TALLER PROGCOMP: TRACK GRAFOS

ALGORITMO DE DIJKSTRA

Gabriel Carmona Tabja

Universidad Técnica Federico Santa María,
Università di Pisa

September 29, 2024

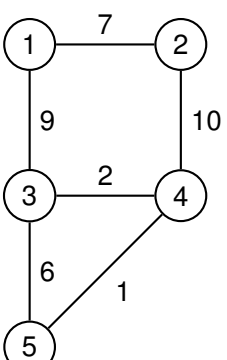
Part I

GRAFOS CON PESO

Grafos con Peso

Definición

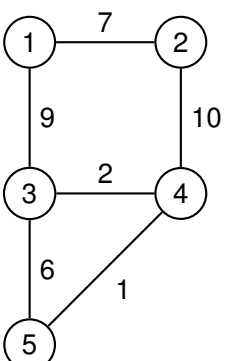
Un grafo con peso es un grafo donde la arista tiene un peso.



Problema Clásico

Camino más Corto

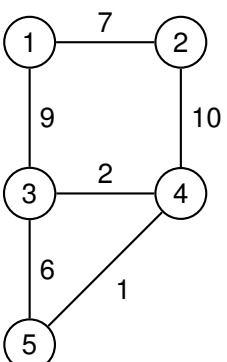
¿Cuál es el costo mínimo de ir de un nodo *s* hasta un nodo *t*?



Problema Clásico

Camino más Corto

¿Cuál es el costo mínimo de ir de un nodo *s* hasta un nodo *t*?



Costo mínimo de ir de 3 a 5 es ¡3!.

Part II

Algoritmo de Dijkstra

Algoritmo de Dijkstra

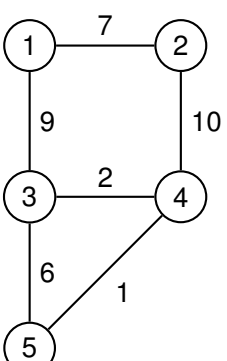
Definición

- Definido por Edsger W. Dijkstra en 1956.

Algoritmo de Dijkstra

Definición

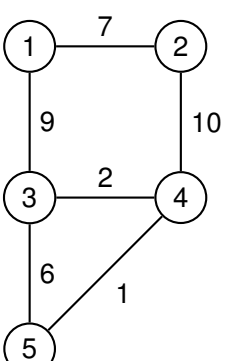
- Definido por Edsger W. Dijkstra en 1956.
- Calcula la distancia más corta de un nodo a todo el resto de nodos.



ALGORITMO DE DIJKSTRA

nodo	1	2	3	4	5
dist min	INF	INF	INF	INF	INF

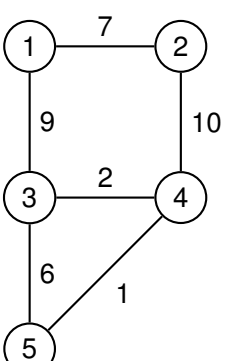
cola de prioridad (dist, nodo)	{}



ALGORITMO DE DIJKSTRA

nodo	1	2	3	4	5
dist min	INF	INF	INF	INF	0

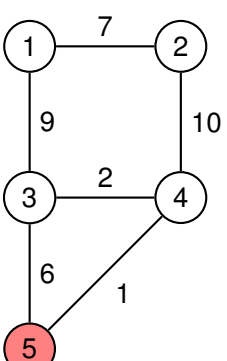
cola de prioridad (dist, nodo)	{(0, 5)}



ALGORITMO DE DIJKSTRA

nodo	1	2	3	4	5
dist min	INF	INF	INF	INF	0

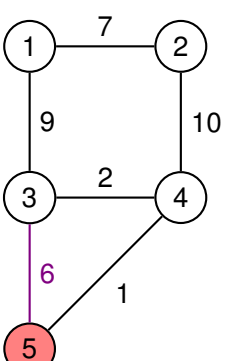
cola de prioridad (dist, nodo)	{}	actual: (0, 5)



ALGORITMO DE DIJKSTRA

nodo	1	2	3	4	5
dist min	INF	INF	INF	INF	0

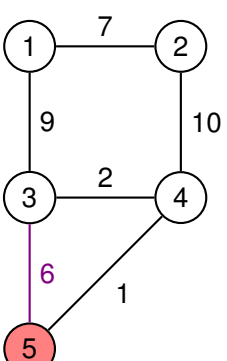
cola de prioridad (dist, nodo)	{}	actual: (0, 5)



ALGORITMO DE DIJKSTRA

nodo	1	2	3	4	5
dist min	`INF`	`INF`	`0 + 6 = 6`	`INF`	`0`

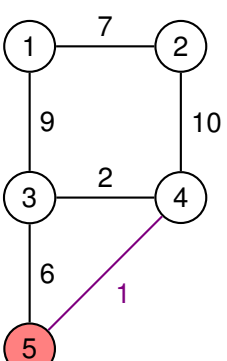
cola de prioridad (dist, nodo)	`{(0 + 6 = 6, 3)}`	actual: `(0, 5)`



ALGORITMO DE DIJKSTRA

nodo	1	2	3	4	5
dist min	INF	INF	6	INF	0

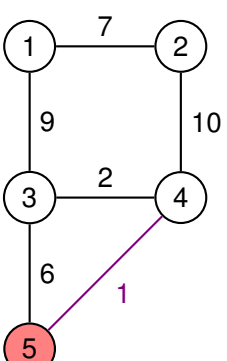
cola de prioridad (dist, nodo)	{(6, 3)}	actual: (0, 5)



ALGORITMO DE DIJKSTRA

nodo	1	2	3	4	5
dist min	`INF`	`INF`	6	`0 + 1 = 1`	0

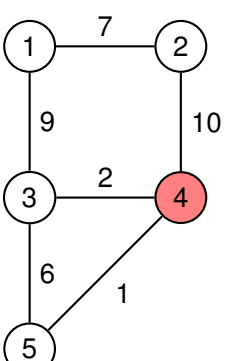
cola de prioridad (dist, nodo)	`{(0 + 1 = 1, 4), (6, 3)}`	actual: `(0, 5)`



ALGORITMO DE DIJKSTRA

nodo	1	2	3	4	5
dist min	INF	INF	6	1	0

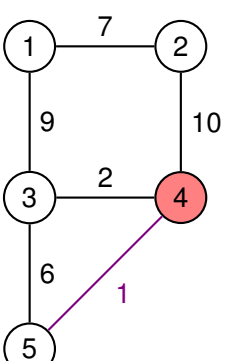
cola de prioridad (dist, nodo)	{(6, 3)}	actual: (1, 4)



ALGORITMO DE DIJKSTRA

nodo	1	2	3	4	5
dist min	INF	INF	6	1	0

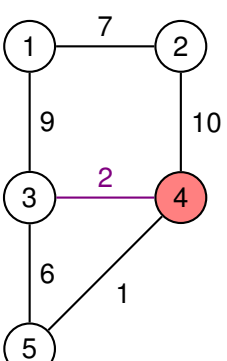
cola de prioridad (dist, nodo)	{(6, 3)}	actual: (1,4)



ALGORITMO DE DIJKSTRA

nodo	1	2	3	4	5
dist min	INF	INF	6	1	0

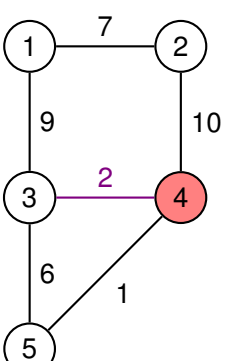
cola de prioridad (dist, nodo)	{(6, 3)}	actual: (1,4)



ALGORITMO DE DIJKSTRA

nodo	1	2	3	4	5
dist min	`INF`	`INF`	`1 + 2 = 3`	`1`	`0`

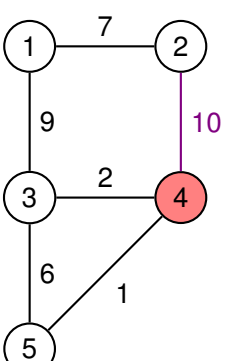
cola de prioridad (dist, nodo)	`{(1 + 2 = 3, 3), (6, 3)}`	actual: `(1,4)`



ALGORITMO DE DIJKSTRA

nodo	1	2	3	4	5
dist min	INF	INF	3	1	0

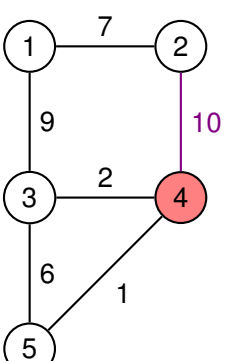
cola de prioridad (dist, nodo)	{(3, 3), (6, 3)}	actual: (1,4)



ALGORITMO DE DIJKSTRA

nodo	1	2	3	4	5
dist min	`INF`	`1 + 10 = 11`	3	1	0

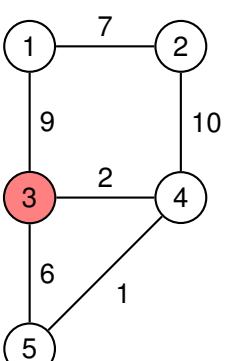
cola de prioridad (dist, nodo)	`{(3, 3), (6, 3), (1 + 10 = 11, 2)}`	actual: `(1,4)`



ALGORITMO DE DIJKSTRA

nodo	1	2	3	4	5
dist min	INF	11	3	1	0

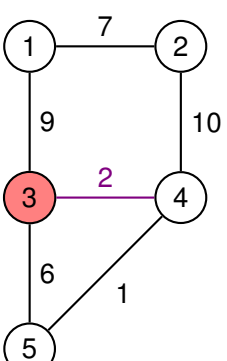
cola de prioridad (dist, nodo)	{(6, 3), (11, 2)}	actual: (3, 3)



ALGORITMO DE DIJKSTRA

nodo	1	2	3	4	5
dist min	INF	11	3	1	0

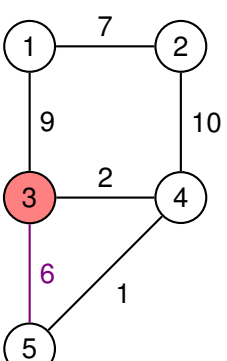
cola de prioridad (dist, nodo)	{(6, 3), (11, 2)}	actual: (3, 3)



ALGORITMO DE DIJKSTRA

nodo	1	2	3	4	5
dist min	INF	11	3	1	0

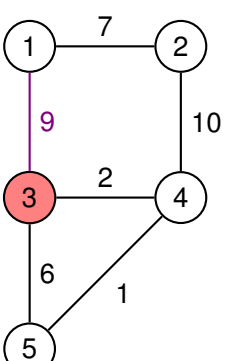
cola de prioridad (dist, nodo)	{(6, 3), (11, 2)}	actual: (3, 3)



ALGORITMO DE DIJKSTRA

nodo	1	2	3	4	5
dist min	INF	11	3	1	0

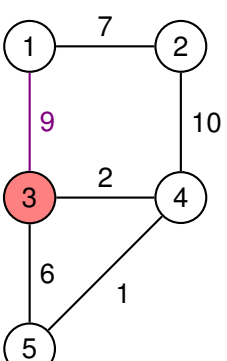
cola de prioridad (dist, nodo)	{(6, 3), (11, 2)}	actual: (3, 3)



ALGORITMO DE DIJKSTRA

nodo	1	2	3	4	5
dist min	`3 + 9 = 12`	`11`	`3`	`1`	`0`

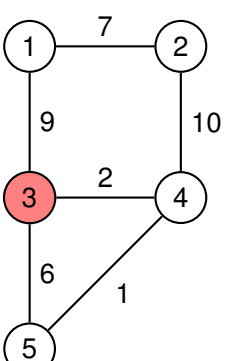
cola de prioridad (dist, nodo)	`{(6, 3), (11, 2), (3 + 9 = 12, 1)}`	actual: `(3, 3)`



ALGORITMO DE DIJKSTRA

nodo	1	2	3	4	5
dist min	12	11	3	1	0

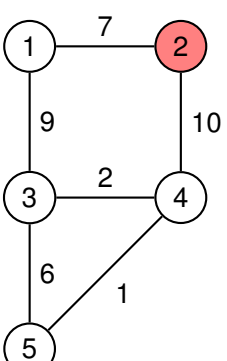
cola de prioridad (dist, nodo)	`{(11, 2), (12, 1)}`	actual: `(6, 3)`



ALGORITMO DE DIJKSTRA

nodo	1	2	3	4	5
dist min	12	11	3	1	0

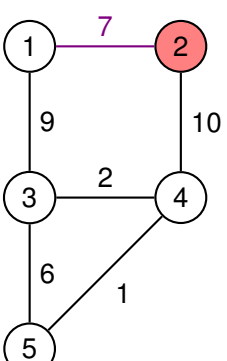
cola de prioridad (dist, nodo)	{(12, 1)}	actual: (11, 2)



ALGORITMO DE DIJKSTRA

nodo	1	2	3	4	5
dist min	12	11	3	1	0

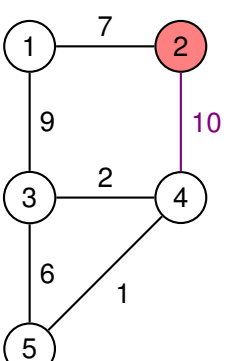
cola de prioridad (dist, nodo)	{(12, 1)}	actual: (11, 2)



ALGORITMO DE DIJKSTRA

nodo	1	2	3	4	5
dist min	12	11	3	1	0

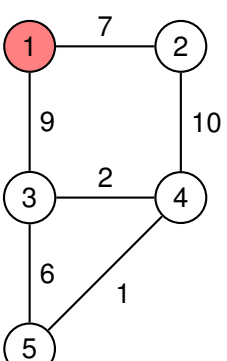
cola de prioridad (dist, nodo)	{(12, 1)}	actual: (11, 2)



ALGORITMO DE DIJKSTRA

nodo	1	2	3	4	5
dist min	12	11	3	1	0

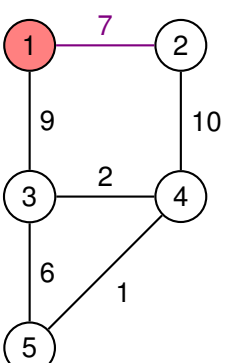
cola de prioridad (dist, nodo)	{}	actual: (12, 1)



ALGORITMO DE DIJKSTRA

nodo	1	2	3	4	5
dist min	12	11	3	1	0

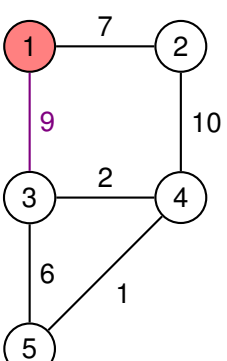
cola de prioridad (dist, nodo)	{}	actual: (12, 1)



ALGORITMO DE DIJKSTRA

nodo	1	2	3	4	5
dist min	12	11	3	1	0

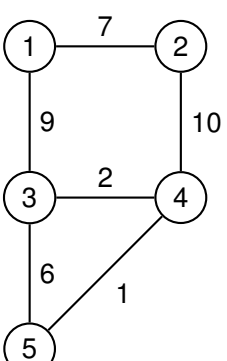
cola de prioridad (dist, nodo)	{}	actual: (12, 1)



ALGORITMO DE DIJKSTRA

nodo	1	2	3	4	5
dist min	12	11	3	1	0

cola de prioridad (dist, nodo)	{}	actual:



Código de Dijkstra

```cpp
int n;
// lista[u][i].first = vecino v del nodo u
// lista[u][i].second = peso de ir desde u hasta el su vecino
vector< vector< pair< int, int > > > lista;
vector< int > d;

void dijsktra(int s) {
  d[s] = 0;
  priority_queue< pair<int,int>, vector< pair<int,int> >, greater< pair<int,int> > > pq;
  pq.push({d[s], s});

  while(!pq.empty()) {
    int u = pq.top().second; int d_u = pq.top().first;
    pq.pop();

    // se encontro previamente otra distancia mas corta
    if(d_u > d[u]) continue;

    for(int i = 0; i < lista[u].size(); i++) {
      int v = lista[u][i].first;
      int w = lista[u][i].second;
      if(d_u + w < d[v]) {
        d[v] = d_u + w;
        pq.push({d[v], v});
      }
    }
  }
}
```

Posible Main

```cpp
int inf = 1000000000;

int main() {
  int e;
  cin >> n;
  cin >> e;
  lista.resize(n);
  d.resize(n, inf);

  for(int i = 0; i < e; i++) {
    int a, b, w;
    cin >> a >> b >> w;

    lista[a].push_back({b, w});
    lista[b].push_back({a, w});
  }

  int s; // nodo inicial
  s = 0; // como ejemplo
  dijsktra(s);

  // aqui hacen lo que quieran con el resultado
}
```

Código de Dijkstra, obtener camino

```cpp
int n;
vector< vector< pair< int, int > > > lista;
vector< int > d;
vector< int > p; // agregamos un vector

void dijkstra(int s) {
  d[s] = 0;
  priority_queue< pair<int,int>, vector< pair<int,int> >, greater< pair<int,int> > > pq;
  pq.push({d[s], s});

  while(!pq.empty()) {
    int u = pq.top().second; int d_u = pq.top().first;
    pq.pop();

    // se encontro previamente otra distancia mas corta
    if(d_u > d[u]) continue;

    for(int i = 0; i < lista[u].size(); i++) {
      int v = lista[u][i].first;
      int w = lista[u][i].second;
      if(d_u + w < d[v]) {
        d[v] = d_u + w;
        pq.push({d[v], v});
        p[v] = u; // decimos que el para llegar a v tienes que usar u
      }
    }
  }
}
```

Posible Main, obtener camino

```cpp
int inf = 1000000000;

int main() {
  int e;
  cin >> n; cin >> e;
  lista.resize(n);
  d.resize(n, inf); p.resize(n, -1);

  for(int i = 0; i < e; i++) {
    int a, b, w;
    cin >> a >> b >> w;

    lista[a].push_back({b, w});
    lista[b].push_back({a, w});
  }

  int s; // nodo inicial
  s = 0; // como ejemplo
  dijsktra(s);

  int f; // definir nodo final
  vector< int > path;
  for(int nodo = f; p[nodo] != -1; nodo = p[nodo]) {
    path.push_back(nodo);
  }
  path.push_back(s); // ojo el camino esta al revez
  // pueden hacer esto si quieren dejarlo desde s hasta f
  // reverse(path.begin(), path.end());
}
```

Detalles

- Complejidad algoritmica: $O((n + a) \log n)$

Algoritmo de Dijkstra

Detalles

- Complejidad algoritmica: $O((n + a) \log n)$
- Grafo dirigido con peso, es lo mismo!

Algoritmo de Dijkstra

Detalles

- Complejidad algoritmica: $O((n + a) \log n)$
- Grafo dirigido con peso, es lo mismo!

OJO

- Solo sirve para la distancia mínima de **UN** nodo a todo el resto
- Si el grafo tiene un arista negativa muere el algoritmo.

REFERENCES I