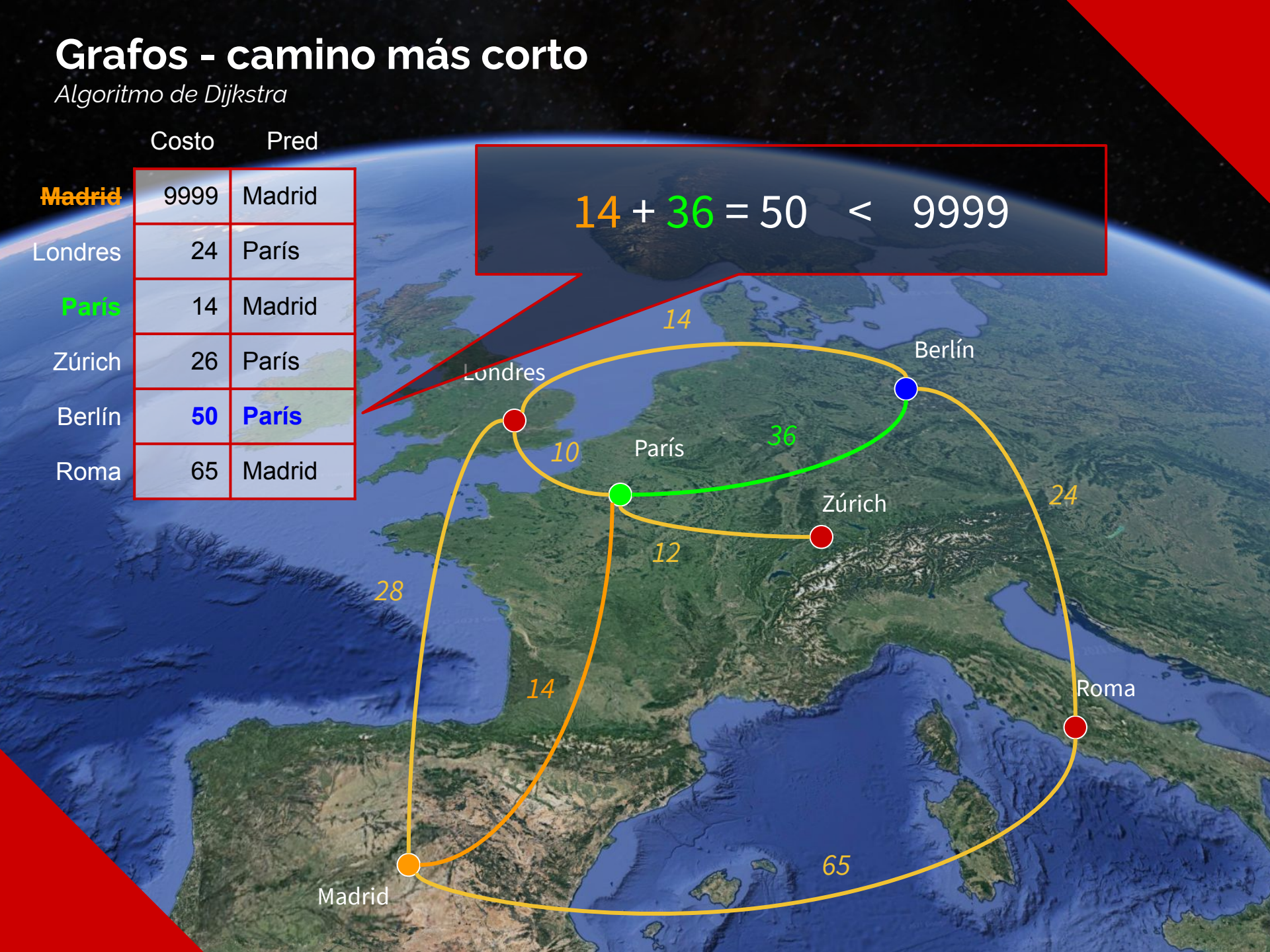


Grafos - camino más corto

Algoritmo de Dijkstra

	Costo	Pred
~~Madrid~~	9999	Madrid
Londres	24	París
París	14	Madrid
Zúrich	26	París
Berlín	**50**	**París**
Roma	65	Madrid

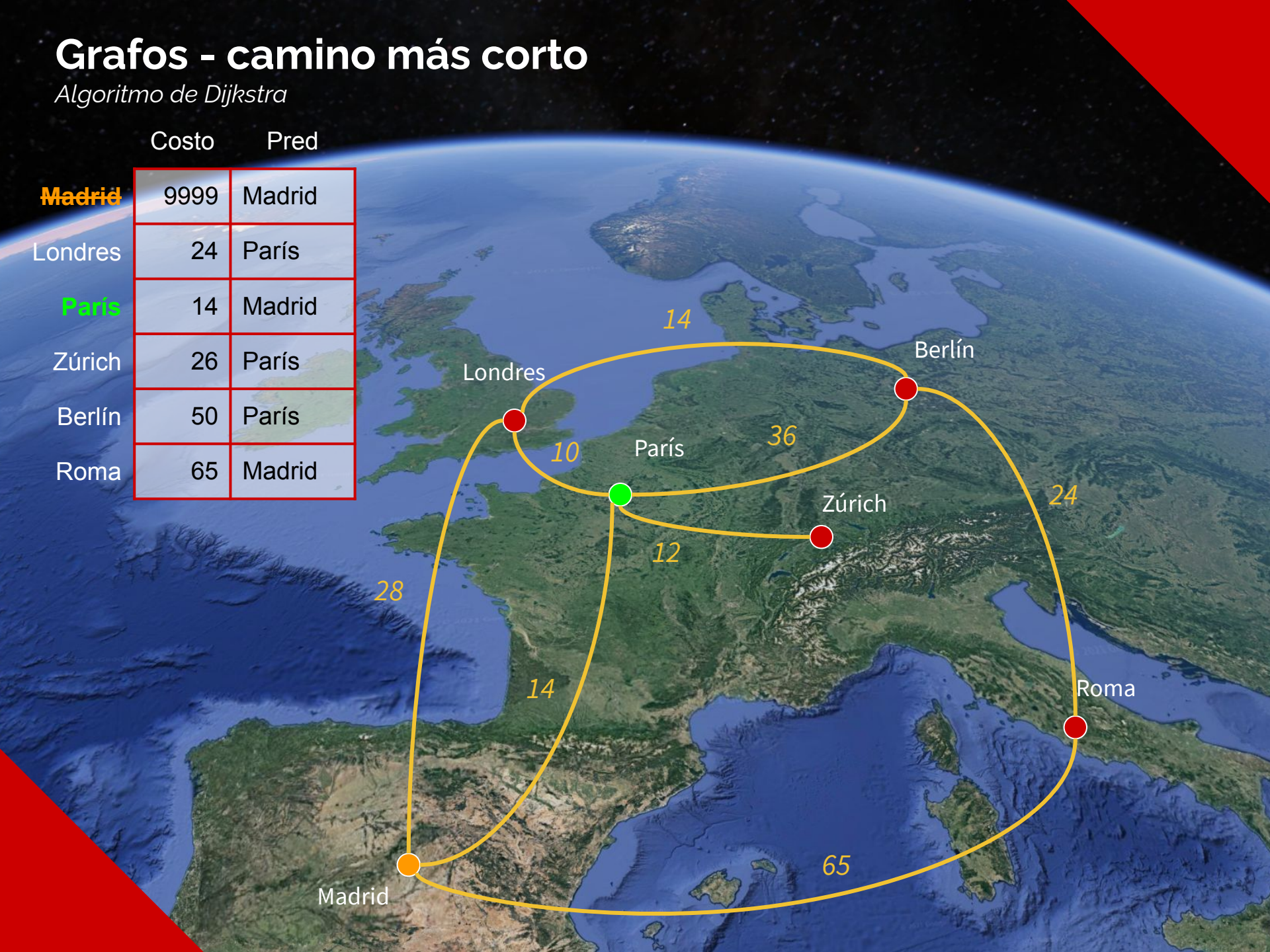
$$14 + 36 = 50 \quad < \quad 9999$$



Grafos - camino más corto

Algoritmo de Dijkstra

	Costo	Pred
~~Madrid~~	9999	Madrid
Londres	24	París
París	14	Madrid
Zúrich	26	París
Berlín	50	París
Roma	65	Madrid

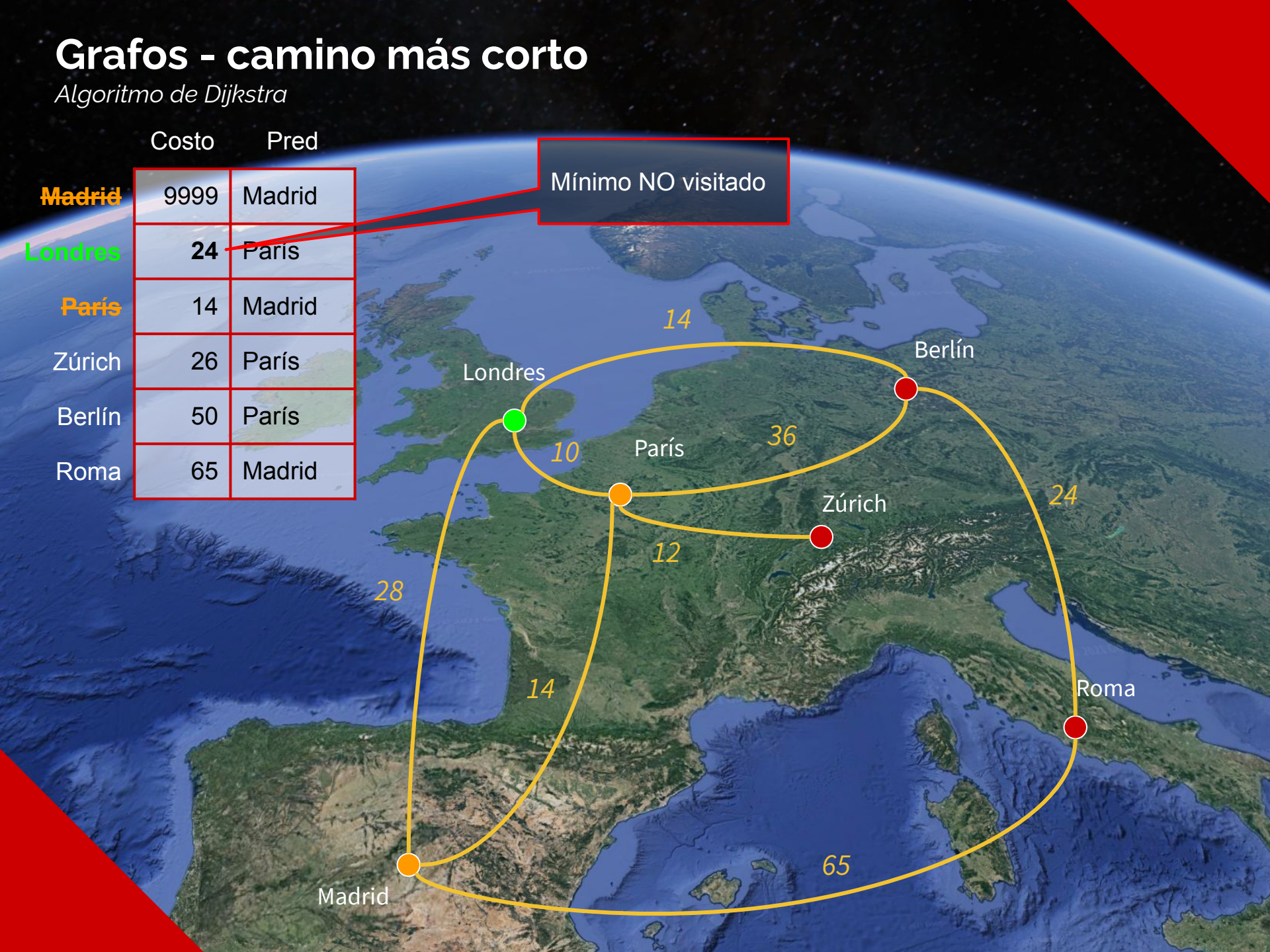


Grafos - camino más corto

Algoritmo de Dijkstra

	Costo	Pred
~~Madrid~~	9999	Madrid
Londres	**24**	París
~~París~~	14	Madrid
Zúrich	26	París
Berlín	50	París
Roma	65	Madrid

Mínimo NO visitado

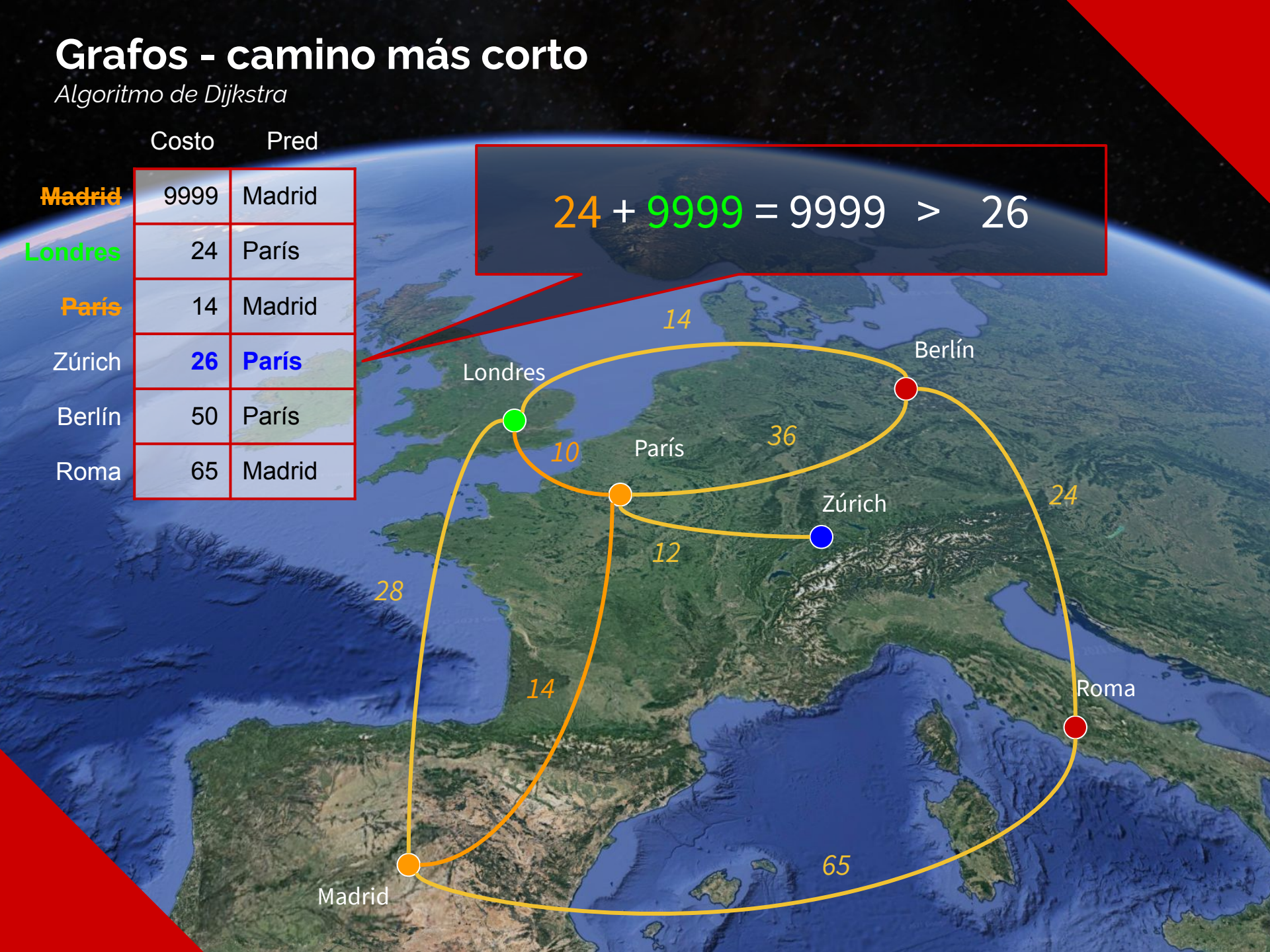


Grafos - camino más corto

Algoritmo de Dijkstra

	Costo	Pred
~~Madrid~~	9999	Madrid
Londres	24	París
~~París~~	14	Madrid
Zúrich	**26**	**París**
Berlín	50	París
Roma	65	Madrid

$$24 + 9999 = 9999 > 26$$

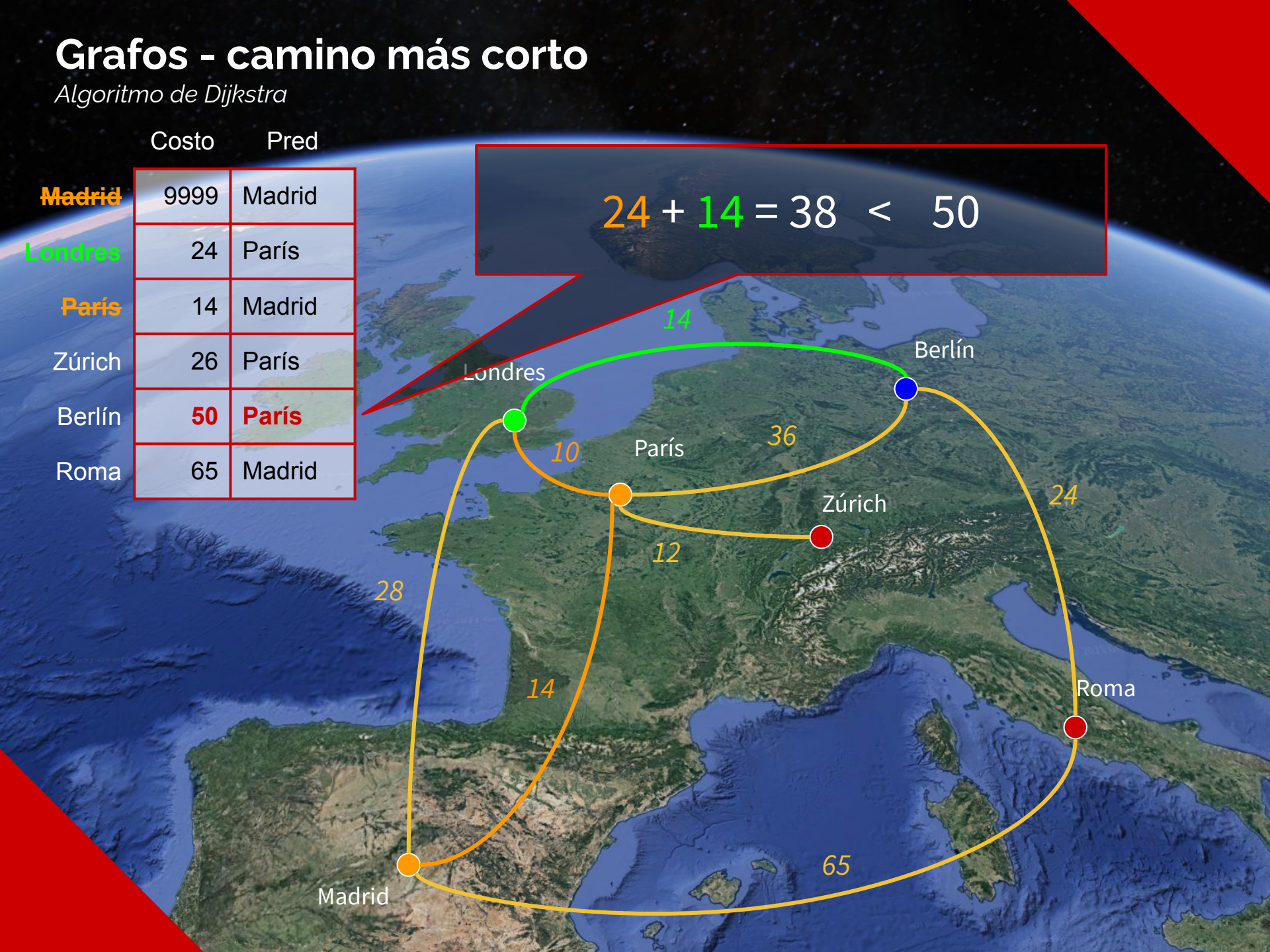


Grafos - camino más corto

Algoritmo de Dijkstra

	Costo	Pred
~~Madrid~~	9999	Madrid
Londres	24	París
~~París~~	14	Madrid
Zúrich	26	París
Berlín	**50**	**París**
Roma	65	Madrid

$24 + 14 = 38 \quad < \quad 50$

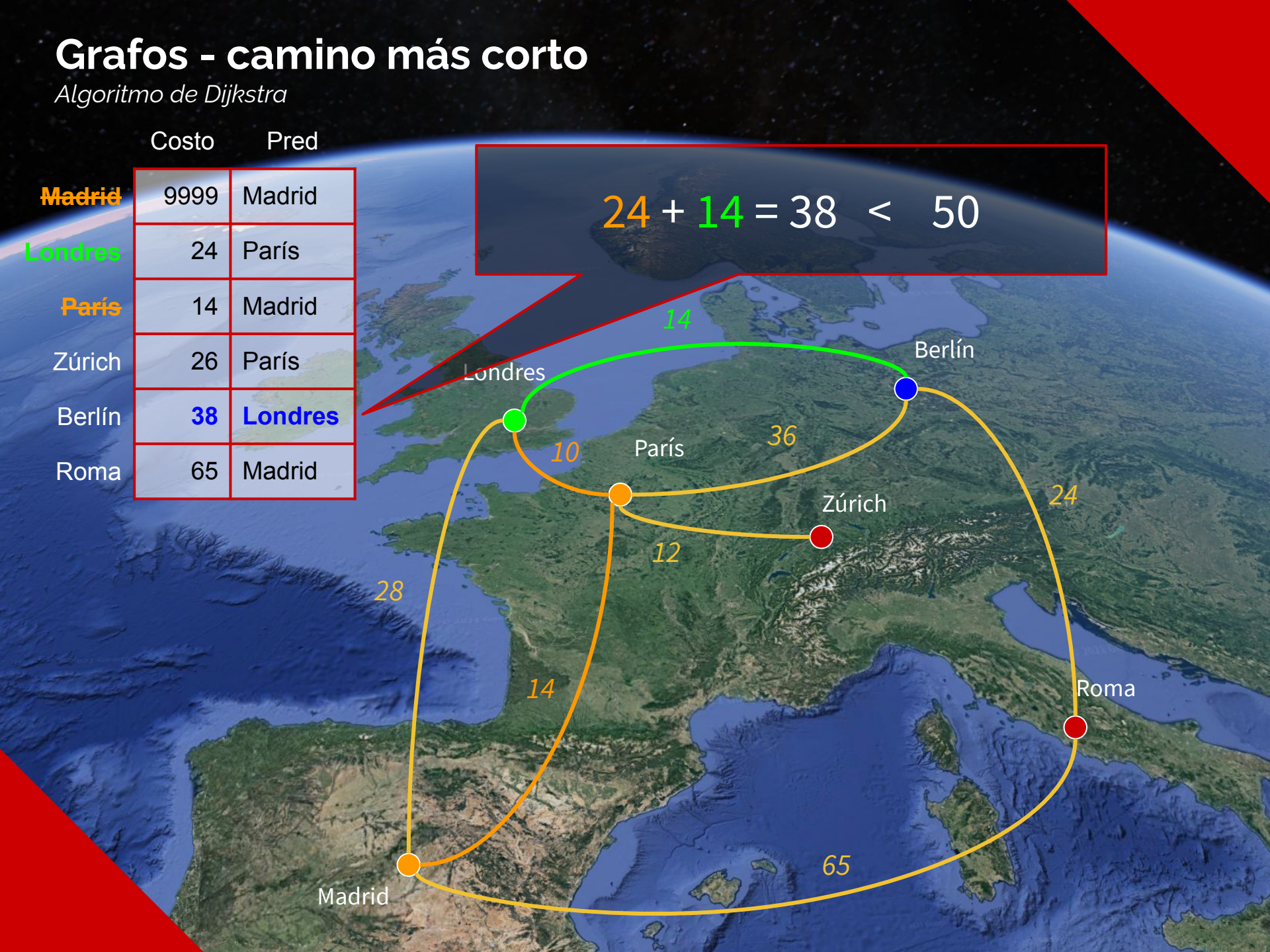


Grafos - camino más corto

Algoritmo de Dijkstra

	Costo	Pred
~~Madrid~~	9999	Madrid
Londres	24	París
~~París~~	14	Madrid
Zúrich	26	París
Berlín	**38**	**Londres**
Roma	65	Madrid

$24 + 14 = 38 \quad < \quad 50$

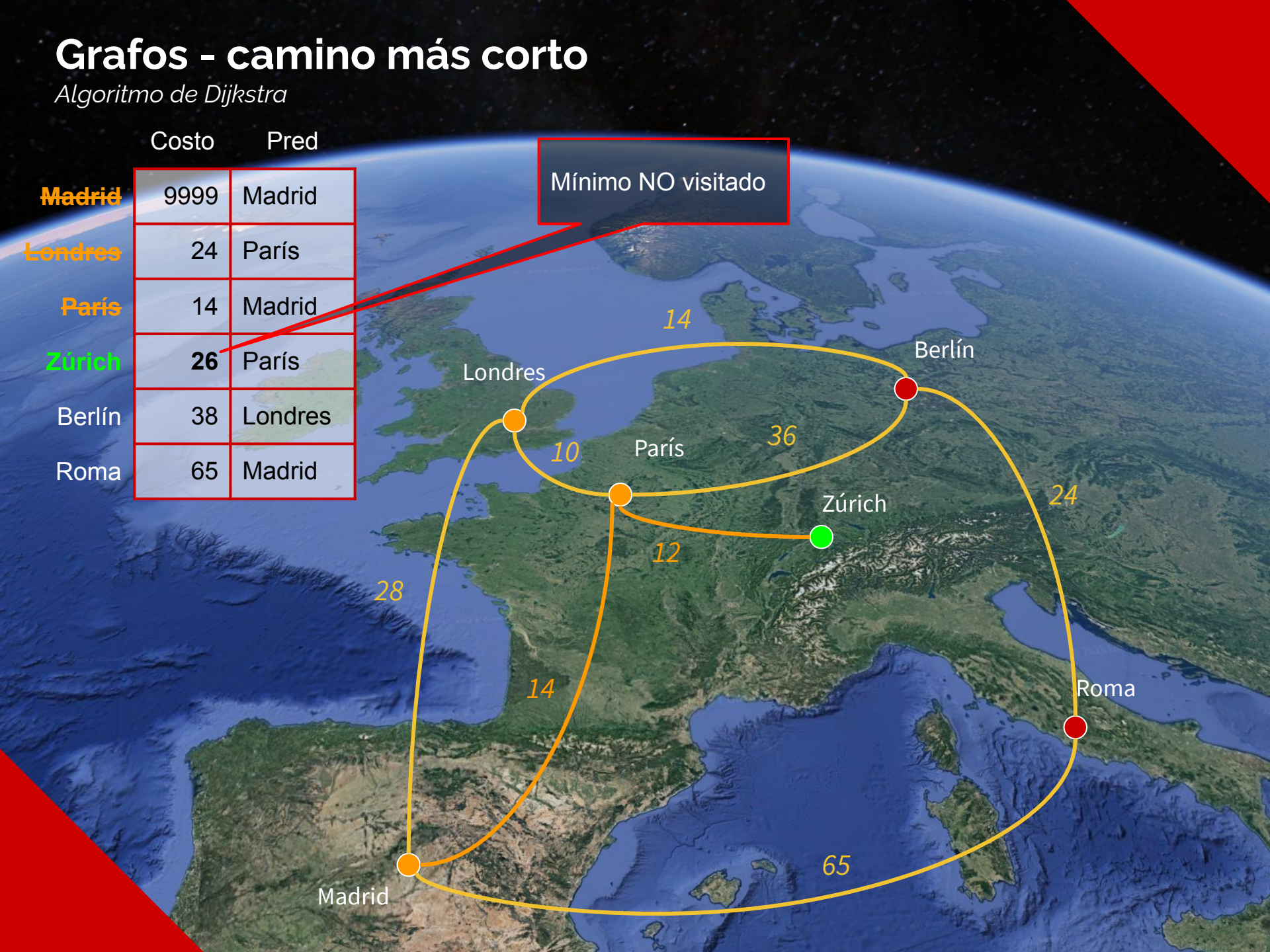


Grafos - camino más corto

Algoritmo de Dijkstra

	Costo	Pred
~~Madrid~~	9999	Madrid
~~Londres~~	24	París
~~París~~	14	Madrid
Zúrich	**26**	París
Berlín	38	Londres
Roma	65	Madrid

Mínimo NO visitado

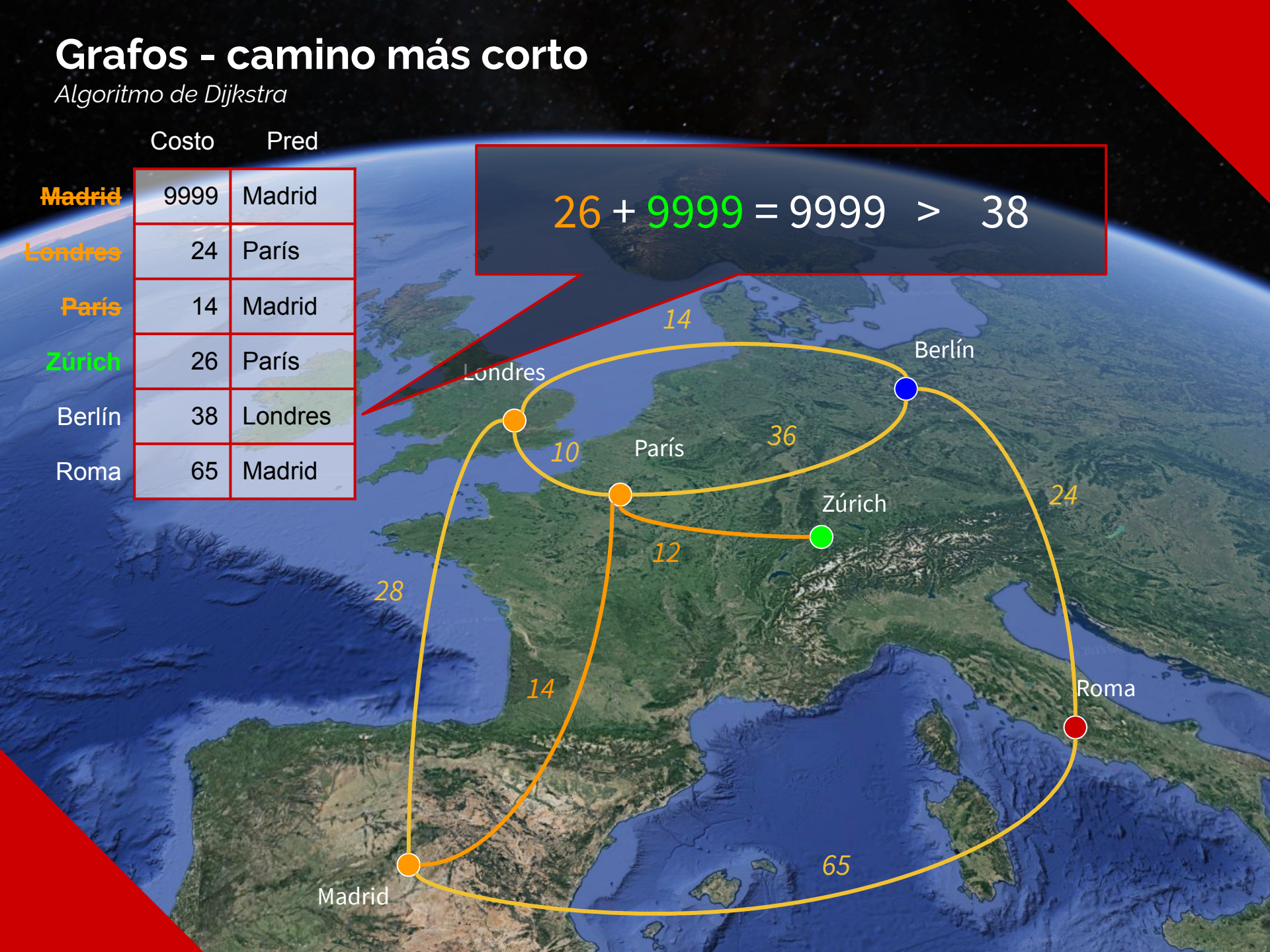


Grafos - camino más corto

Algoritmo de Dijkstra

	Costo	Pred
~~Madrid~~	9999	Madrid
~~Londres~~	24	París
~~París~~	14	Madrid
Zúrich	26	París
Berlín	38	Londres
Roma	65	Madrid

26 + 9999 = 9999 > 38

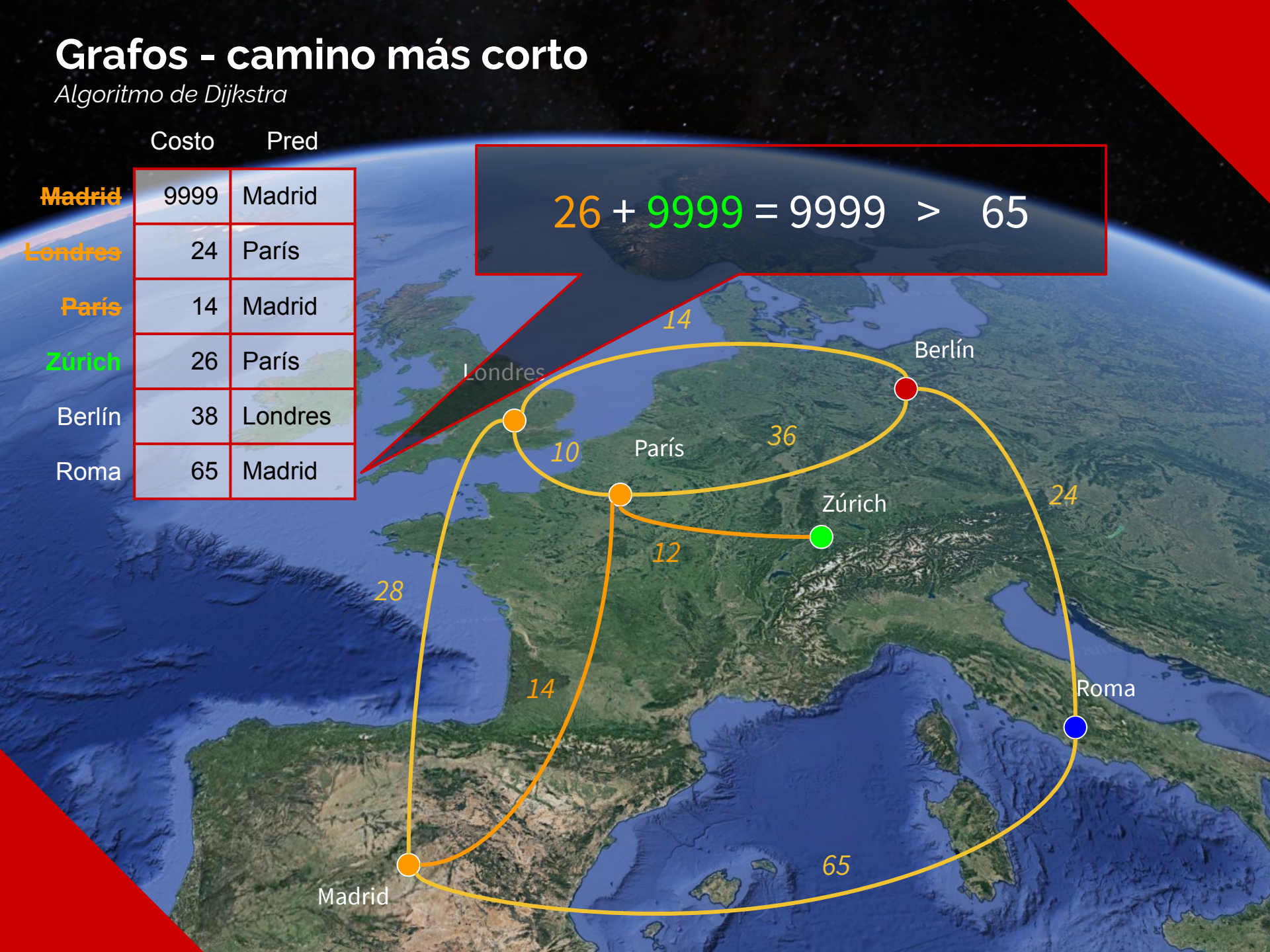


Grafos - camino más corto

Algoritmo de Dijkstra

	Costo	Pred
~~Madrid~~	9999	Madrid
~~Londres~~	24	París
~~París~~	14	Madrid
Zúrich	26	París
Berlín	38	Londres
Roma	65	Madrid

$26 + 9999 = 9999 > 65$

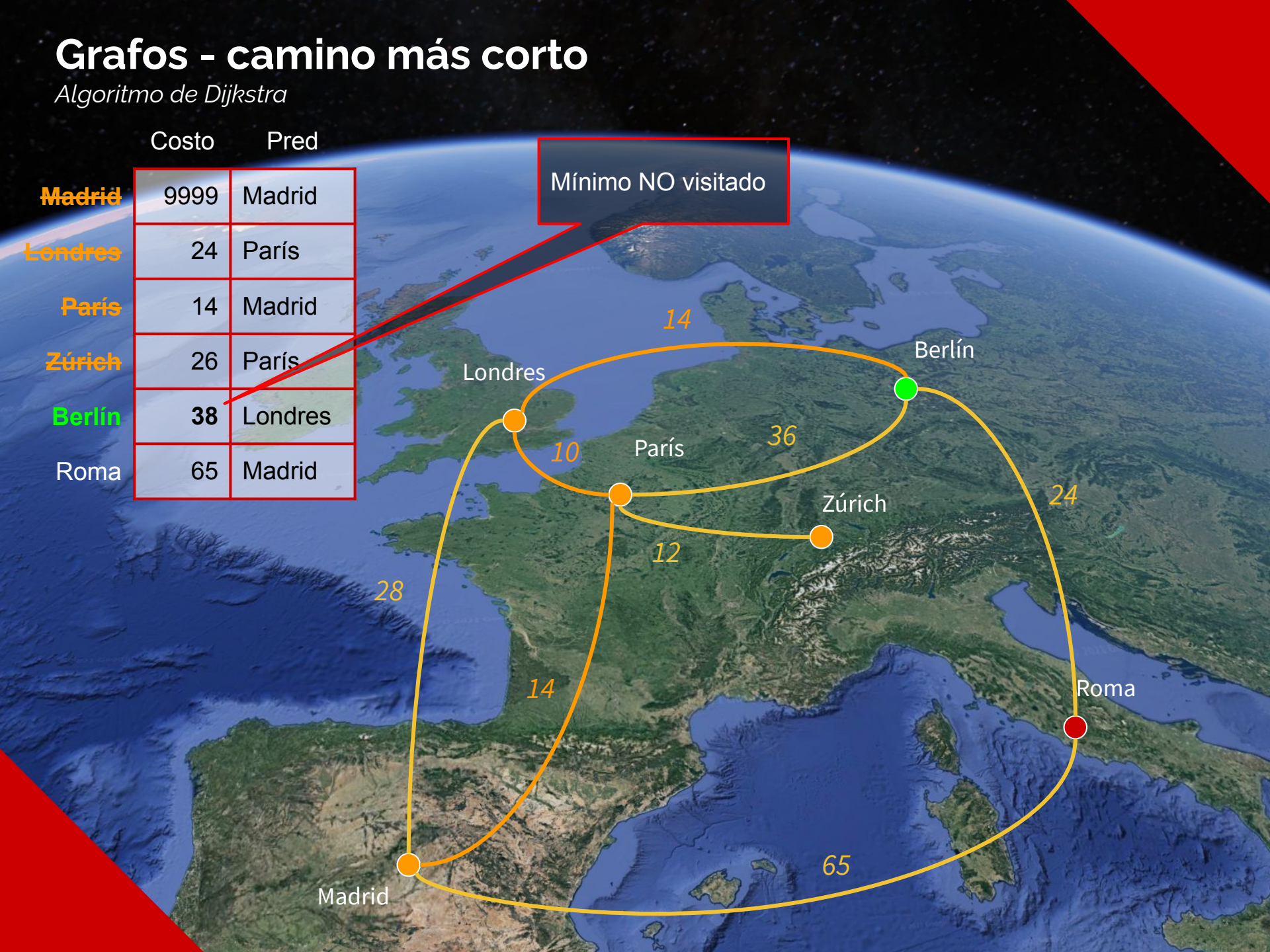


Grafos - camino más corto

Algoritmo de Dijkstra

	Costo	Pred
~~Madrid~~	9999	Madrid
~~Londres~~	24	París
~~París~~	14	Madrid
~~Zúrich~~	26	París
Berlín	**38**	Londres
Roma	65	Madrid

Mínimo NO visitado

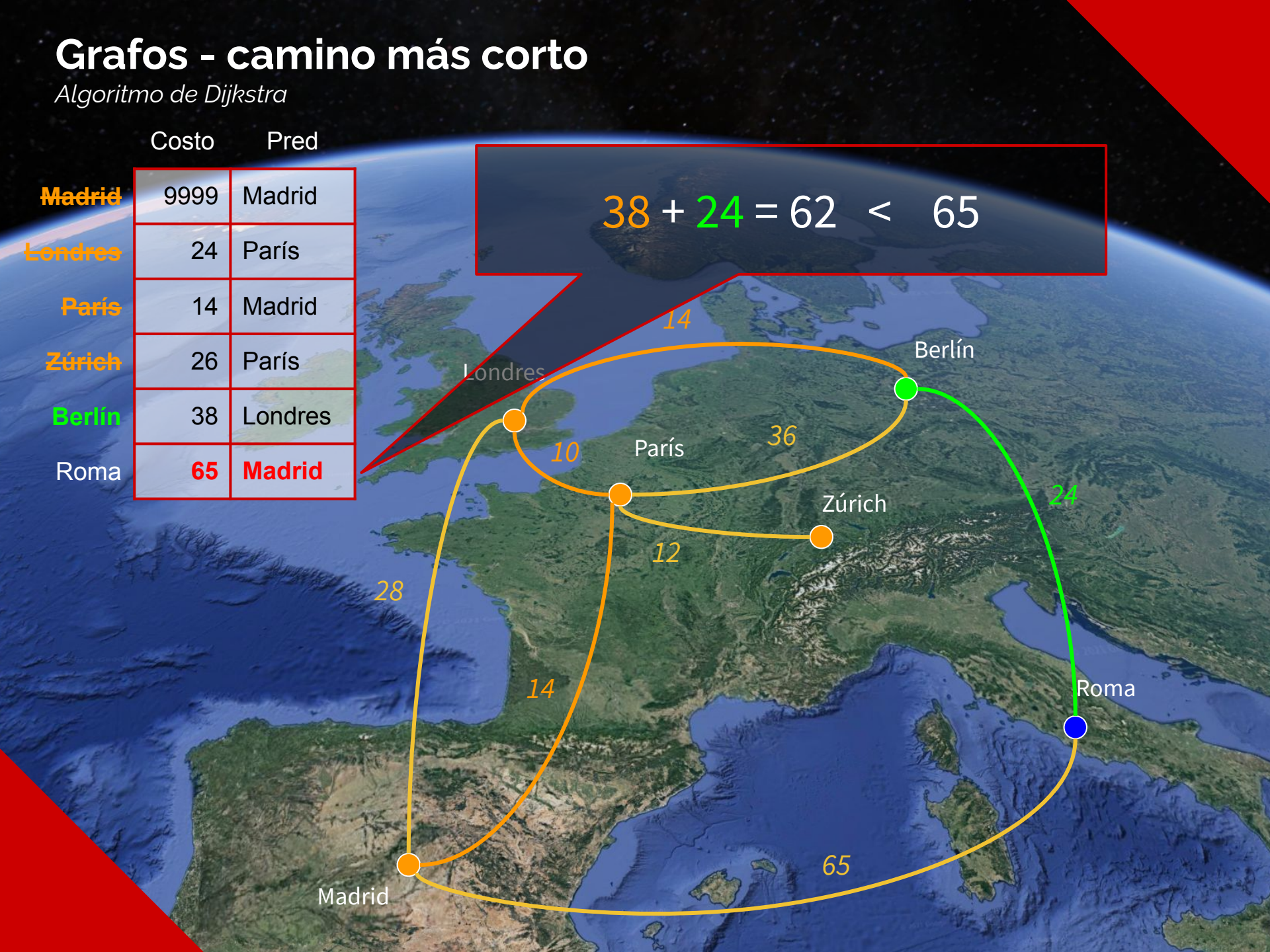


Grafos - camino más corto

Algoritmo de Dijkstra

	Costo	Pred
~~Madrid~~	9999	Madrid
~~Londres~~	24	París
~~París~~	14	Madrid
~~Zúrich~~	26	París
Berlín	38	Londres
Roma	**65**	**Madrid**

$$38 + 24 = 62 \quad < \quad 65$$

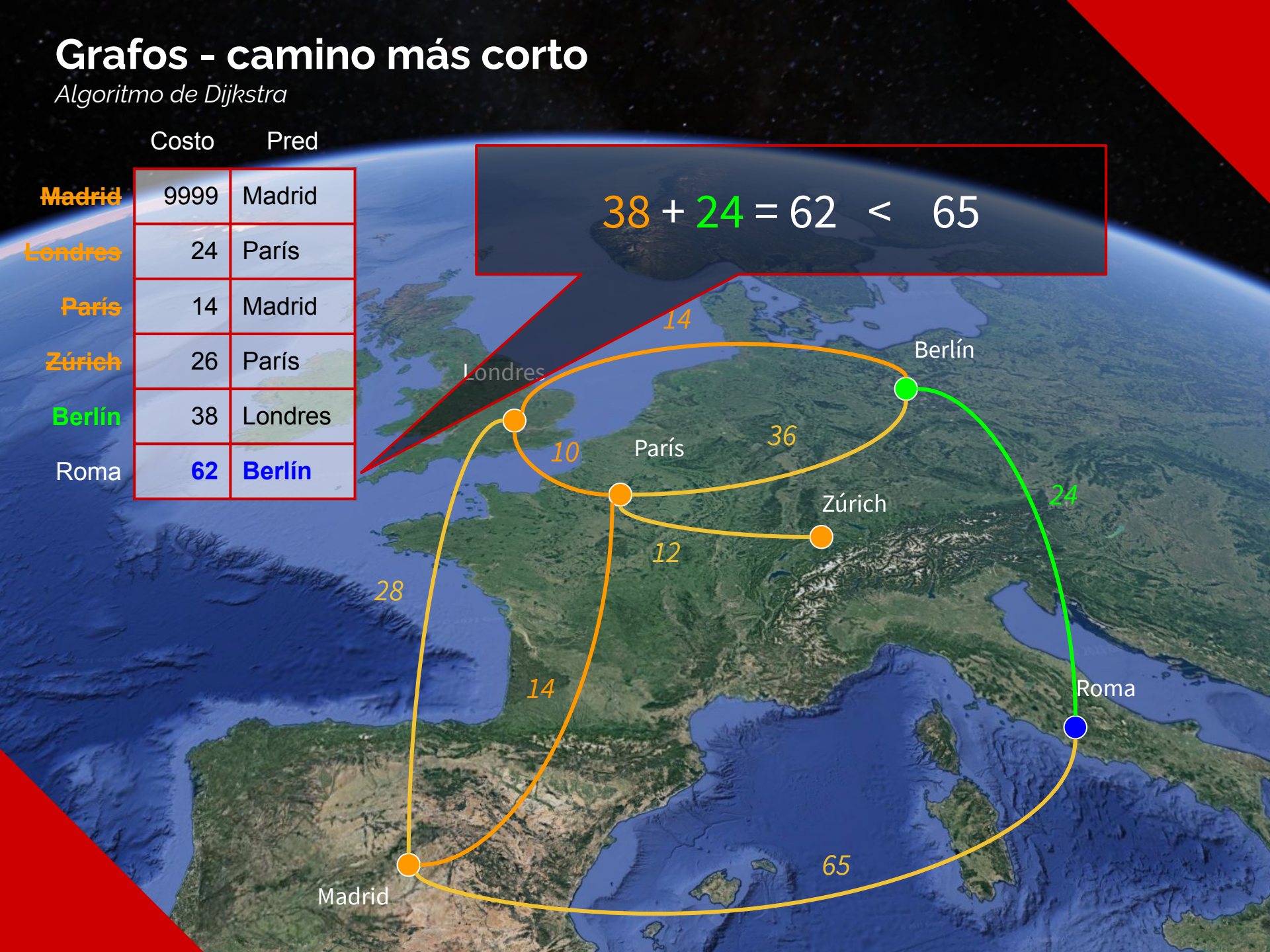


Grafos - camino más corto

Algoritmo de Dijkstra

	Costo	Pred
~~Madrid~~	9999	Madrid
~~Londres~~	24	París
~~París~~	14	Madrid
~~Zúrich~~	26	París
Berlín	38	Londres
Roma	**62**	**Berlín**

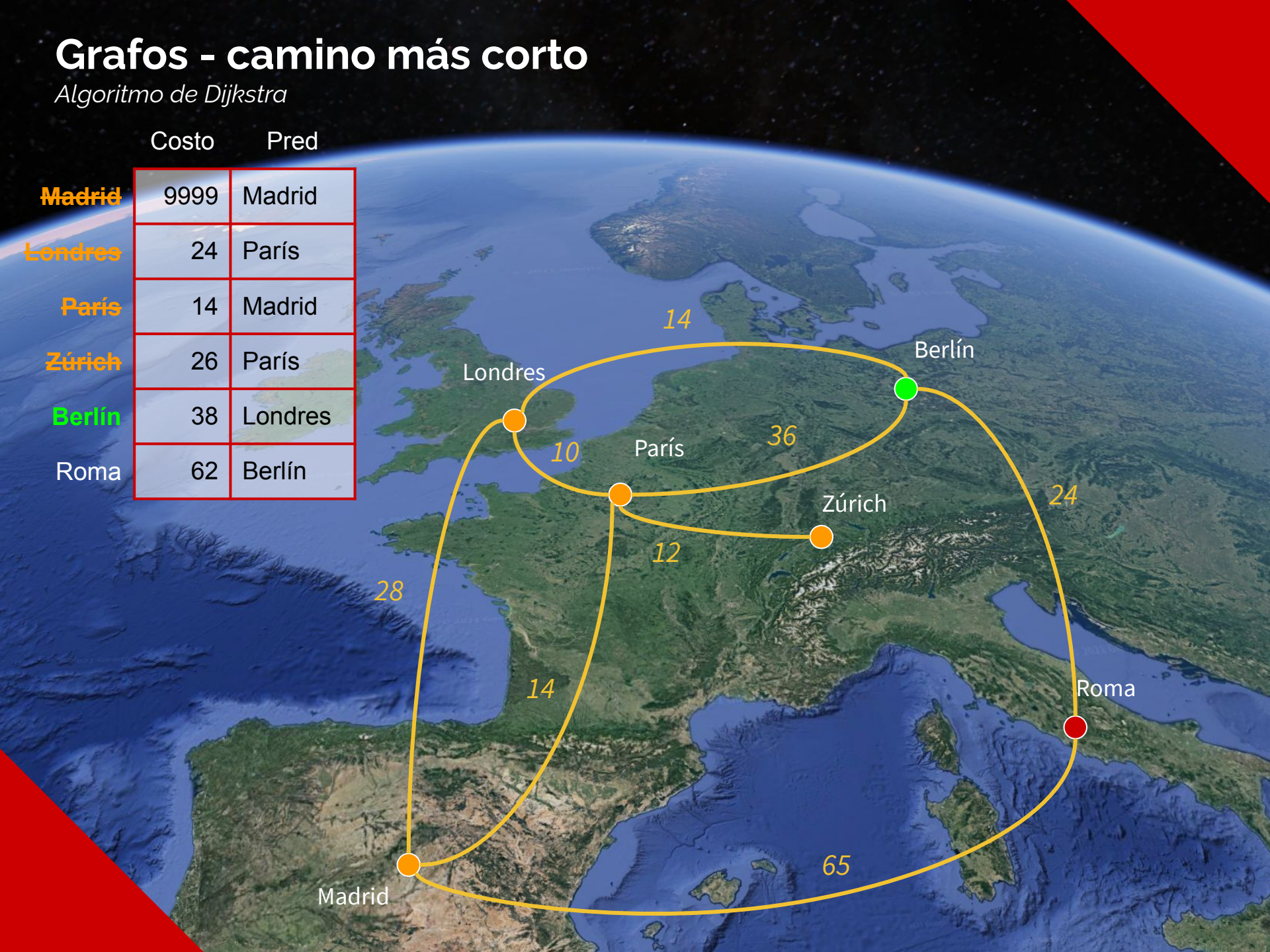
$38 + 24 = 62 \quad < \quad 65$



Grafos - camino más corto

Algoritmo de Dijkstra

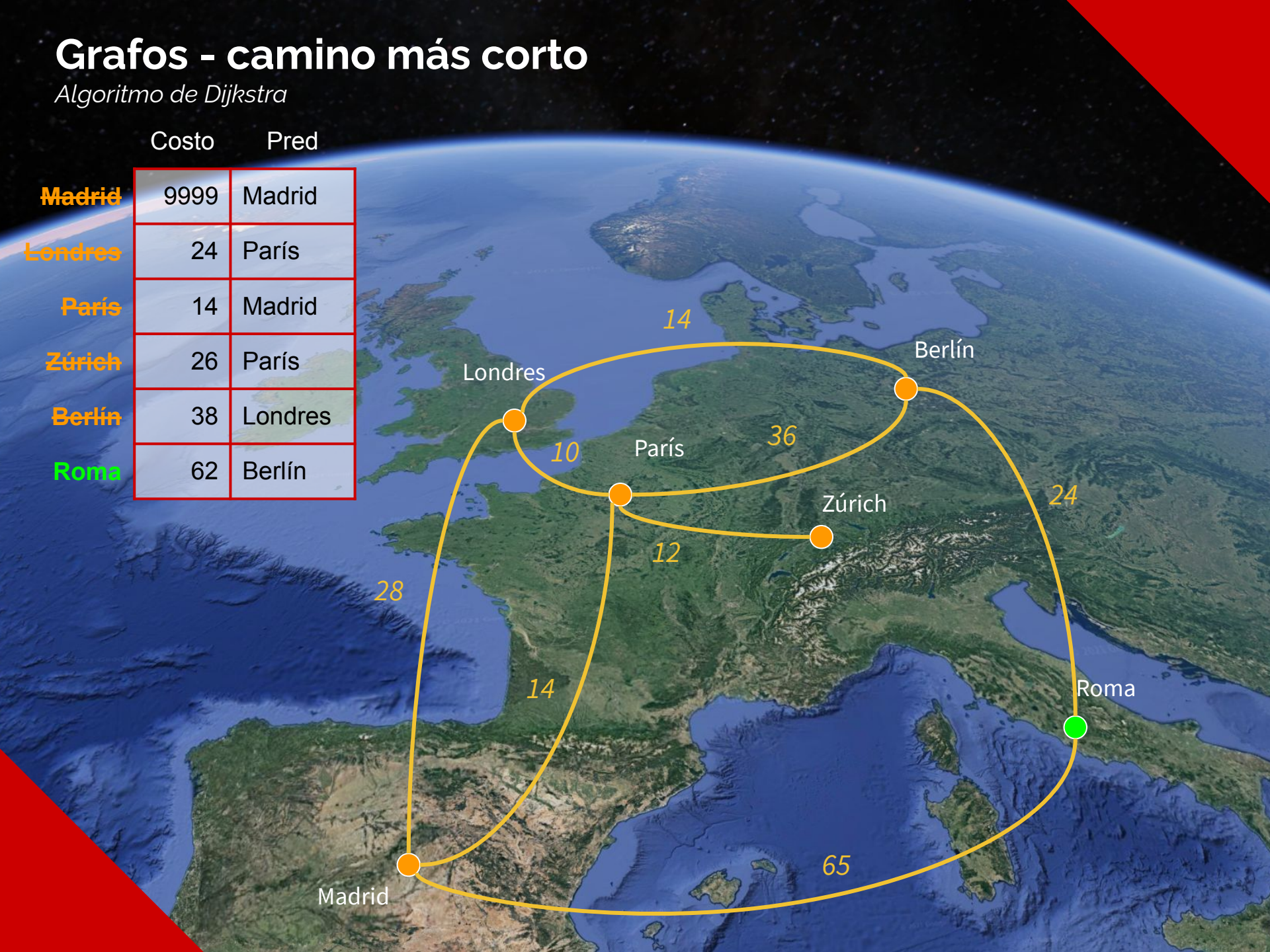
	Costo	Pred
~~Madrid~~	9999	Madrid
~~Londres~~	24	París
~~París~~	14	Madrid
~~Zúrich~~	26	París
Berlín	38	Londres
Roma	62	Berlín



Grafos - camino más corto

Algoritmo de Dijkstra

	Costo	Pred
~~Madrid~~	9999	Madrid
~~Londres~~	24	París
~~París~~	14	Madrid
~~Zúrich~~	26	París
~~Berlín~~	38	Londres
Roma	62	Berlín

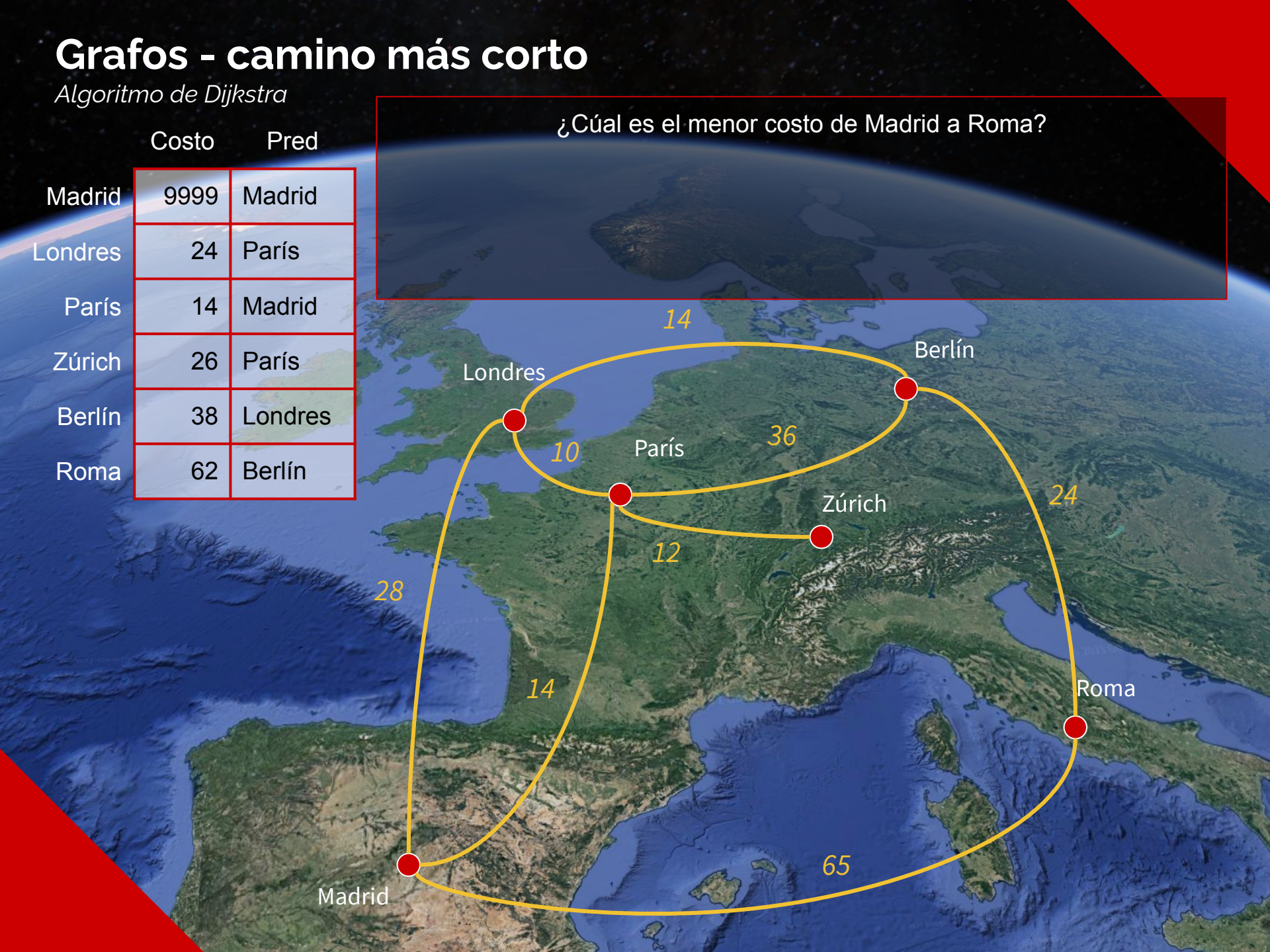


Grafos - camino más corto

Algoritmo de Dijkstra

¿Cúal es el menor costo de Madrid a Roma?

	Costo	Pred
Madrid	9999	Madrid
Londres	24	París
París	14	Madrid
Zúrich	26	París
Berlín	38	Londres
Roma	62	Berlín



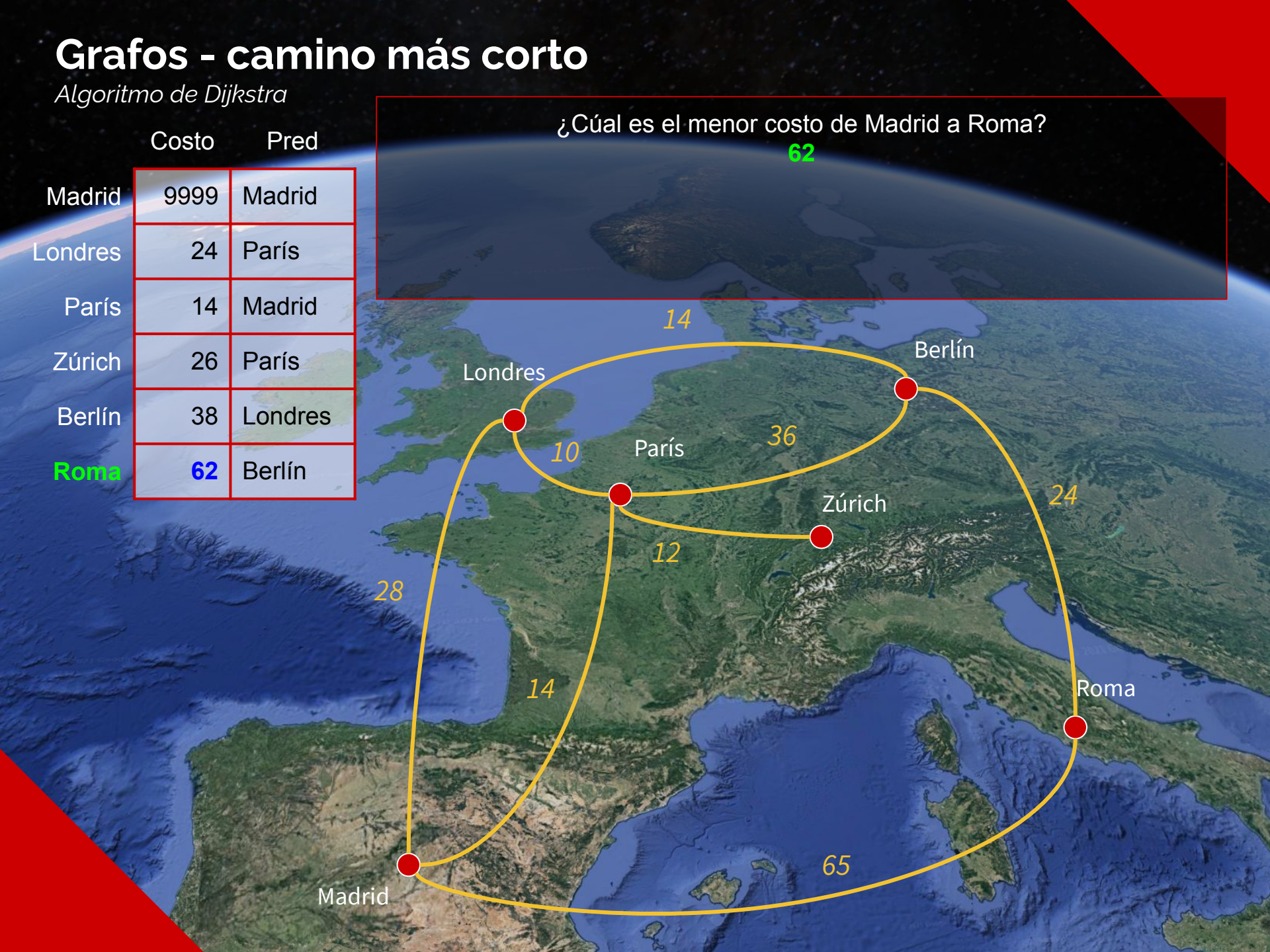
Grafos - camino más corto

Algoritmo de Dijkstra

	Costo	Pred
Madrid	9999	Madrid
Londres	24	París
París	14	Madrid
Zúrich	26	París
Berlín	38	Londres
Roma	**62**	Berlín

¿Cúal es el menor costo de Madrid a Roma?

62



Grafos - camino más corto

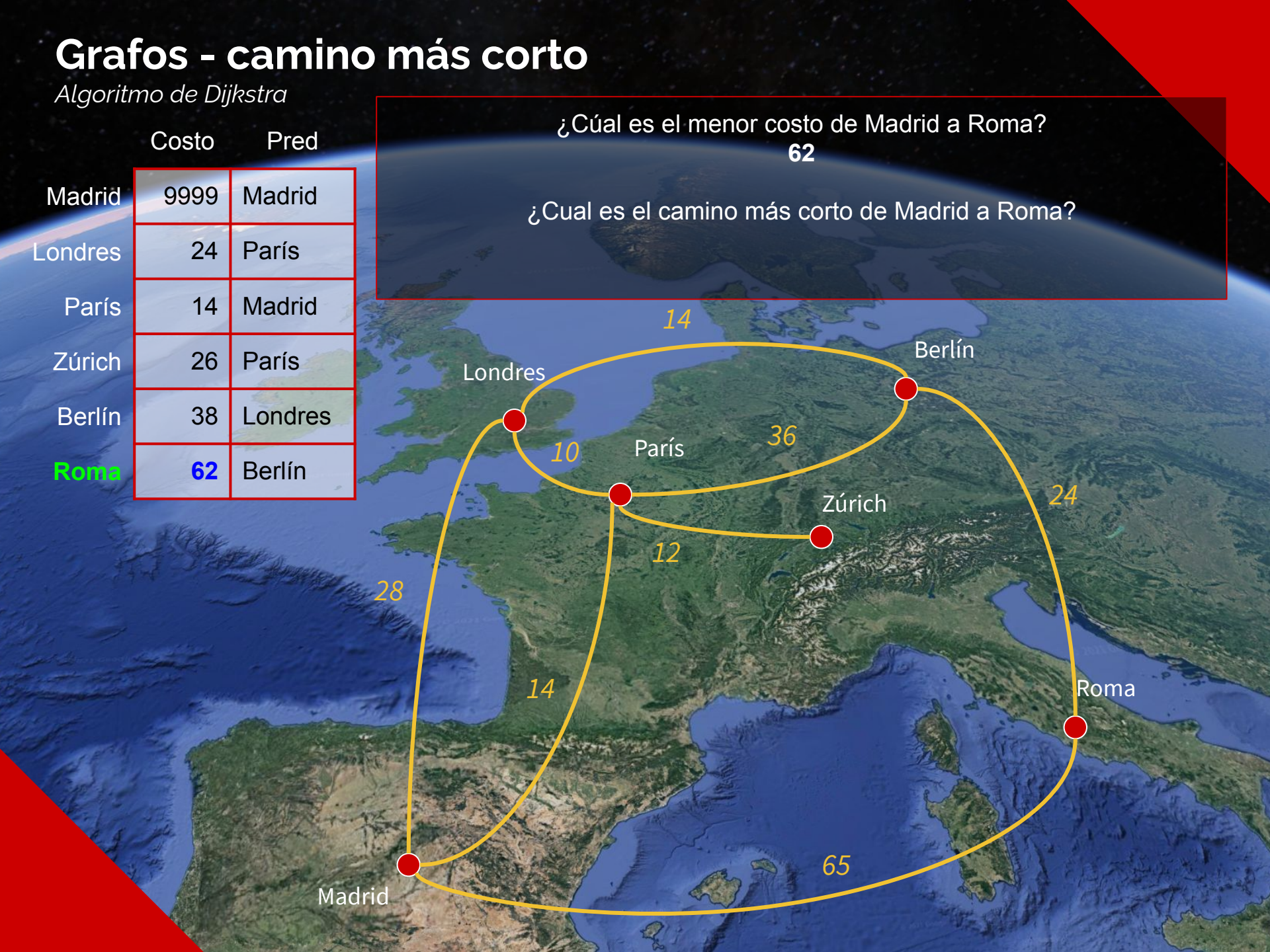
Algoritmo de Dijkstra

	Costo	Pred
Madrid	9999	Madrid
Londres	24	París
París	14	Madrid
Zúrich	26	París
Berlín	38	Londres
Roma	**62**	Berlín

¿Cúal es el menor costo de Madrid a Roma?

62

¿Cual es el camino más corto de Madrid a Roma?



Grafos - camino más corto

Algoritmo de Dijkstra

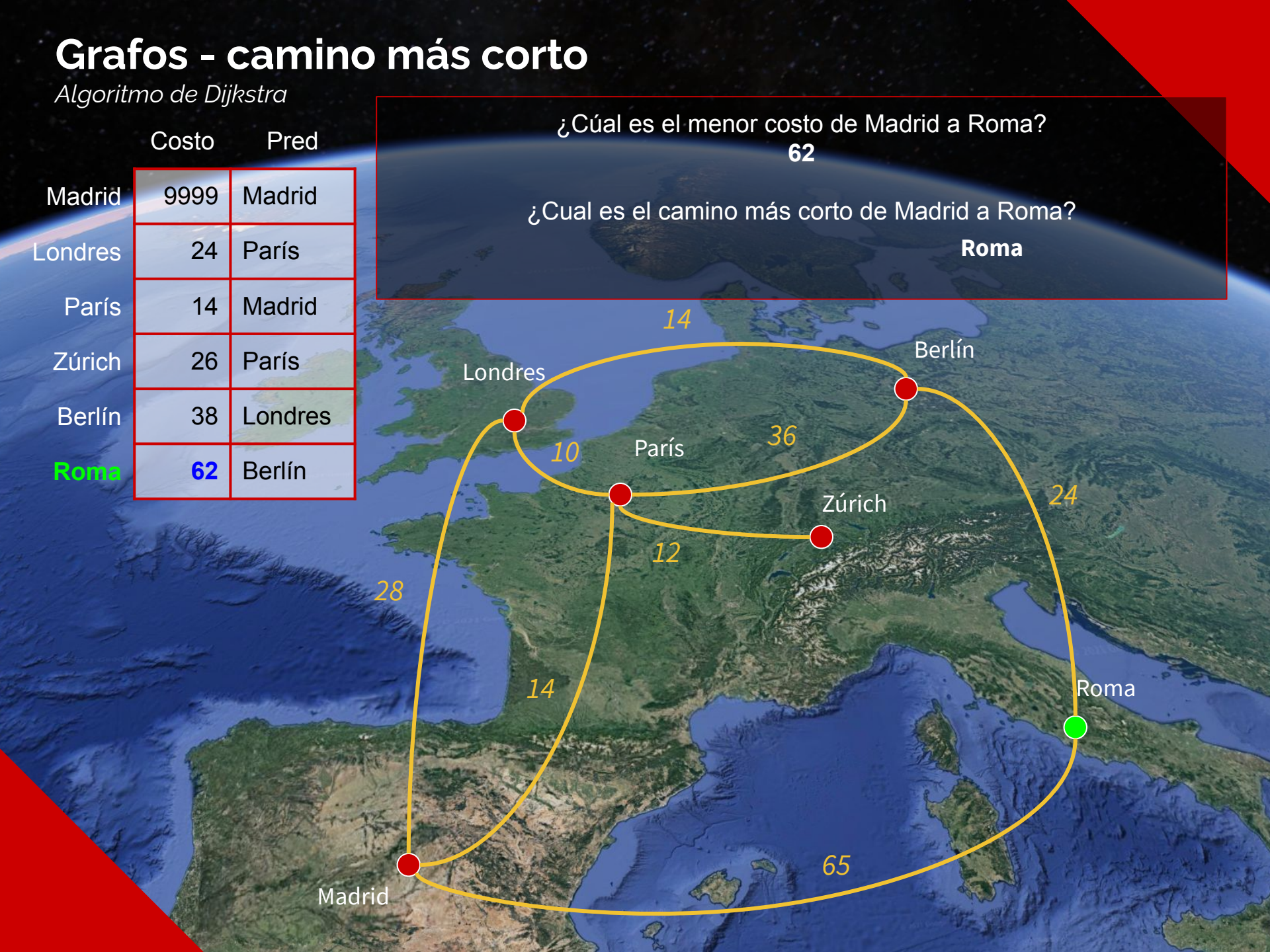
	Costo	Pred
Madrid	9999	Madrid
Londres	24	París
París	14	Madrid
Zúrich	26	París
Berlín	38	Londres
Roma	**62**	Berlín

¿Cúal es el menor costo de Madrid a Roma?

62

¿Cual es el camino más corto de Madrid a Roma?

Roma



Grafos - camino más corto

Algoritmo de Dijkstra

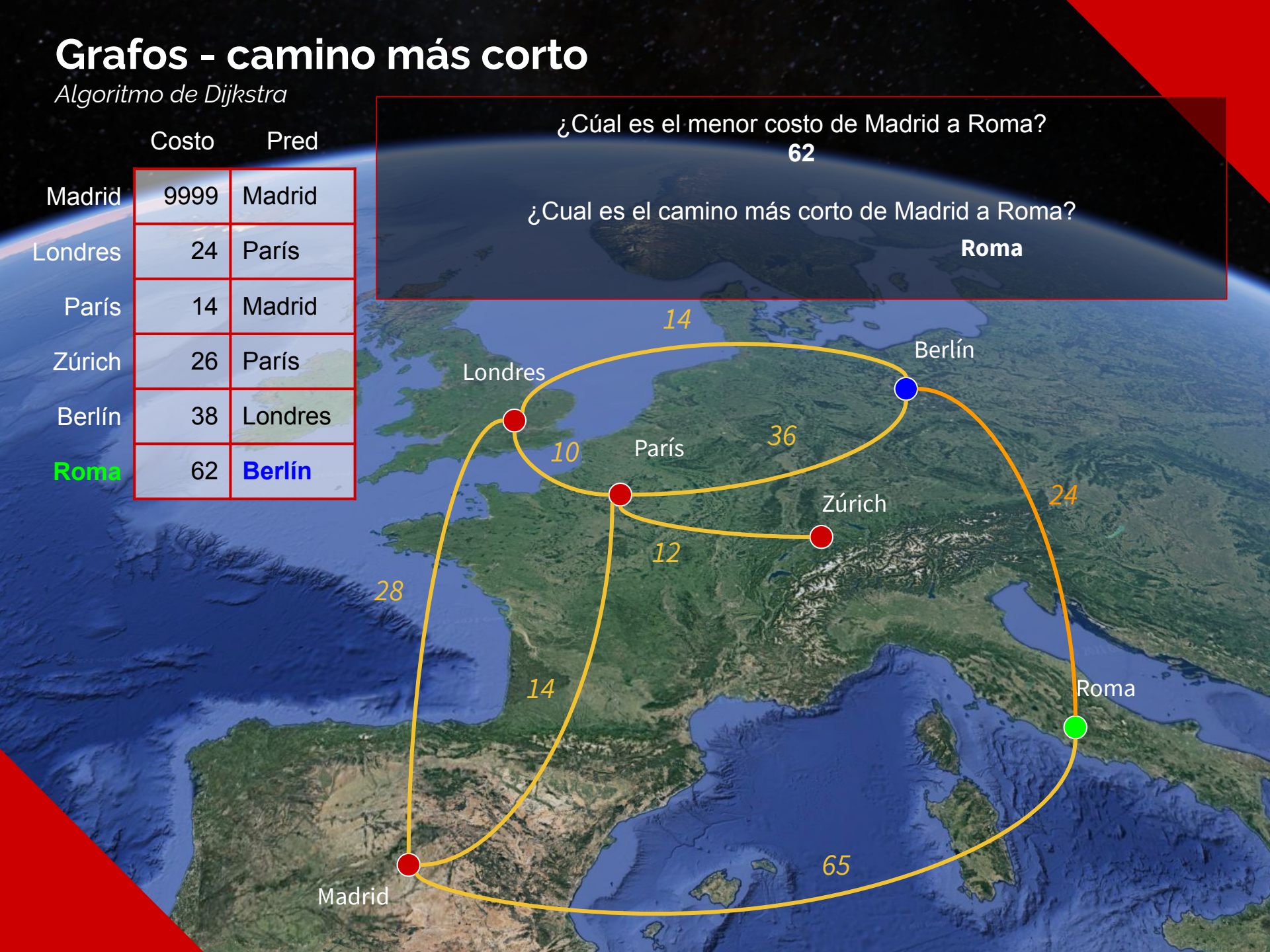
	Costo	Pred
Madrid	9999	Madrid
Londres	24	París
París	14	Madrid
Zúrich	26	París
Berlín	38	Londres
Roma	62	**Berlín**

¿Cúal es el menor costo de Madrid a Roma?

62

¿Cual es el camino más corto de Madrid a Roma?

Roma



Grafos - camino más corto

Algoritmo de Dijkstra

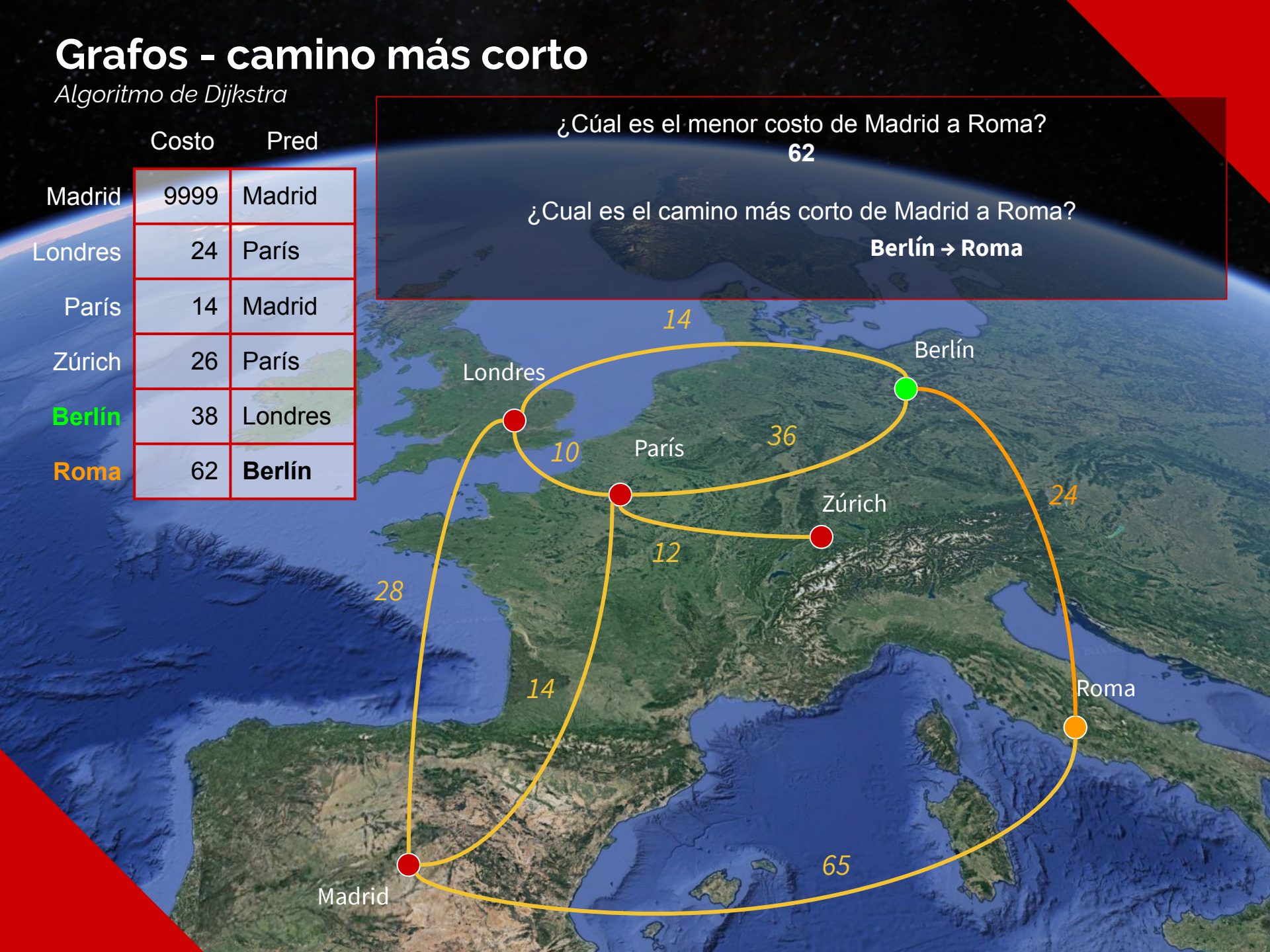
	Costo	Pred
Madrid	9999	Madrid
Londres	24	París
París	14	Madrid
Zúrich	26	París
Berlín	38	Londres
Roma	62	**Berlín**

¿Cúal es el menor costo de Madrid a Roma?

62

¿Cual es el camino más corto de Madrid a Roma?

Berlín → Roma



Grafos - camino más corto

Algoritmo de Dijkstra

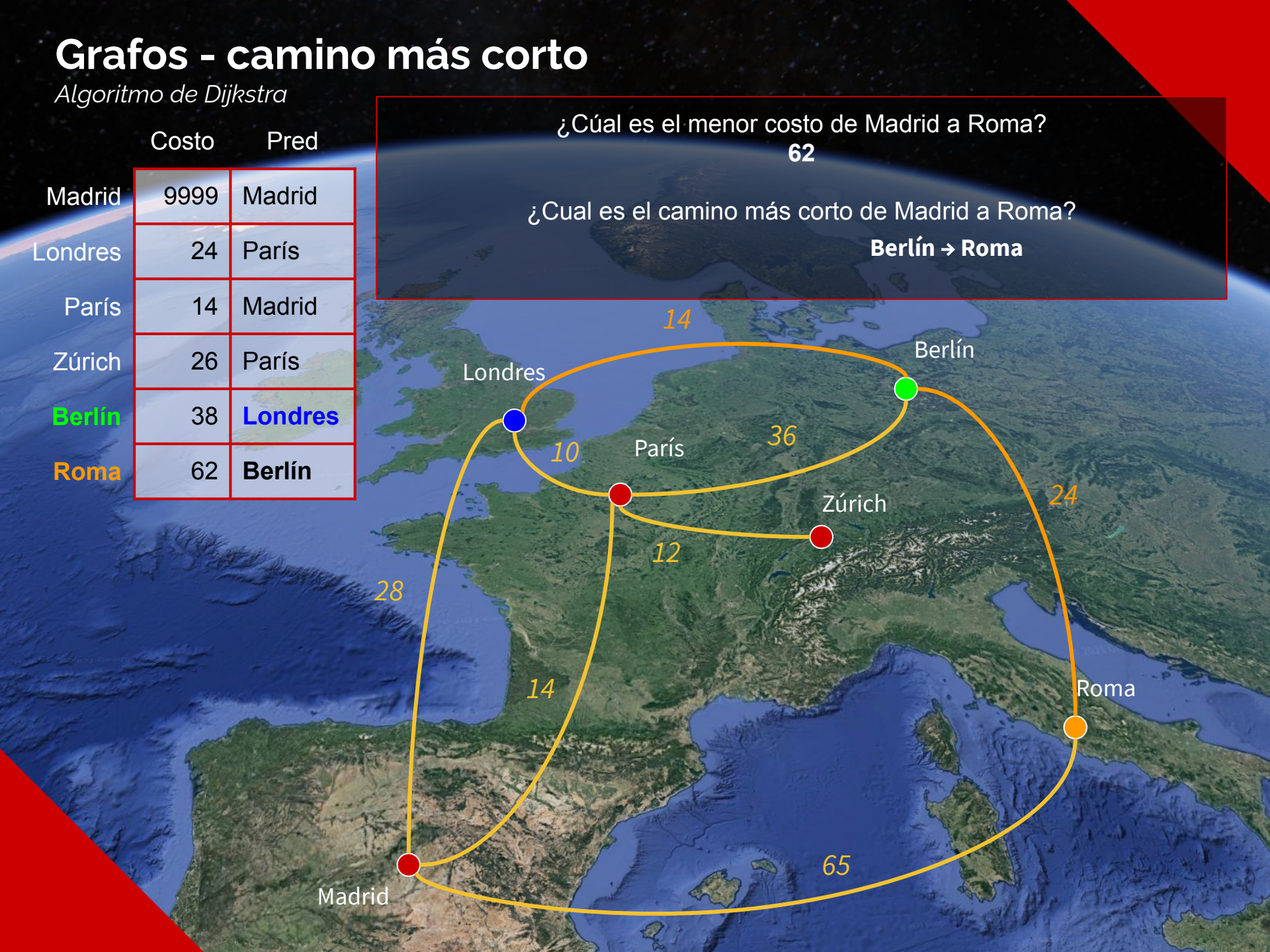
	Costo	Pred
Madrid	9999	Madrid
Londres	24	París
París	14	Madrid
Zúrich	26	París
Berlín	38	**Londres**
Roma	62	**Berlín**

¿Cúal es el menor costo de Madrid a Roma?

62

¿Cual es el camino más corto de Madrid a Roma?

Berlín → Roma



Grafos - camino más corto

Algoritmo de Dijkstra

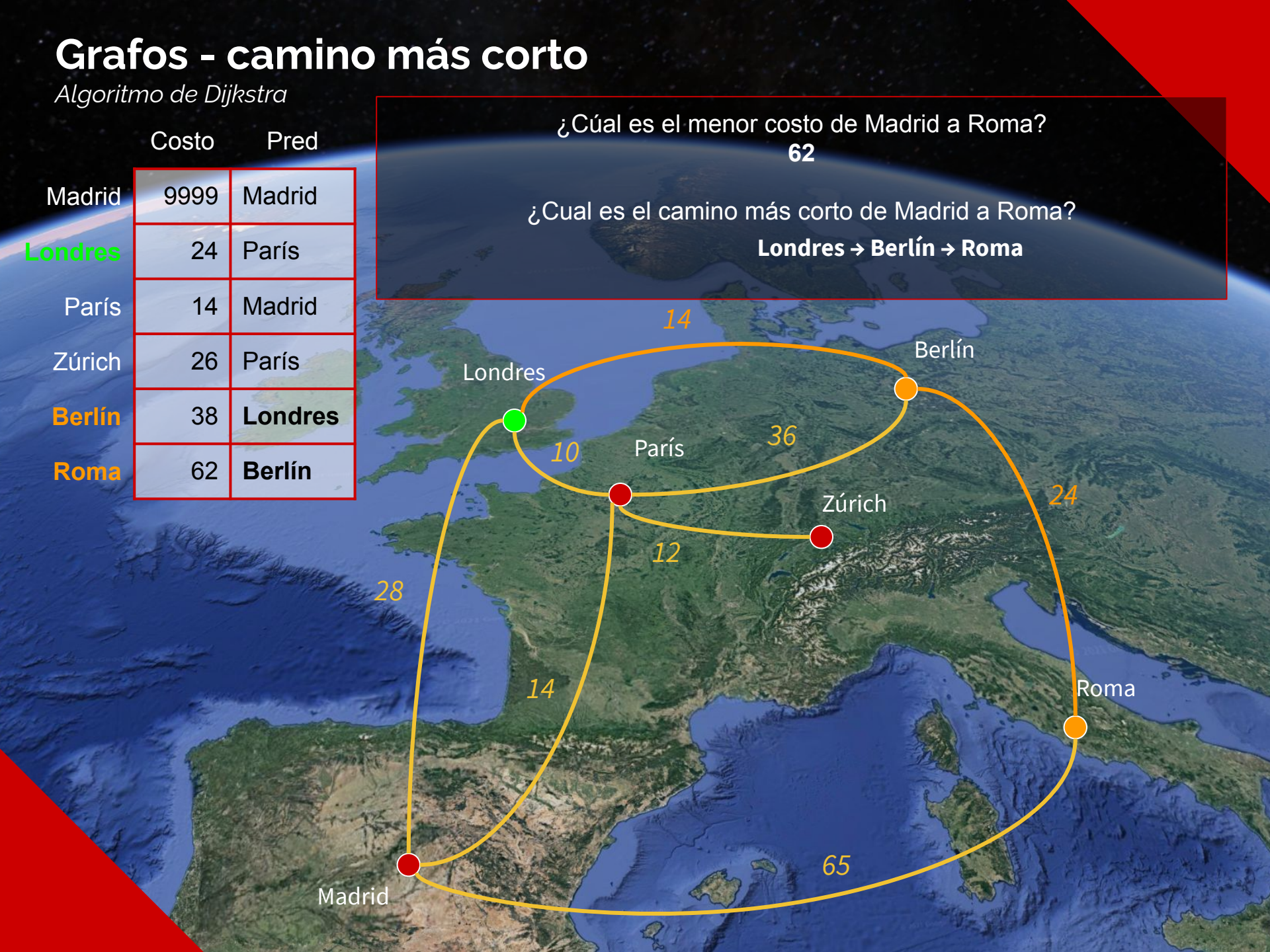
	Costo	Pred
Madrid	9999	Madrid
Londres	24	París
París	14	Madrid
Zúrich	26	París
Berlín	38	**Londres**
Roma	62	**Berlín**

¿Cúal es el menor costo de Madrid a Roma?

62

¿Cual es el camino más corto de Madrid a Roma?

Londres → Berlín → Roma



Grafos - camino más corto

Algoritmo de Dijkstra

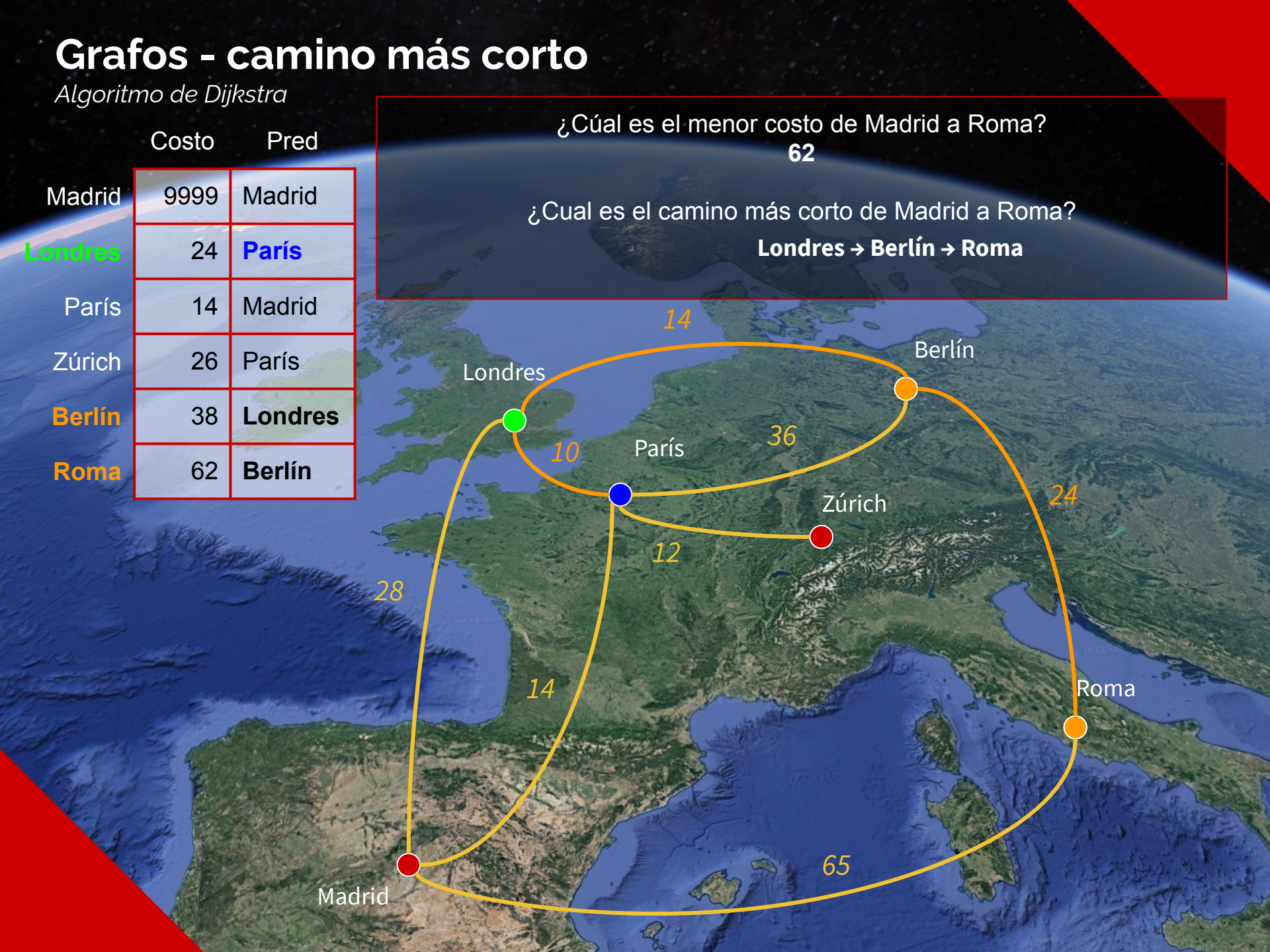
	Costo	Pred
Madrid	9999	Madrid
Londres	24	**París**
París	14	Madrid
Zúrich	26	París
Berlín	38	**Londres**
Roma	62	**Berlín**

¿Cúal es el menor costo de Madrid a Roma?

62

¿Cual es el camino más corto de Madrid a Roma?

Londres → Berlín → Roma



Grafos - camino más corto

Algoritmo de Dijkstra

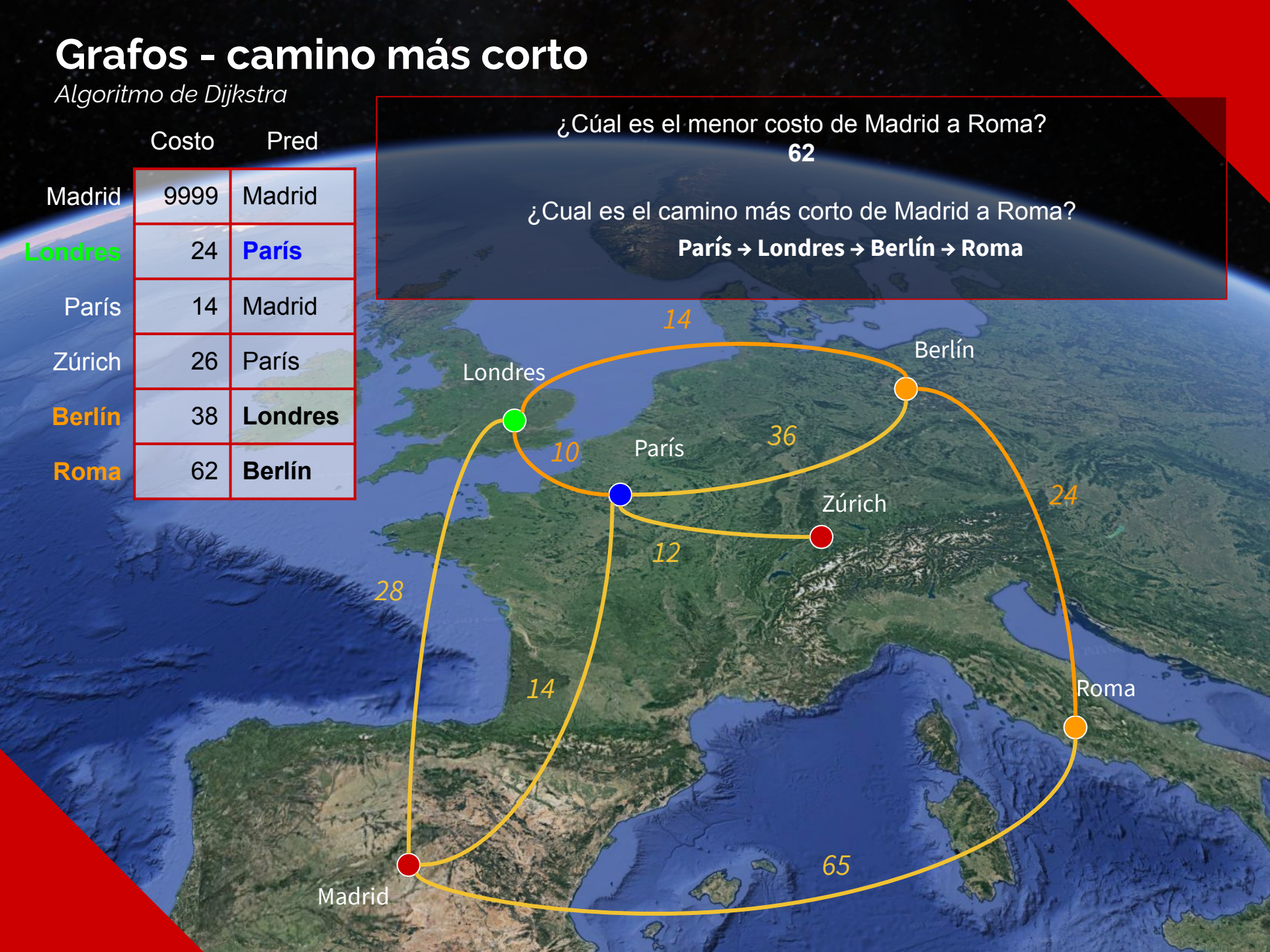
	Costo	Pred
Madrid	9999	Madrid
Londres	24	**París**
París	14	Madrid
Zúrich	26	París
Berlín	38	**Londres**
Roma	62	**Berlín**

¿Cúal es el menor costo de Madrid a Roma?

62

¿Cual es el camino más corto de Madrid a Roma?

París → Londres → Berlín → Roma



Grafos - camino más corto

Algoritmo de Dijkstra

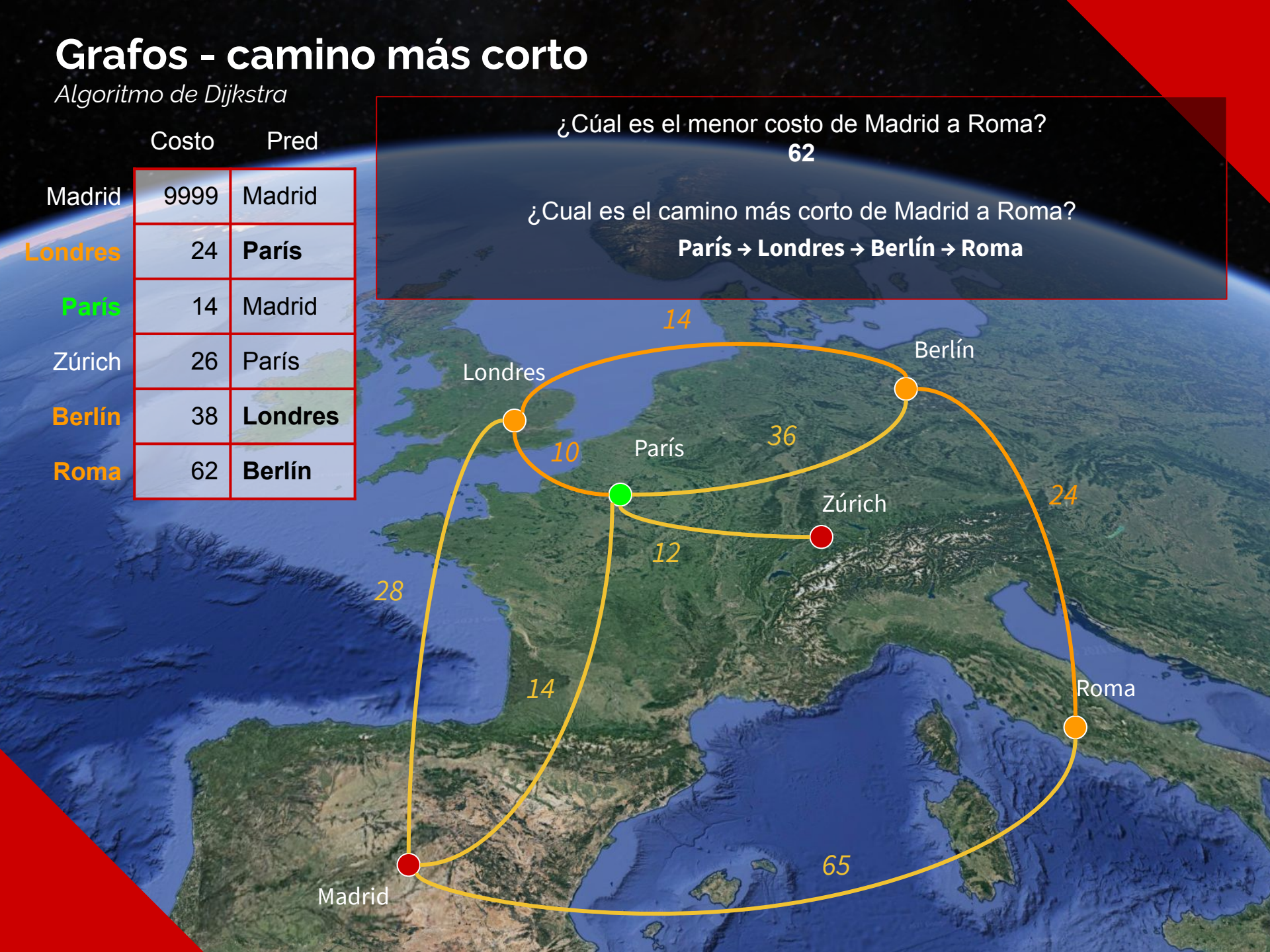
	Costo	Pred
Madrid	9999	Madrid
Londres	24	**París**
París	14	Madrid
Zúrich	26	París
Berlín	38	**Londres**
Roma	62	**Berlín**

¿Cúal es el menor costo de Madrid a Roma?

62

¿Cual es el camino más corto de Madrid a Roma?

París → Londres → Berlín → Roma



Grafos - camino más corto

Algoritmo de Dijkstra

	Costo	Pred
Madrid	9999	Madrid
Londres	24	**París**
París	14	**Madrid**
Zúrich	26	París
Berlín	38	**Londres**
Roma	62	**Berlín**

¿Cúal es el menor costo de Madrid a Roma?

62

¿Cual es el camino más corto de Madrid a Roma?

París → Londres → Berlín → Roma

Grafos - camino más corto

Algoritmo de Dijkstra

	Costo	Pred
Madrid	9999	Madrid
Londres	24	**París**
París	14	**Madrid**
Zúrich	26	París
Berlín	38	**Londres**
Roma	62	**Berlín**

¿Cúal es el menor costo de Madrid a Roma?

62

¿Cual es el camino más corto de Madrid a Roma?

Madrid → París → Londres → Berlín → Roma

Unidad 6: Grafos

Algoritmos y Estructuras de Datos

Concepto y Clasificación
Implementaciones
Algoritmo de Dijkstra

Ing. Juan Ignacio Iturriaga