



UNIVERSIDAD NACIONAL DE CAJAMARCA  
FACULTAD DE INGENIERÍA  
PROGRAMA DE INGENIERÍA DE SISTEMAS



# **ALGORITMOS Y ESTRUCTURA DE DATOS II**

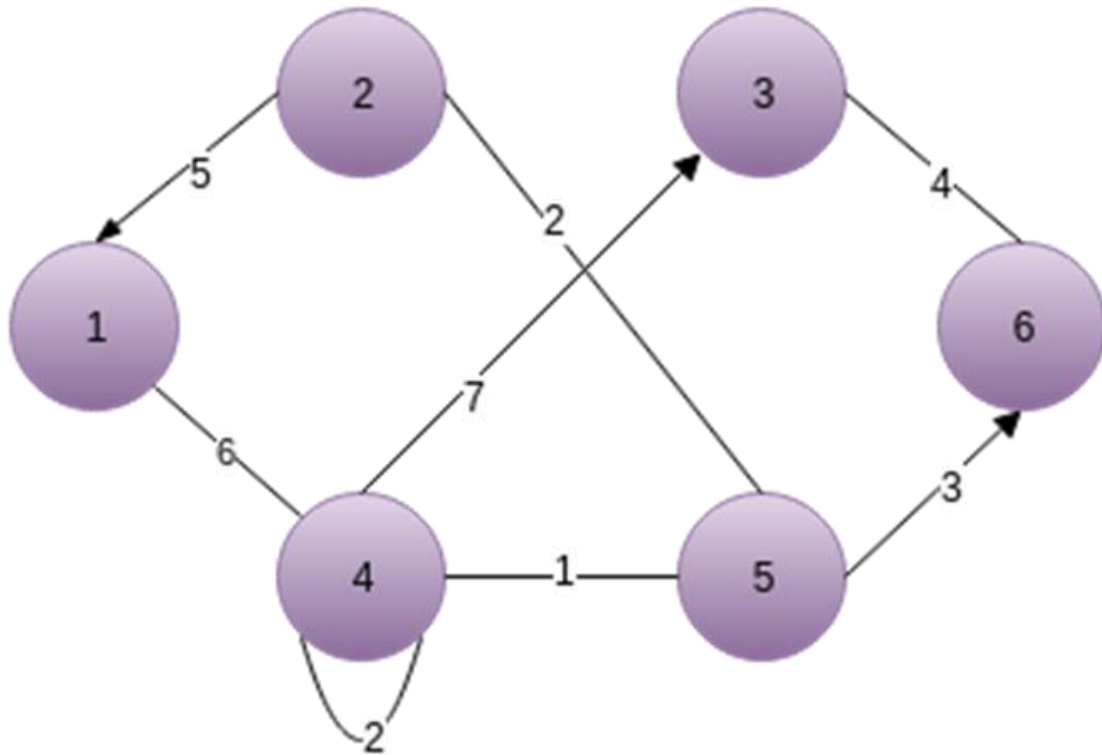
## **15: TALLER DE EJERCICIOS II**

# INTRODUCCIÓN

Semana 15 - Taller de ejercicios II



# ¿QUÉ PUEDE DECIR DEL GRÁFICO MOSTRADO?



	1	2	3	4	5	6
1	0	0	0	6	0	0
2	5	0	0	0	2	0
3	0	0	0	0	0	4
4	6	0	7	2	1	0
5	0	2	0	1	0	3
6	0	0	4	0	0	0

# RECORDEMOS

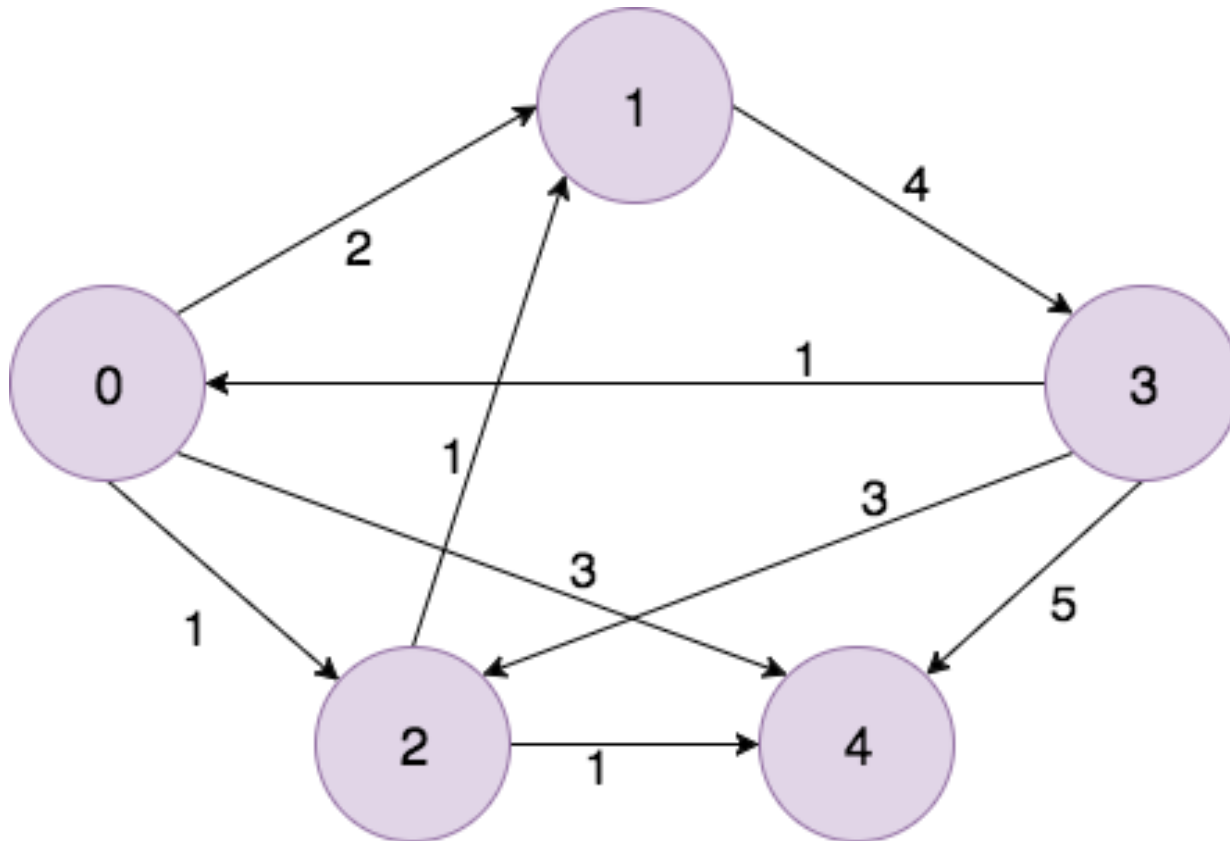


- ¿Qué es un grafo?
- ¿Qué es un vértice?
- ¿Qué entiende por el grado de un vértice?
- ¿Qué representa la siguiente expresión:  $G = (V, A)$ ?
- ¿Qué tipos de grafos recuerda?
- ¿Qué es una matriz de adyacencia?
- De qué maneras se puede buscar en un grafo?
- Ejemplos de grafos

# ¿CÓMO LO RESUELVO?



Encontrar el camino más corto de un vértice elegido a cualquier otro vértice del grafo



	0	1	2	3	4
0	0	2	1	-	3
1	-	0	-	4	-
2	-	1	0	-	1
3	1	-	3	0	5
4	-	-	-	-	0



## LOGRO ESPERADO

- **Al término de la sesión, el estudiante resuelve dos problemas, sobre caminos mínimos en grafos, lo resuelve usando Dijkstra y Floyd Warshal**



## DESARROLLO DEL TEMA

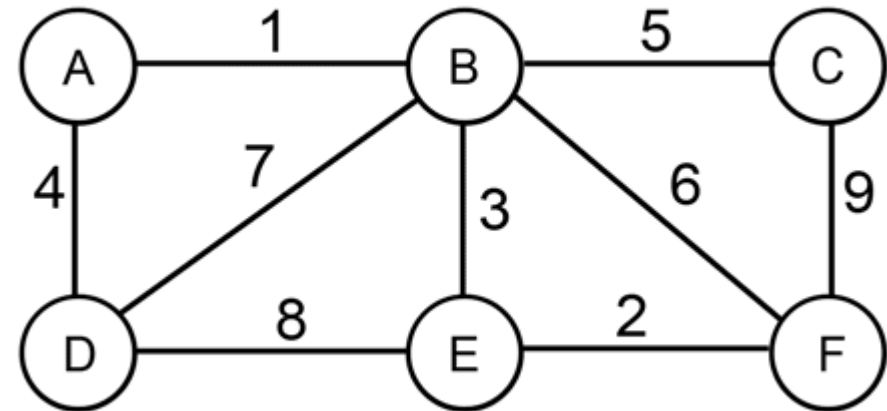
### Estructura de Datos



# ALGORITMOS DE GRAFOS DIRIGIDOS



- Algoritmos de determinación de los caminos más cortos:
  - Algoritmo de Dijkstra.
  - Algoritmo de Floyd-Warshall.





# ALGORITMO DE DIJKSTRA

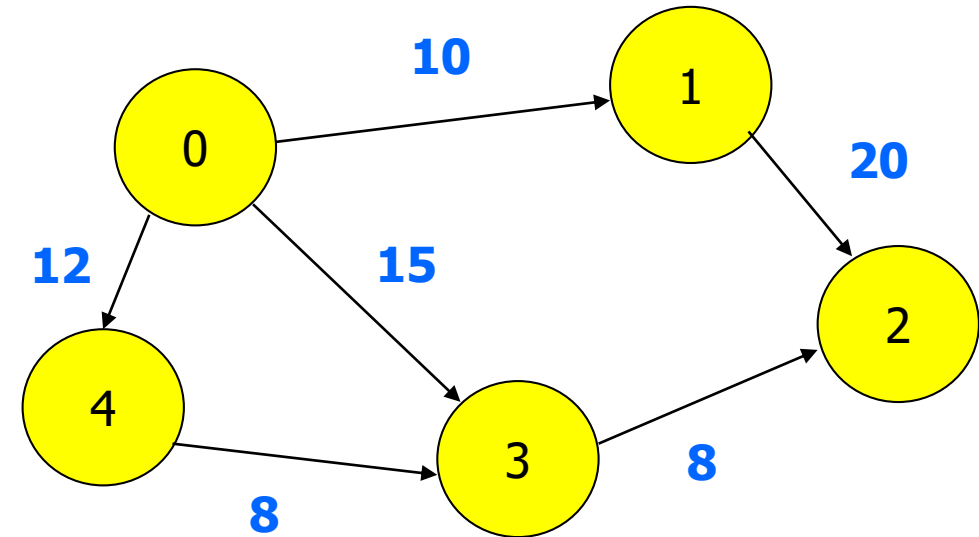


- El algoritmo de Dijkstra, también llamado algoritmo de caminos mínimos, es un algoritmo para la determinación del camino más corto dado un vértice origen al resto de vértices en un grafo dirigido y con pesos en cada arco.
- Su nombre se refiere a Edsger Dijkstra, quien lo describió por primera vez en 1959.





# ALGORITMO DE DIJKSTRA

- La idea subyacente en este algoritmo consiste en ir explorando todos los caminos más cortos que parten del vértice origen y que llevan a todos los demás vértices.
- Cuando se obtiene el camino más corto desde el vértice origen, al resto de vértices que componen el grafo, el algoritmo se detiene.




# ALGORITMO DE DIJKSTRA





## ALGORITMO DE DIJKSTRA

Dado un grafo  $G = (V, E)$ , el algoritmo de Dijkstra halla el camino más corto desde una fuente  $s \in V$  a todos los nodos  $v \in V$  cuando los pesos de las aristas son no negativos. Este es el algoritmo más rápido conocido para resolver el problema de SSSP para grafos arbitrarios con pesos no negativos.



# ALGORITMO DE DIJKSTRA



```
int const LIM = 20;
int const MAX = 1000000;

struct EstadoVertice{
    int ultimo;
    int distancia;
};

void leerMAd(int m[][LIM], int n);
void mostrarMAd(int m[][LIM], int n);
void inicializaM(int m[][LIM], int n, int v);
void dijkstra(int m[][LIM], int n, int origen, EstadoVertice dist[]);
int minimo(int f[], EstadoVertice dist[], int n);
```

```
void leerMAd(int m[][LIM], int n){
    inicializaM(m,n,MAX);
    cout<<"numero de aristas: ";
    int na;
    cin>>na;
    for (int i=0; i<na; i++){
        int verticeOrigen;
        int verticeDestino;
        do {
            cout<<"ARISTA "<<(i+1)<<": ";
            cout <<"vertice origen: ";
            cin>>verticeOrigen;
            cout <<"vertice destino: ";
            cin>>verticeDestino;
            cout<<"Costo: ";
            cin>>m[verticeOrigen][verticeDestino];
        } while (verticeDestino<0 || verticeOrigen<0 || verticeOrigen>=n || verticeDestino>=n);
    }
}
```

```
int minimo(int f[], EstadoVertice dist[], int n){
    int v;
    int may = MAX ;
    for (int i = 1; i<n; i++){
        if(!f[i] && (may >= dist[i].distancia)){
            may = dist[i].distancia;
            v = i;
        }
    }
    return v;
}
```

```
void dijkstra(int m[][LIM], int n, int origen, EstadoVertice dist[]){
    int f[LIM];
    f[origen]=1;
    for (int i = 1; i< n; i++){
        f[i] = 0;
        dist[i].distancia = m[0][i];
        dist[i].ultimo = 0;
    }
    int v ;
    for (int i = 1; i< n; i++){
        v = minimo(f,dist,n);
        f[v] = 1;

        for(int j = 1; j< n; j++){
            if(!f[j]){
                if((dist[v].distancia + m[v][j]) < dist[j].distancia){
                    dist[j].distancia = dist[v].distancia + m[v][j];
                    dist[j].ultimo = v;
                }
            }
        }
    }
}
```

}

# ALGORITMO DE DIJKSTRA



Matriz de Adyacencia

Nro de vertices: 5

numero de aristas: 6

ARISTA 1: vertice origen: 0

vertice destino: 1

Costo: 10

ARISTA 2: vertice origen: 0

vertice destino: 3

Costo: 15

ARISTA 3: vertice origen: 0

vertice destino: 4

Costo: 12

ARISTA 4: vertice origen: 4

vertice destino: 3

Costo: 8

ARISTA 5: vertice origen: 3

vertice destino: 2

Costo: 8

ARISTA 6: vertice origen: 1

vertice destino: 2

Costo: 20

Matriz de Adyacencia

1000000	10	1000000	15	12
1000000	1000000	20	1000000	1000000
1000000	1000000	1000000	1000000	1000000
1000000	1000000	8	1000000	1000000
1000000	1000000	1000000	8	1000000

el camino mas corto desde 0 a 1 es 10

el camino mas corto desde 0 a 2 es 23

el camino mas corto desde 0 a 3 es 15

el camino mas corto desde 0 a 4 es 12

**EVALUACIÓN DEL TEMA DESARROLLADO**

Reflexionemos!





# RECORDEMOS



- ¿Qué es un grafo ponderado?
- ¿Para qué nos sirve una matriz de adyacencia?
- ¿Indique algunos ejemplos en donde podemos aplicar lo aprendido en el tema de grafos?
- ¿Cuáles son los algoritmos de determinación de los caminos más cortos?
- En que consiste el algoritmo de Dijkstra?



## EJERCICIOS DE APLICACIÓN



# APLICANDO LO APRENDIDO



- Dado un grafo ponderado y utilizando el programa creado con el algoritmo de Dijkstra, encuentre las distancias más cortas desde un nodo cualquiera a los demás nodos del grafo.
- ¿Cómo podría encontrar la distancia más corta entre cada par de nodos del grafo? Investigar el algoritmo de Floyd Warshall