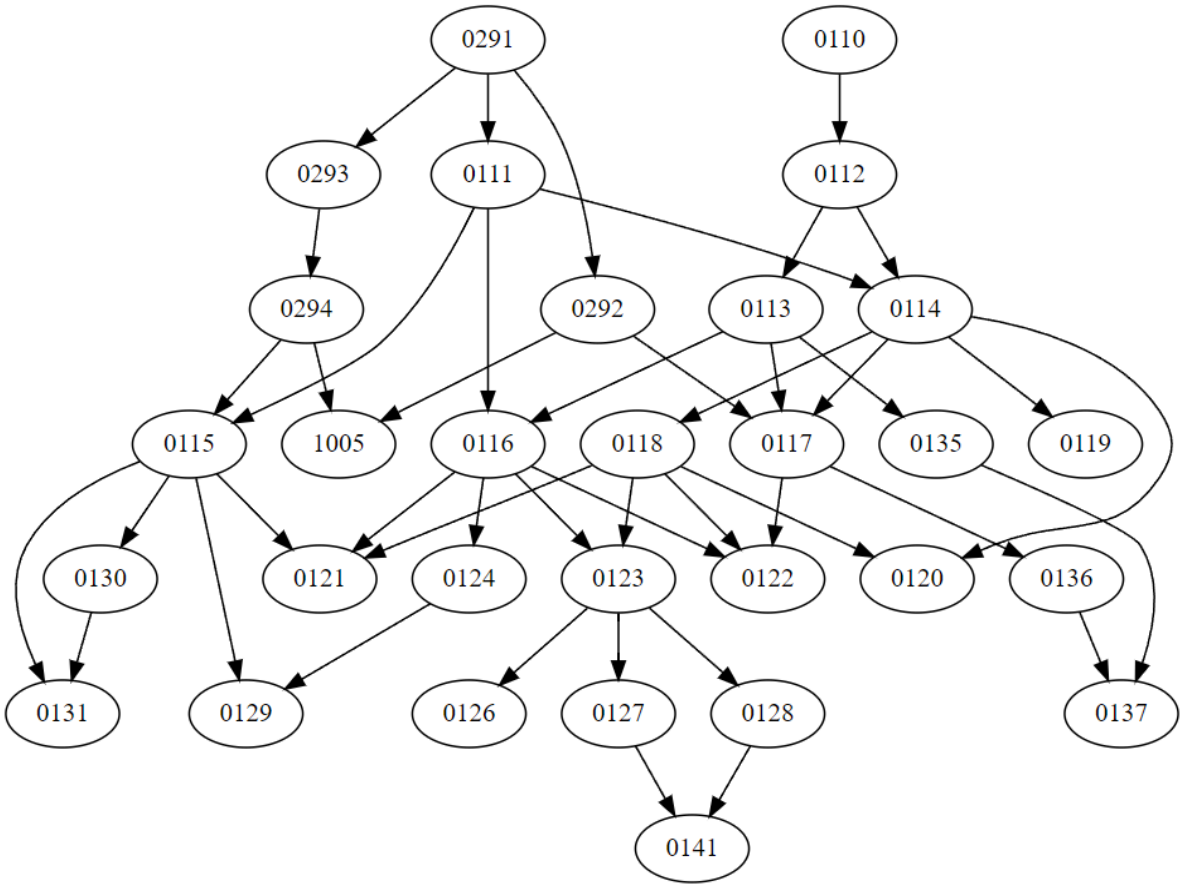
**Universidad de Costa Rica**
Escuela de Ciencias de la Computación e Informática
CI0116 Análisis de Algoritmos y Estructuras de Datos - Grupo 02
Docente: Allan Berrocal

**Quiz 09**

Luis Antonio Fonseca Chinchilla - C03035
Ericka Melissa Araya Hidalgo - C20553

## 01. Ordenamiento topológico

Para realizar el ordenamiento topológico, primeramente construimos el grafo sin pesos.




Construimos el grafo de ordenamiento topológico aplicando *Deep-first Search* (DFS) en el grafo anterior, y cada que un vértice fija su tiempo final se inserta en la lista al frente, tal como indica el algoritmo del libro de *Cormen et al* (2022).

```
TOPOLOGICAL-SORT(G)
1  call DFS(G) to compute finish times v.f for each vertex v
2  as each vertex is finished, insert it onto the front of a linked list
3  return the linked list of vertices
```

| Iteración | tiempo | u | u.d | u.f | u.color | v | v.π |
|---|---|---|---|---|---|---|---|
| 1 | 0 | | | | B | | nil |
| 2 | 1 | 0110 | 1 | | G | 0112 | 0110 |
| 3 | 2 | 0112 | 2 | | G | 0113 | 0112 |
| 4 | 3 | 0113 | 3 | | G | 0116 | 0113 |
| 5 | 4 | 0116 | 4 | | G | 0121 | 0116 |
| 6 | 5 | 0121 | 5 | | G | | |
| 7 | 6 | 0121 | 5 | 6 | N | | |
| 8 | 6 | 0116 | 4 | | G | 0122 | 1116 |
| 9 | 7 | 0122 | 7 | | G | | |
| 10 | 8 | 0122 | 7 | 8 | N | | |
| 11 | 8 | 0116 | 4 | | G | 0123 | 0116 |
| 12 | 9 | 0123 | 9 | | G | 0126 | 0123 |
| 13 | 10 | 0126 | 10 | | G | | |
| 14 | 11 | 0126 | 10 | 11 | N | | |
| 15 | 11 | 0123 | 9 | | G | 0127 | 0123 |
| 16 | 12 | 0127 | 12 | | G | 0141 | 1027 |
| 17 | 13 | 0141 | 13 | | G | | |
| 18 | 14 | 0141 | 13 | 14 | N | | |
| 19 | 15 | 0127 | 12 | 15 | N | | |
| 20 | 15 | 0123 | 9 | | G | 0128 | 0123 |
| 21 | 16 | 0128 | 16 | | G | | |
| 22 | 17 | 0128 | 16 | 17 | N | | |
| 23 | 18 | 0123 | 9 | 18 | N | | |
| 24 | 18 | 0116 | 4 | | G | 0124 | 0116 |
| 25 | 19 | 0124 | 19 | | G | 0129 | 0124 |
| 26 | 20 | 0129 | 20 | | G | | |
| 27 | 21 | 0129 | 20 | 21 | N | | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 28 | 22 | 0124 | 19 | 22 | N | | |
| 29 | 23 | 0116 | 4 | 23 | N | | |
| 30 | 23 | 0113 | 3 | | G | 0117 | 0113 |
| 31 | 24 | 0117 | 24 | | G | 0136 | 0117 |
| 32 | 25 | 0136 | 25 | | G | 0137 | 0136 |
| 33 | 26 | 0137 | 26 | | G | | |
| 34 | 27 | 0137 | 26 | 27 | N | | |
| 35 | 28 | 0136 | 25 | 28 | N | | |
| 36 | 29 | 0117 | 24 | 29 | N | | |
| 37 | 29 | 0113 | 3 | | G | 0135 | 0113 |
| 38 | 30 | 0135 | 30 | | G | | |
| 39 | 31 | 0135 | 30 | 31 | N | | |
| 40 | 32 | 0113 | 3 | 32 | N | | |
| 41 | 32 | 0112 | 2 | | G | 0114 | 0112 |
| 42 | 33 | 0114 | 33 | | G | 0118 | 0114 |
| 43 | 34 | 0118 | 34 | | G | 0120 | 0118 |
| 44 | 35 | 0120 | 35 | | G | | |
| 45 | 36 | 0120 | 35 | 36 | N | | |
| 46 | 37 | 0118 | 34 | 37 | N | | |
| 47 | 37 | 0114 | 33 | | G | 0119 | 0114 |
| 48 | 38 | 0119 | 38 | | G | | |
| 49 | 39 | 0119 | 38 | 39 | N | | |
| 50 | 40 | 0114 | 33 | 40 | N | | |
| 51 | 41 | 0112 | 2 | 41 | N | | |
| 52 | 42 | 0110 | 1 | 42 | N | | |
| 53 | 43 | 0291 | 43 | | G | 0111 | 0291 |
| 54 | 44 | 0111 | 44 | | G | 0115 | 0111 |
| 55 | 45 | 0115 | 45 | | G | 0130 | 0115 |
| 56 | 46 | 0130 | 46 | | G | 0131 | 0130 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **57** | **47** | **0131** | **47** | | **G** | | |
| **58** | **48** | **0131** | **47** | **48** | **N** | | |
| **59** | **49** | **0130** | **46** | **49** | **N** | | |
| **60** | **50** | **0115** | **45** | **50** | **N** | | |
| **61** | **51** | **0111** | **44** | **51** | **N** | | |
| **62** | **51** | **0291** | **43** | | **G** | **0292** | **0291** |
| **63** | **52** | **0292** | **52** | | **G** | **1005** | **0292** |
| **64** | **53** | **1005** | **53** | | **G** | | |
| **65** | **54** | **1005** | **53** | **54** | **N** | | |
| **66** | **55** | **0292** | **52** | **55** | **N** | | |
| **67** | **55** | **0291** | **43** | | **G** | **0293** | **0291** |
| **68** | **56** | **0293** | **56** | | **G** | **0294** | **0293** |
| **69** | **57** | **0294** | **57** | | **G** | | |
| **70** | **58** | **0294** | **57** | **58** | **N** | | |
| **71** | **59** | **0293** | **56** | **59** | **N** | | |
| **72** | **60** | **0291** | **43** | **60** | **N** | | |

**a. Lista final (grafo de orden topológico)**

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0291 | 0293 | 0294 | 0292 | 1005 | 0111 | 0115 | 0130 | 0131 | 0110 | 0112 | 0114 | 0119 | 0118 | 0120 |

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0113 | 0135 | 0117 | 0136 | 0137 | 0116 | 0124 | 0129 | 0123 | 0128 | 0127 | 0141 | 0126 | 0122 | 0121 |

## 02. Búsqueda de camino más corto

Para identificar los caminos más cortos alcanzables a partir del vértice *0110* usamos el algoritmo *Single source shortest paths in directed acyclic graphs*, como se indica en el libro de *Cormen et al (2022)*.

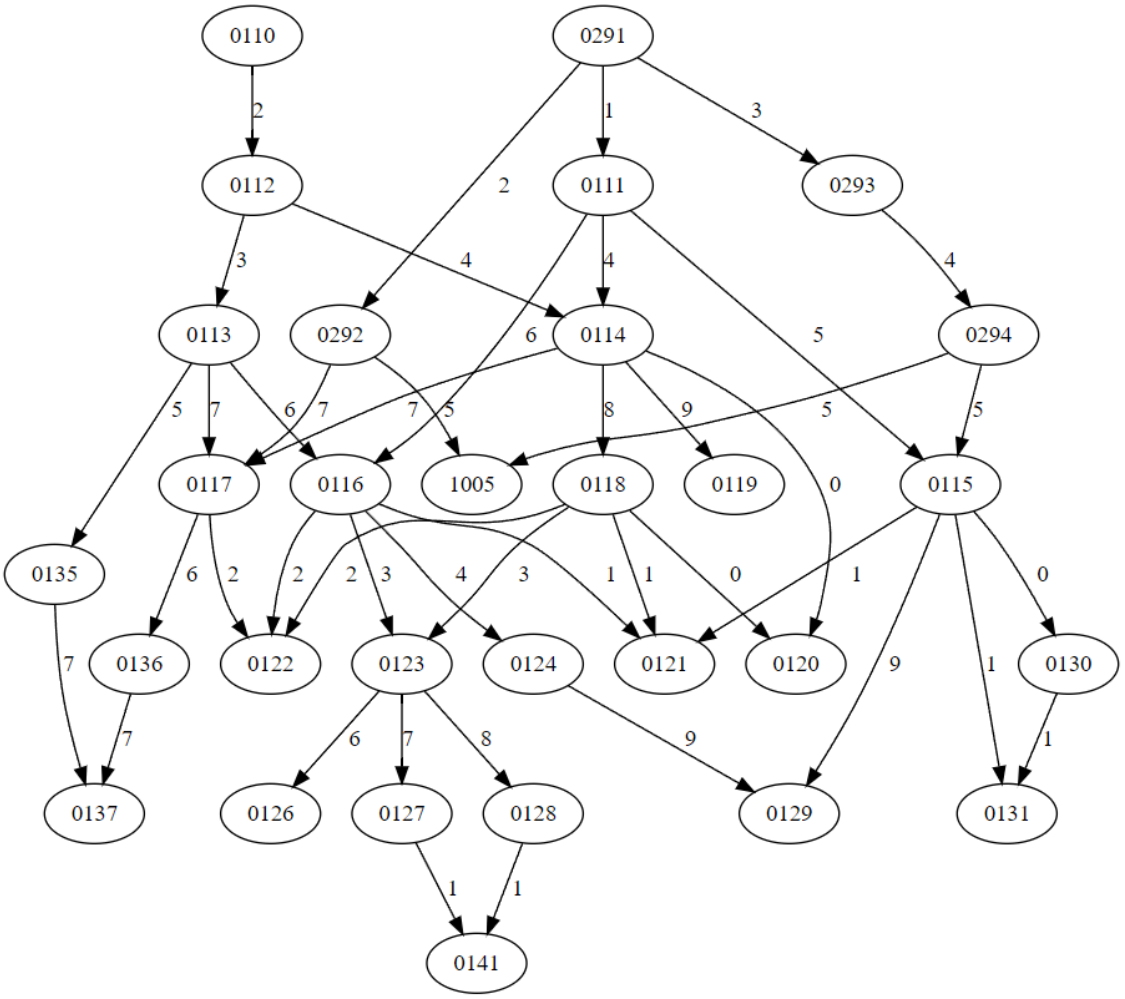
```
DAG-SHORTEST-PATHS(G, w, s)
1  topologically sort the vertices of G
2  INITIALIZE-SINGLE-SOURCE(G, s)
3  for each vertex u ∈ G.V, taken in topologically sorted order
4       for each vertex v in G.Adj[u]
5            RELAX(u, v, w)
```

Donde:

```
INITIALIZE-SINGLE-SOURCE(G, s)
1  for each vertex v ∈ G.V
2       v.d = ∞
3       v.π = NIL
4  s.d = 0
```

```
RELAX(u, v, w)
1  if v.d > u.d + w(u, v)
2       v.d = u.d + w(u, v)
3       v.π = u
```

Para esto necesitamos el grafo con pesos:



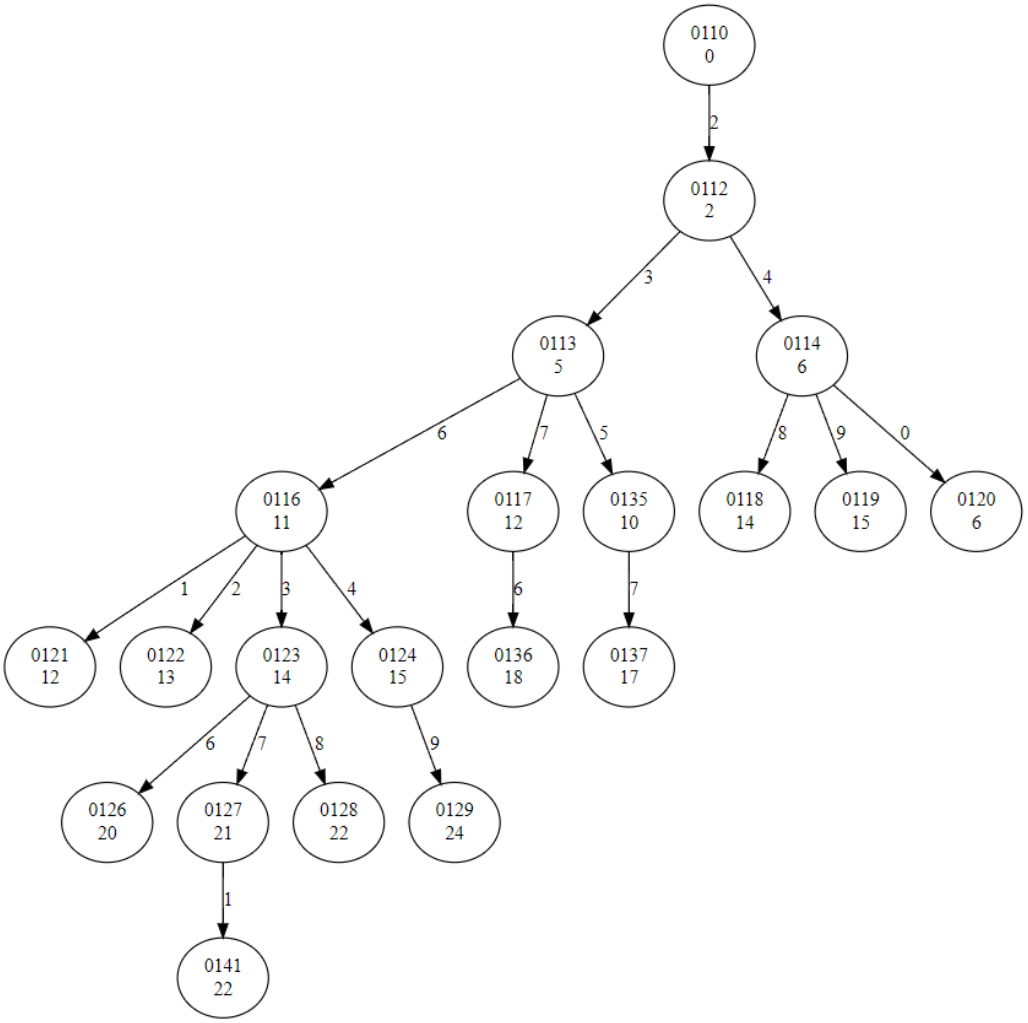

Para simplificar la tabla con las operaciones para obtener el árbol de camino más corto, se sabe que no hay forma de que haya un camino a los vértices anteriores al de partida en la lista de orden topológico, por lo cuál éstos quedan en infinito:

| 0291 | 0293 | 0294 | 0292 | 1005 | 0111 | 0115 | 0130 | 0131 |
|---|---|---|---|---|---|---|---|---|
| ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ |

El color amarillo representa el vértice al que se está buscando cada vértice adyacente para ejecutar el método de *RELAX(U, V, W)*.
El color rojo representa cuando se encontró otra ruta hacia ese vértice; sin embargo, no era más corta a la que ya estaba.
El color azul representa cuando se encontró otra ruta hacia ese vértice; y esta si era más corta, por lo cual se cambia.
Y la última columna hace referencia al padre del vértice amarillo.

| 01 10 | 01 12 | 01 14 | 01 19 | 01 18 | 01 20 | 01 13 | 01 35 | 01 17 | 01 36 | 01 37 | 01 16 | 01 24 | 01 29 | 01 23 | 01 28 | 01 27 | 01 41 | 01 26 | 01 22 | 01 21 | $\pi$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | nil |
| 0 | 2 | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | nil |
| 0 | 2 | 6 | | | | | | | | | | | | | | | | | | | 01 10 |
| 0 | 2 | 6 | 15 | 14 | 6 | 5 | | 13 | | | | | | | | | | | | | 01 12 |
| 0 | 2 | 6 | 15 | 14 | 6 | 5 | | 13 | | | | | | | | | | | | | 01 14 |
| 0 | 2 | 6 | 15 | 14 | 6 | 5 | | 13 | | | | | | 17 | | | | | 16 | 15 | 01 14 |
| 0 | 2 | 6 | 15 | 14 | 6 | 5 | | 13 | | | | | | 17 | | | | | 16 | 15 | 01 20 |
| 0 | 2 | 6 | 15 | 14 | 6 | 5 | 10 | 12 | | | 11 | | | 17 | | | | | 16 | 15 | 01 12 |
| 0 | 2 | 6 | 15 | 14 | 6 | 5 | 10 | 12 | | 17 | 11 | | | 17 | | | | | 16 | 15 | 01 13 |
| 0 | 2 | 6 | 15 | 14 | 6 | 5 | 10 | 12 | 18 | 17 | 11 | | | 17 | | | | | 14 | 15 | 01 13 |
| 0 | 2 | 6 | 15 | 14 | 6 | 5 | 10 | 12 | 18 | 17 | 11 | | | 17 | | | | | 14 | 15 | 01 17 |
| 0 | 2 | 6 | 15 | 14 | 6 | 5 | 10 | 12 | 18 | 17 | 11 | | | 17 | | | | | 14 | 15 | 01 35 |
| 0 | 2 | 6 | 15 | 14 | 6 | 5 | 10 | 12 | 18 | 17 | 11 | 15 | | 14 | | | | | 13 | 12 | 01 13 |
| 0 | 2 | 6 | 15 | 14 | 6 | 5 | 10 | 12 | 18 | 17 | 11 | 15 | 24 | 14 | | | | | 13 | 12 | 01 16 |
| 0 | 2 | 6 | 15 | 14 | 6 | 5 | 10 | 12 | 18 | 17 | 11 | 15 | 24 | 14 | | | | | 13 | 12 | 01 24 |
| 0 | 2 | 6 | 15 | 14 | 6 | 5 | 10 | 12 | 18 | 17 | 11 | 15 | 24 | 14 | 22 | 21 | | 20 | 13 | 12 | 01 16 |
| 0 | 2 | 6 | 15 | 14 | 6 | 5 | 10 | 12 | 18 | 17 | 11 | 15 | 24 | 14 | 22 | 21 | 24 | 20 | 13 | 12 | 01 23 |
| 0 | 2 | 6 | 15 | 14 | 6 | 5 | 10 | 12 | 18 | 17 | 11 | 15 | 24 | 14 | 22 | 21 | 22 | 20 | 13 | 12 | 01 23 |
| 0 | 2 | 6 | 15 | 14 | 6 | 5 | 10 | 12 | 18 | 17 | 11 | 15 | 24 | 14 | 22 | 21 | 22 | 20 | 13 | 12 | 01 27 |
| 0 | 2 | 6 | 15 | 14 | 6 | 5 | 10 | 12 | 18 | 17 | 11 | 15 | 24 | 14 | 22 | 21 | 22 | 20 | 13 | 12 | 01 23 |
| 0 | 2 | 6 | 15 | 14 | 6 | 5 | 10 | 12 | 18 | 17 | 11 | 15 | 24 | 14 | 22 | 21 | 22 | 20 | 13 | 12 | 01 16 |
| 0 | 2 | 6 | 15 | 14 | 6 | 5 | 10 | 12 | 18 | 17 | 11 | 15 | 24 | 14 | 22 | 21 | 22 | 20 | 13 | 12 | 01 16 |

Por lo tanto, el árbol de caminos más cortos construido a partir del algoritmo *Single source shortest paths in directed acyclic graphs* iniciando en el nodo *0110* es:



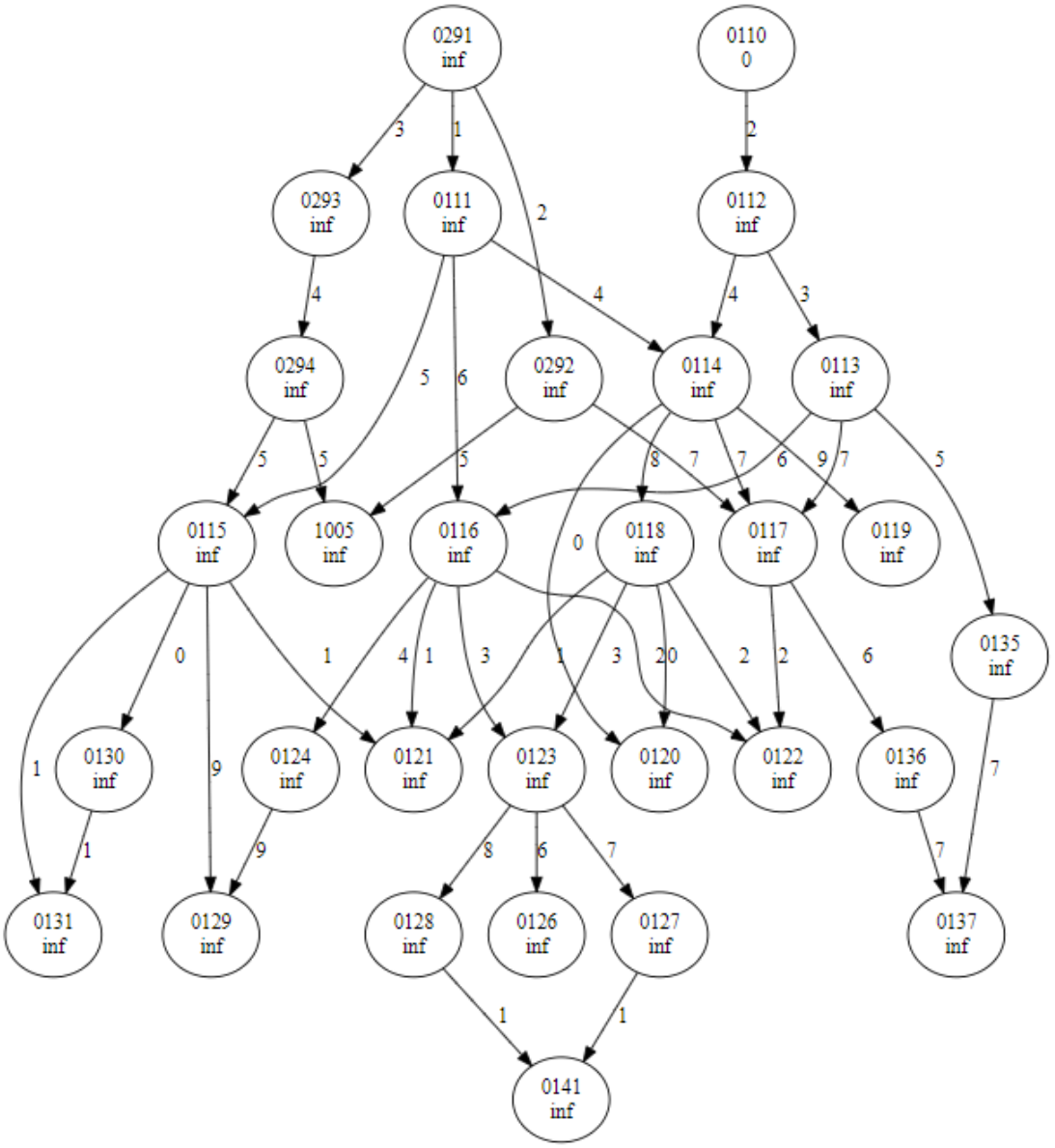
### 03. Algoritmo de Dijkstra

Ahora se debe encontrar el camino más corto aplicando el algoritmo de Dijkstra:

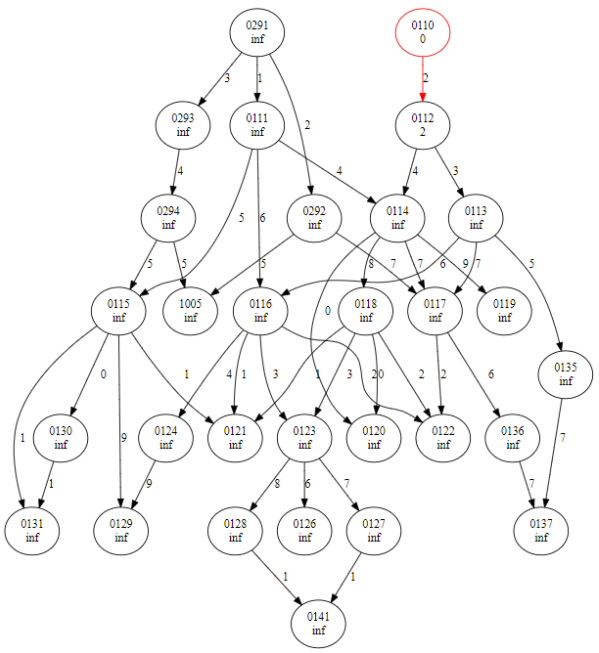
```
DIJKSTRA(G, w, s)
 1  INITIALIZE-SINGLE-SOURCE(G, s)
 2  S = ∅
 3  Q = ∅
 4  for each vertex u ∈ G.V
 5      INSERT(Q, u)
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      S = S ∪ {u}
 9      for each vertex v in G.Adj[u]
10          RELAX(u, v, w)
11          if the call of RELAX decreased v.d
12              DECREASE-KEY(Q, v, v.d)
```

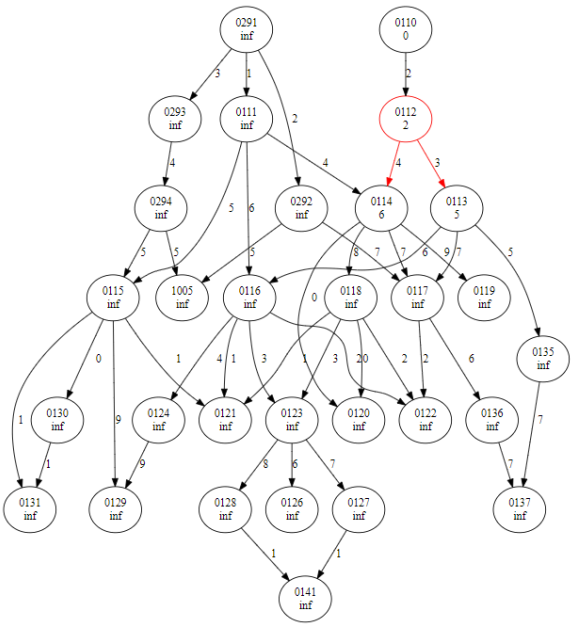
Para esto también se usa el grafo con pesos del punto anterior.

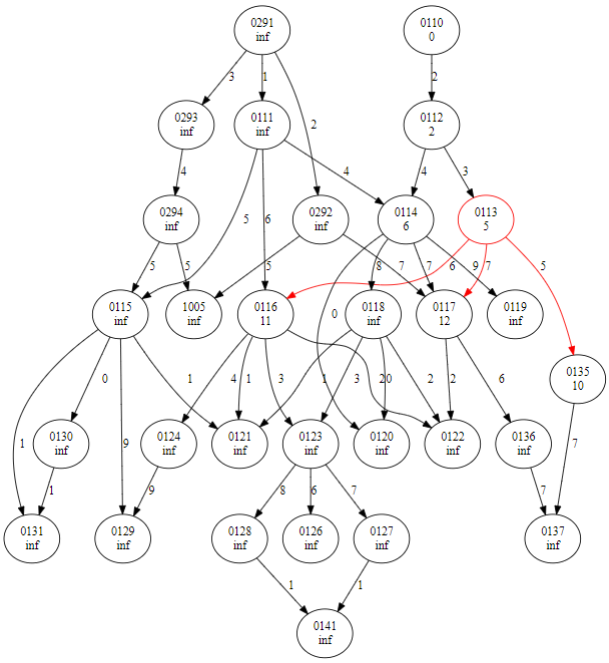
En este caso, se puede pensar en el algoritmo de Dijkstra como una generalización del *Breadth-first Search (BFS)* para grafos con pesos. Una ola emana de la fuente, y la primera vez que una ola llega a un vértice, una nueva ola emana de ese vértice. Mientras que en el *Breadth-first Search (BFS)* cada ola tarda una unidad de tiempo en atravesar una arista, en un grafo ponderado el tiempo que tarda una ola en atravesar una arista viene dado por el peso de la arista. Dado que el camino más corto en un grafo ponderado puede no tener el menor número de aristas, no basta con una simple cola *first-in, first-out* para elegir el siguiente vértice desde el que enviar una ola.
Resolviendo desde un enfoque visual tenemos que, después de las líneas 1 a 3 del algoritmo, el grafo corresponde a:

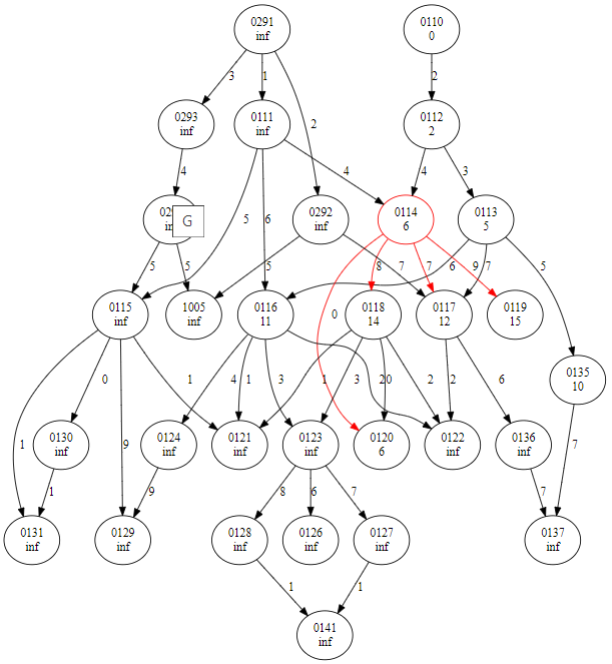


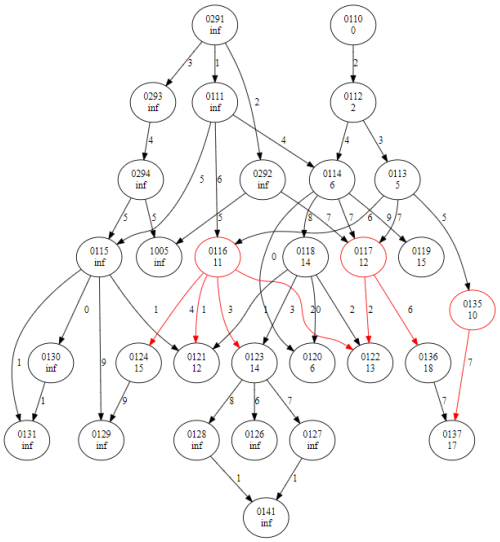

Luego, para cada iteración el grafo se comporta de la forma:

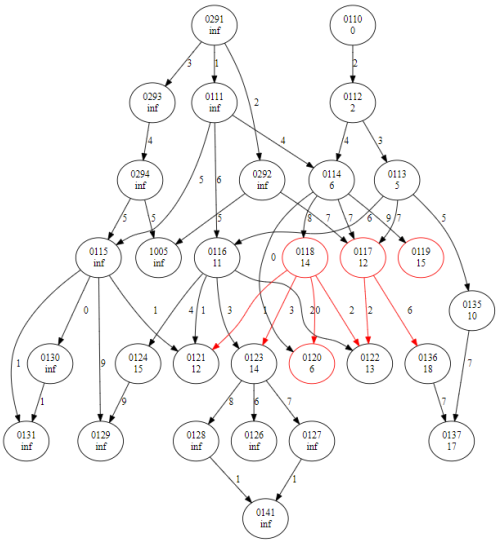


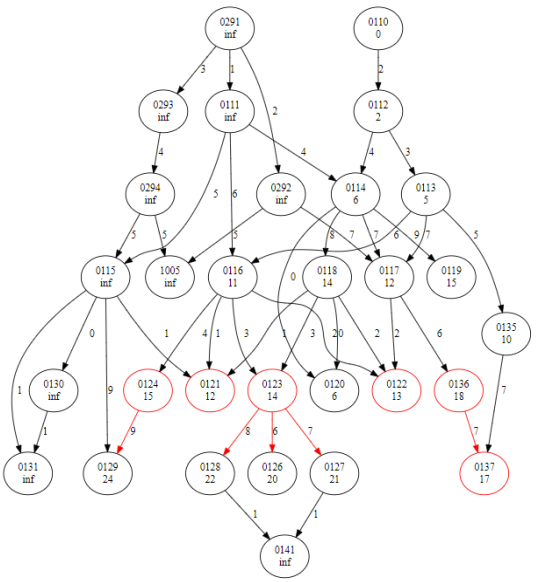


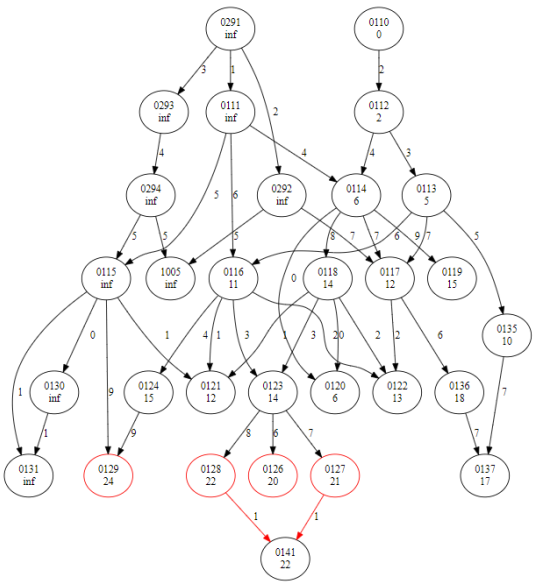

0291 inf
0110 0
0293 inf
0111 inf
0112 2
0294 inf
0292 inf
0114 6
0113 5
0115 inf
1005 inf
0116 11
0118 inf
0117 12
0119 inf
0135 10
0130 inf
0124 inf
0121 inf
0123 inf
0120 inf
0122 inf
0136 inf
0131 inf
0129 inf
0128 inf
0126 inf
0127 inf
0137 inf
0141 inf


0291 inf
0110 0
0293 inf
0111 inf
0112 2
0292 inf
0114 6
0113 5
0115 inf
1005 inf
0116 11
0118 14
0117 12
0119 15
0135 10
0130 inf
0124 inf
0121 inf
0123 inf
0120 6
0122 inf
0136 inf
0131 inf
0129 inf
0128 inf
0126 inf
0127 inf
0137 inf
0141 inf


0291 inf
0110 0
0293 inf
0111 inf
0112 2
0294 inf
0292 inf
0114 6
0113 5
0115 inf
1005 inf
0116 11
0118 14
0117 12
0119 15
0135 10
0130 inf
0124 15
0121 12
0123 14
0120 6
0122 13
0136 18
0131 inf
0129 inf
0128 inf
0126 inf
0127 inf
0137 17
0141 inf


0291 inf
0110 0
0293 inf
0111 inf
0112 2
0294 inf
0292 inf
0114 6
0113 5
0115 inf
1005 inf
0116 11
0118 14
0117 12
0119 15
0135 10
0130 inf
0124 15
0121 12
0123 14
0120 6
0122 13
0136 18
0131 inf
0129 inf
0128 inf
0126 inf
0127 inf
0137 17
0141 inf


0291 inf
0110 0
0293 inf
0111 inf
0112 2
0294 inf
0292 inf
0114 6
0113 5
0115 inf
1005 inf
0116 11
0118 14
0117 12
0119 15
0135 10
0130 inf
0124 15
0121 12
0123 14
0120 6
0122 13
0136 18
0131 inf
0129 24
0128 22
0126 20
0127 21
0137 17
0141 inf


0291 inf
0110 0
0293 inf
0111 inf
0112 2
0294 inf
0292 inf
0114 6
0113 5
0115 inf
1005 inf
0116 11
0118 14
0117 12
0119 15
0135 10
0130 inf
0124 15
0121 12
0123 14
0120 6
0122 13
0136 18
0131 inf
0129 24
0128 22
0126 20
0127 21
0137 17
0141 22

Por lo tanto, el árbol de caminos más cortos construido a partir del algoritmo de Dijkstra iniciando en el nodo *0110* es: