Algoritmos y estructuras de datos

Caminos de costo mínimo

Francisco Javier Zaragoza Martínez

Universidad Autónoma Metropolitana Unidad Azcapotzalco
Departamento de Sistemas

Universidad Autónoma Metropolitana
Casa abierta al tiempo Azcapotzalco

Posgrado en Optimización

11 de junio de 2021

Citas

Joan Manuel Serrat

Caminante no hay camino, se hace camino al andar.

Confucio

El camino no está lejos del humano. Si los humanos intentan seguir un camino lejano a las indicaciones comunes de la conciencia, ese camino no se puede considerar el camino.

Jiddu Krishnamurti

La verdad es una tierra sin caminos y no te le puedes acercar por ningún camino.

Jean de La Fontaine

Todos los caminos llevan a Roma, pero nuestros antagonistas piensan que debieramos escoger caminos distintos.

Redes y redes dirigidas

Definiciones

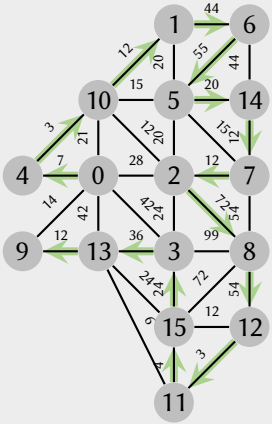
Las gráficas y digráficas que aparecen en la práctica vienen usualmente acompañadas de información numérica en sus vértices (importancia, prioridad), en sus aristas (longitud, duración) y en sus arcos (costo, capacidad). En general, lo llamaremos vector de costos.

Una red consta de una gráfica $G = (V, E)$ y al menos un vector de costos en sus vértices o en sus aristas. Una red dirigida consta de una digráfica $D = (V, A)$ y al menos un vector de costos en sus vértices o en sus arcos.

Los problemas de diseño de redes se tratan de encontrar una subred H de G (o de D) que sea de costo mínimo (o máximo) y que al mismo tiempo satisfaga alguna propiedad deseada. El costo de H está dado por la suma de los costos de sus vértices, aristas o arcos.

Gráficas

Caminos



Dados dos vértices u y v de una gráfica G, un camino de u a v es una secuencia de vértices $u = w_0, w_1, \ldots, w_k = v$ tal que cada pareja w_i, w_{i+1} de vértices consecutivos está unida por una arista de G.

Si G forma parte de una red con costos en las aristas, entonces el costo de cualquier camino de G es la suma de los costos de sus aristas. Si todos los costos son positivos, entonces un camino de u a v de costo mínimo es simple, es decir, todos sus vértices $w_0, w_1, \ldots, w_k$ son distintos.

El problema es muy difícil si se permiten costos negativos y se busca un camino simple de costo mínimo de u a v.

Caminos de costo mínimo

Algoritmo de Dijkstra

El algoritmo de Dijkstra es un tipo de búsqueda por prioridad variable. Al principio, el vértice inicial u tendrá prioridad $p_u = 0$ y los otros vértices $v \neq u$ tendrán prioridad $p_v = +\infty$. Cada que visitemos un vértice u actualizaremos la prioridad de sus vecinos v con el mínimo de p_v y $p_u + c_{uv}$. Esto hace que al final p_v sea el costo mínimo de un camino de u a v. Dichos caminos más cortos se pueden recuperar del arreglo `arista`.

```
void actDijkstra(red r, int u, int p[], int visto[], int arista[])
{
  for (int v = 0; v < r.n; v++)
    if (r.a[u][v]==1 && visto[v]==0 && p[u]+r.c[u][v]<p[v]) {
      p[v] = p[u] + r.c[u][v];
      arista[v] = u;     // ultimo vertice en actualizar a v
    }
}
```

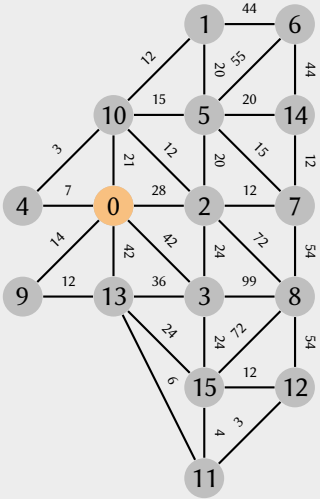
Caminos de costo mínimo

Implementación del algoritmo de Dijkstra con matriz de adyacencia

```
void dijkstra(red r, int u, int arista[], int p[]) {
  int *visto = (int *) calloc(r.n, sizeof(int));
  arista[u] = u;                    // u no genera una arista
  for (int v = 0; v < r.n; v++)
    p[v] = (v == u) ? 0 : INT_MAX;    // prioridad inicial
  for (int orden = 1; orden <= r.n; orden++) {
    visto[u] = orden;
    actDijkstra(r, u, p, visto, arista);
    u = minPrioridad(r, p, visto);
  }
  free(visto);
}
```

Caminos de costo mínimo

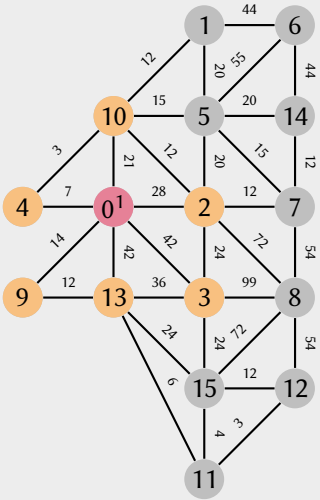
Ejemplo del algoritmo de Dijkstra



u	v[u]	p[u]	a[u]
0	0	0	0
1	0	$+\infty$	×
2	0	$+\infty$	×
3	0	$+\infty$	×
4	0	$+\infty$	×
5	0	$+\infty$	×
6	0	$+\infty$	×
7	0	$+\infty$	×
8	0	$+\infty$	×
9	0	$+\infty$	×
10	0	$+\infty$	×
11	0	$+\infty$	×
12	0	$+\infty$	×
13	0	$+\infty$	×
14	0	$+\infty$	×
15	0	$+\infty$	×

Caminos de costo mínimo

Ejemplo del algoritmo de Dijkstra

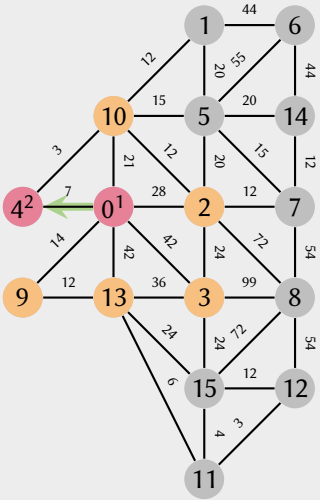


u	v[u]	p[u]	a[u]
0	1	0	0
1	0	$+\infty$	$\times$
2	0	28	0
3	0	42	0
4	0	7	0
5	0	$+\infty$	$\times$
6	0	$+\infty$	$\times$
7	0	$+\infty$	$\times$
8	0	$+\infty$	$\times$
9	0	14	0
10	0	21	0
11	0	$+\infty$	$\times$
12	0	$+\infty$	$\times$
13	0	42	0
14	0	$+\infty$	$\times$
15	0	$+\infty$	$\times$

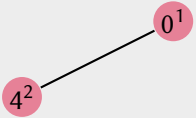
0^1

Caminos de costo mínimo

Ejemplo del algoritmo de Dijkstra

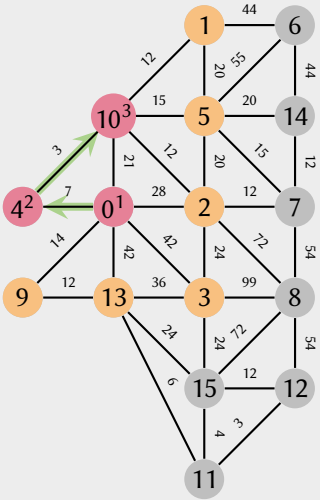


u	v[u]	p[u]	a[u]
0	1	0	0
1	0	$+\infty$	×
2	0	28	0
3	0	42	0
4	2	7	0
5	0	$+\infty$	×
6	0	$+\infty$	×
7	0	$+\infty$	×
8	0	$+\infty$	×
9	0	14	0
10	0	10	4
11	0	$+\infty$	×
12	0	$+\infty$	×
13	0	42	0
14	0	$+\infty$	×
15	0	$+\infty$	×

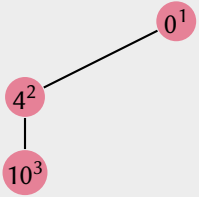


Caminos de costo mínimo

Ejemplo del algoritmo de Dijkstra

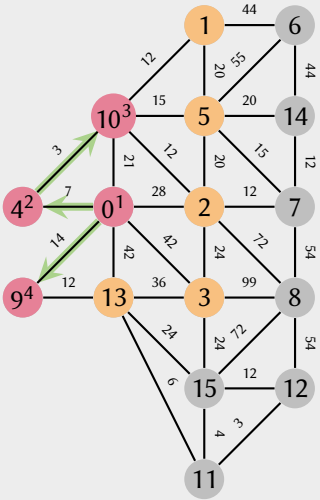


u	v[u]	p[u]	a[u]
0	1	0	0
1	0	22	10
2	0	22	10
3	0	42	0
4	2	7	0
5	0	25	10
6	0	$+\infty$	$\times$
7	0	$+\infty$	$\times$
8	0	$+\infty$	$\times$
9	0	14	0
10	3	10	4
11	0	$+\infty$	$\times$
12	0	$+\infty$	$\times$
13	0	42	0
14	0	$+\infty$	$\times$
15	0	$+\infty$	$\times$

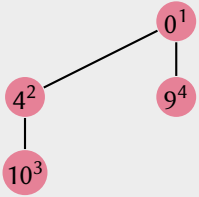


Caminos de costo mínimo

Ejemplo del algoritmo de Dijkstra

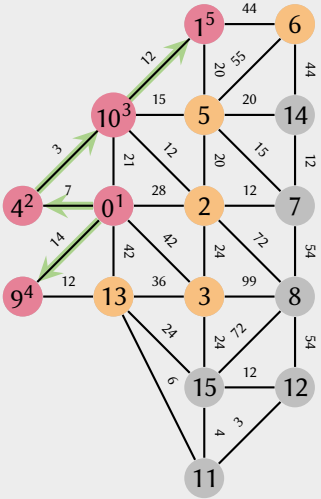


u	v[u]	p[u]	a[u]
0	1	0	0
1	0	22	10
2	0	22	10
3	0	42	0
4	2	7	0
5	0	25	10
6	0	$+\infty$	×
7	0	$+\infty$	×
8	0	$+\infty$	×
9	4	14	0
10	3	10	4
11	0	$+\infty$	×
12	0	$+\infty$	×
13	0	26	9
14	0	$+\infty$	×
15	0	$+\infty$	×

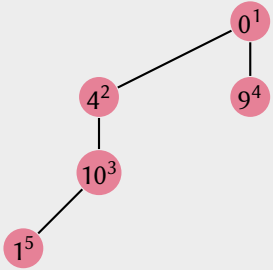


Caminos de costo mínimo

Ejemplo del algoritmo de Dijkstra

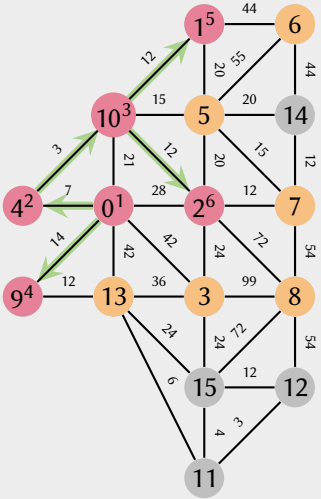


u	v[u]	p[u]	a[u]
0	1	0	0
1	5	22	10
2	0	22	10
3	0	42	0
4	2	7	0
5	0	25	10
6	0	66	1
7	0	$+\infty$	$\times$
8	0	$+\infty$	$\times$
9	4	14	0
10	3	10	4
11	0	$+\infty$	$\times$
12	0	$+\infty$	$\times$
13	0	26	9
14	0	$+\infty$	$\times$
15	0	$+\infty$	$\times$

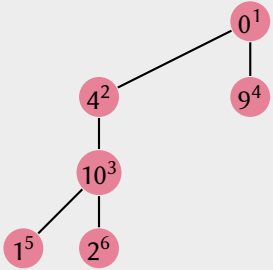


Caminos de costo mínimo

Ejemplo del algoritmo de Dijkstra

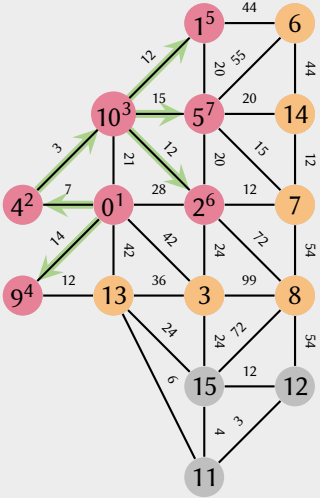


u	v[u]	p[u]	a[u]
0	1	0	0
1	5	22	10
2	6	22	10
3	0	42	0
4	2	7	0
5	0	25	10
6	0	66	1
7	0	34	2
8	0	94	2
9	4	14	0
10	3	10	4
11	0	$+\infty$	×
12	0	$+\infty$	×
13	0	26	9
14	0	$+\infty$	×
15	0	$+\infty$	×

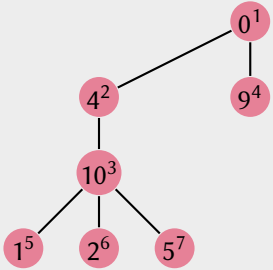


Caminos de costo mínimo

Ejemplo del algoritmo de Dijkstra

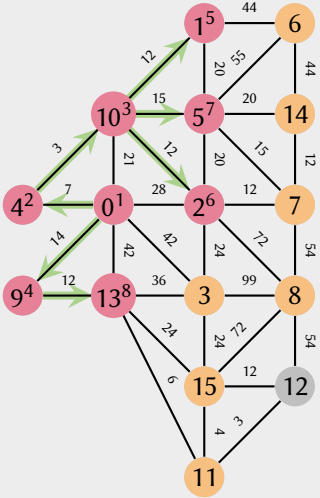


u	v[u]	p[u]	a[u]
0	1	0	0
1	5	22	10
2	6	22	10
3	0	42	0
4	2	7	0
5	7	25	10
6	0	66	1
7	0	34	2
8	0	94	2
9	4	14	0
10	3	10	4
11	0	$+\infty$	$\times$
12	0	$+\infty$	$\times$
13	0	26	9
14	0	45	5
15	0	$+\infty$	$\times$

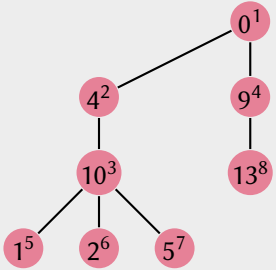


Caminos de costo mínimo

Ejemplo del algoritmo de Dijkstra

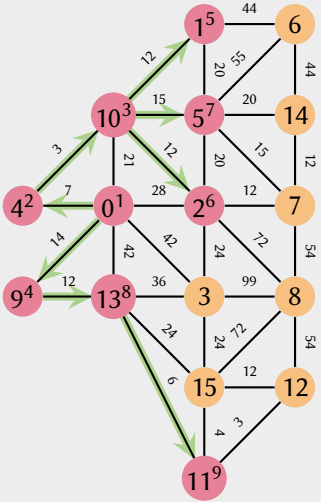


u	v[u]	p[u]	a[u]
0	1	0	0
1	5	22	10
2	6	22	10
3	0	42	0
4	2	7	0
5	7	25	10
6	0	66	1
7	0	34	2
8	0	94	2
9	4	14	0
10	3	10	4
11	0	32	13
12	0	$+\infty$	$\times$
13	8	26	9
14	0	45	5
15	0	50	13

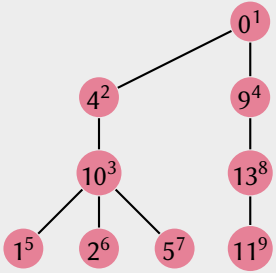


Caminos de costo mínimo

Ejemplo del algoritmo de Dijkstra

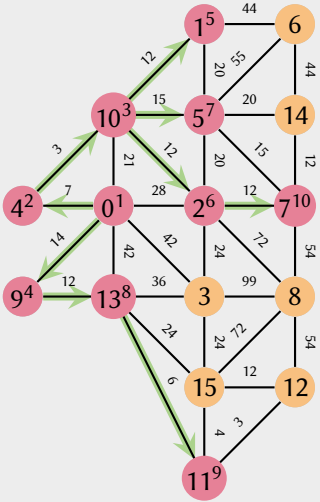


u	v[u]	p[u]	a[u]
0	1	0	0
1	5	22	10
2	6	22	10
3	0	42	0
4	2	7	0
5	7	25	10
6	0	66	1
7	0	34	2
8	0	94	2
9	4	14	0
10	3	10	4
11	9	32	13
12	0	35	11
13	8	26	9
14	0	45	5
15	0	36	11

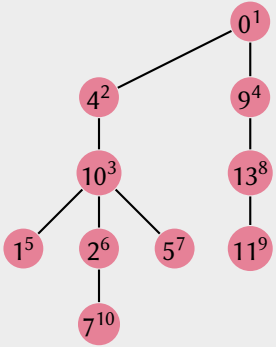


Caminos de costo mínimo

Ejemplo del algoritmo de Dijkstra

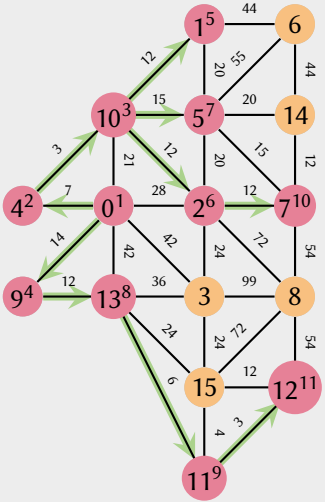


u	v[u]	p[u]	a[u]
0	1	0	0
1	5	22	10
2	6	22	10
3	0	42	0
4	2	7	0
5	7	25	10
6	0	66	1
7	10	34	2
8	0	88	7
9	4	14	0
10	3	10	4
11	9	32	13
12	0	35	11
13	8	26	9
14	0	45	5
15	0	36	11

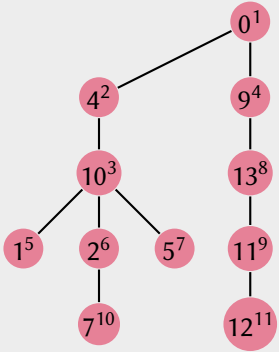


Caminos de costo mínimo

Ejemplo del algoritmo de Dijkstra

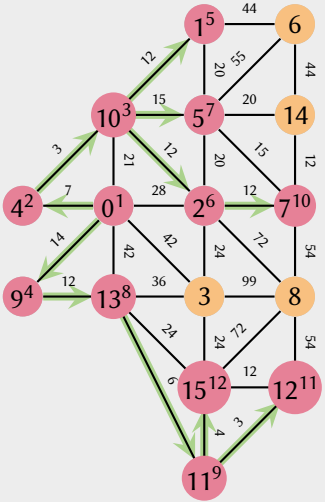


u	v[u]	p[u]	a[u]
0	1	0	0
1	5	22	10
2	6	22	10
3	0	42	0
4	2	7	0
5	7	25	10
6	0	66	1
7	10	34	2
8	0	88	7
9	4	14	0
10	3	10	4
11	9	32	13
12	11	35	11
13	8	26	9
14	0	45	5
15	0	36	11

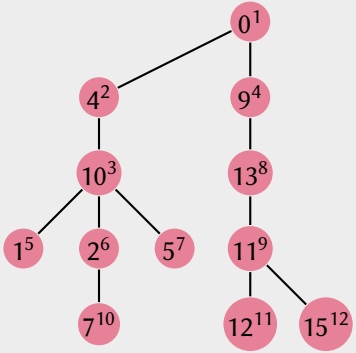


Caminos de costo mínimo

Ejemplo del algoritmo de Dijkstra

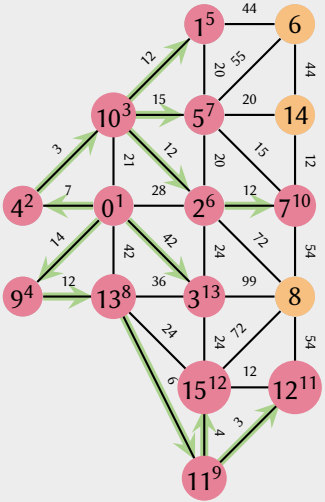


u	v[u]	p[u]	a[u]
0	1	0	0
1	5	22	10
2	6	22	10
3	0	42	0
4	2	7	0
5	7	25	10
6	0	66	1
7	10	34	2
8	0	88	7
9	4	14	0
10	3	10	4
11	9	32	13
12	11	35	11
13	8	26	9
14	0	45	5
15	12	36	11

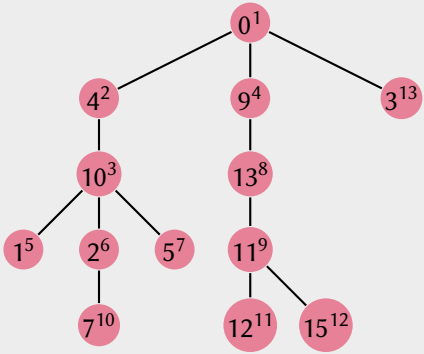


Caminos de costo mínimo

Ejemplo del algoritmo de Dijkstra

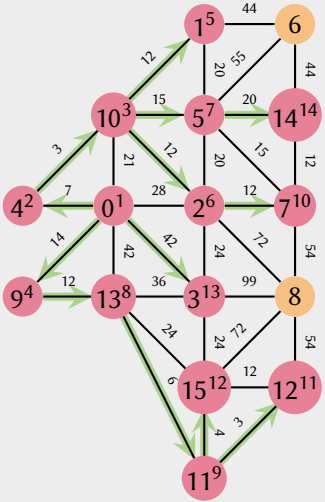


u	v[u]	p[u]	a[u]
0	1	0	0
1	5	22	10
2	6	22	10
3	13	42	0
4	2	7	0
5	7	25	10
6	0	66	1
7	10	34	2
8	0	88	7
9	4	14	0
10	3	10	4
11	9	32	13
12	11	35	11
13	8	26	9
14	0	45	5
15	12	36	11

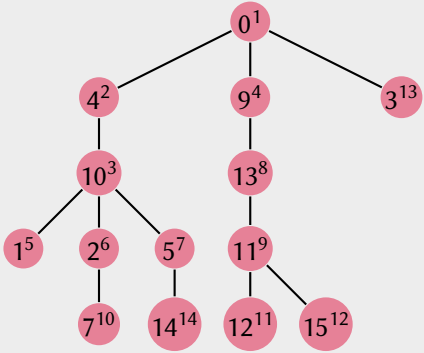


Caminos de costo mínimo

Ejemplo del algoritmo de Dijkstra

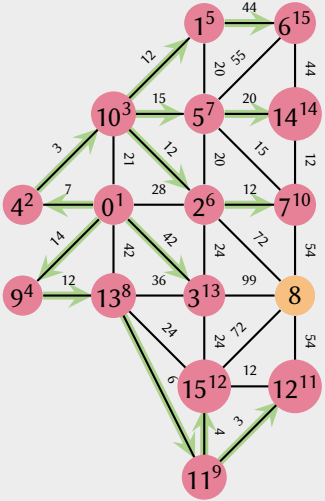


u	v[u]	p[u]	a[u]
0	1	0	0
1	5	22	10
2	6	22	10
3	13	42	0
4	2	7	0
5	7	25	10
6	0	66	1
7	10	34	2
8	0	88	7
9	4	14	0
10	3	10	4
11	9	32	13
12	11	35	11
13	8	26	9
14	14	45	5
15	12	36	11

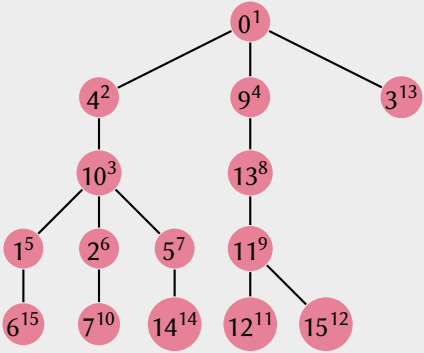


Caminos de costo mínimo

Ejemplo del algoritmo de Dijkstra

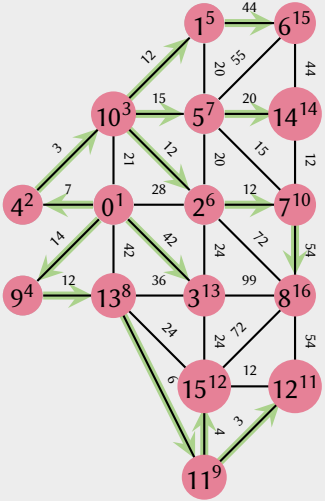


u	v[u]	p[u]	a[u]
0	1	0	0
1	5	22	10
2	6	22	10
3	13	42	0
4	2	7	0
5	7	25	10
6	15	66	1
7	10	34	2
8	0	88	7
9	4	14	0
10	3	10	4
11	9	32	13
12	11	35	11
13	8	26	9
14	14	45	5
15	12	36	11

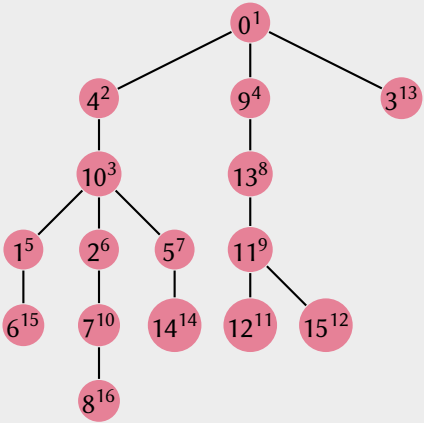


Caminos de costo mínimo

Ejemplo del algoritmo de Dijkstra



u	v[u]	p[u]	a[u]
0	1	0	0
1	5	22	10
2	6	22	10
3	13	42	0
4	2	7	0
5	7	25	10
6	15	66	1
7	10	34	2
8	16	88	7
9	4	14	0
10	3	10	4
11	9	32	13
12	11	35	11
13	8	26	9
14	14	45	5
15	12	36	11



Caminos de costo mínimo

Recuperación de los caminos

Suponiendo que ya se ejecutó el algoritmo de Dijkstra desde el vértice u, el siguiente código sigue las aristas del camino más corto de u a v de la última a la primera.

```
void camino(int u, int v, int arista[]) {
  while (v != u)
    v = arista[v]; // arista[v] -> v
}
```

En una gráfica esto no es importante (pues las conexiones son simétricas) pero en una digráfica hay que recordar que el arco va de `arista[v]` a `v`.

Caminos de costo mínimo

Todas las parejas de vértices

En ocasiones se requieren obtener caminos de costo mínimo de cada vértice u a cada vértice v. Una forma de lograr esto es simplemente llamar al algoritmo de Dijkstra una vez desde cada vértice y almacenar las n respuestas en un arreglo de $n \times n$.

El algoritmo de Dijkstra hace n^2 pasos con una matriz de adyacencia, así que llamarlo n veces tardará n^3 pasos.

Afortunadamente hay una forma más sencilla que también hace n^3 pasos.

Caminos de costo mínimo

Algoritmo de Floyd

El algoritmo de Floyd está basado en la idea de que si para ir de u a v conviene pasar por w, entonces se hace. Hay que hacerlo en cierto orden, pero resulta en un código simple. Se asume que `r.c[u][v]` contiene un **`int`** muy grande (∞) cuando no hay arista de `u` a `v` (y `0` si `u==v`). Como estos **`int`** se sumarán, dicho **`int`** no debe superar a `INT_MAX/2`.

```
void floyd(red r) {
  for (int w = 0; w < r.n; w++)
    for (int u = 0; u < r.n; u++)
      for (int v = 0; v < r.n; v++)
        if (r.c[u][w] + r.c[w][v] < r.c[u][v])
          r.c[u][v] = r.c[u][w] + r.c[w][v];
}
```

Caminos de costo mínimo

Observaciones

1. Tanto el algoritmo de Dijkstra como el algoritmo de Floyd funcionan en digráficas.
2. El algoritmo de Dijkstra se puede implementar también con listas de adyacencia, pero para obtener el vértice no visto de menor prioridad la búsqueda lineal sería muy lenta. En este caso se requiere una implementación de montículos que permita disminuir la prioridad de un vértice. Esto se puede hacer en $\approx \log_2 n$ pasos, por lo que el tiempo de ejecución del algoritmo es proporcional a $(n + m) \log_2 n$.
3. El algoritmo de Floyd se puede adaptar para que se puedan recuperar los caminos. En este caso se requiere almacenar el último w que mejoró un camino de u a v.
4. Ni el algoritmo de Dijkstra, ni el algoritmo de Floyd sirven para obtener caminos de costo máximo, ni caminos de costo mínimo cuando haya aristas de costo negativo de forma arbitraria. Hay algoritmos eficientes si no hay ciclos de costo negativo.