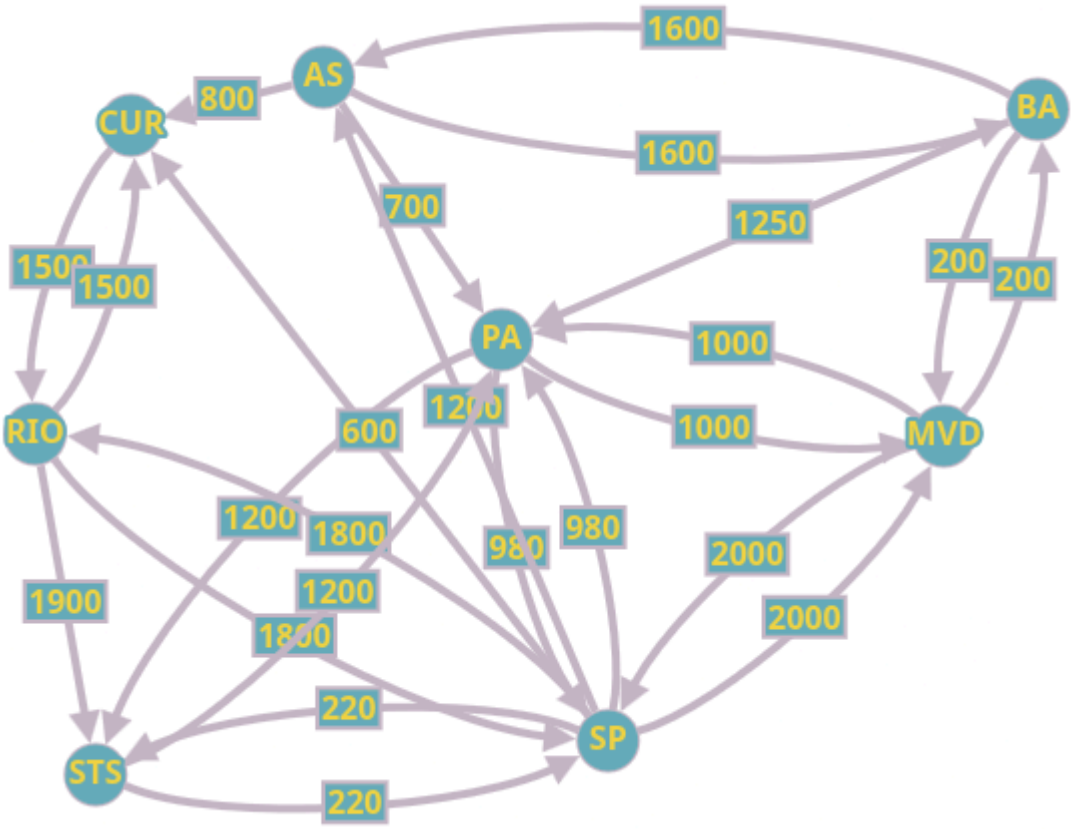
Algoritmos y Estructuras de Datos
UT7-PD3

Santiago Blanco

11-06-2025



Ejercicio #1

1. El costo de volar de Montevideo a Rio de Janeiro es: b. 3780.
2. El costo de volar de Montevideo a Curitiba es: a. 2580
3. Los servicios de mantenimiento se instalan en: d. Porto Alegre, dado que es el centro del grafo.

Ejercicio #2

Salida del programa:

```
1         System.out.println("Existe camino entre Montevideo y Curitiba: " +
gd.existeCamino("Montevideo", "Curitiba"));
2         System.out.println("Existe camino entre Porto Alegre y Santos: " +
gd.existeCamino("PortoAlegre", "Santos"));
3
4         ----------------------
5
6 Existe camino entre Montevideo y Curitiba: true
7 Existe camino entre Porto Alegre y Santos: true
```

Ejercicio #3

```
bpf(origen: Comparable) -> Collection<TVertice>
COM
  visitados <- conjunto vacío de Comparable // O(1)
  bpfRecursivo(origen, visitados) // O(n^2)
  RETORNAR lista de vértices correspondiente a las etiquetas en visitados
FIN

bpfRecursivo(actual: Comparable, visitados: Conjunto<Comparable>)
COM
  visitados.add(actual) // O(1)
  verticeActual <- vertices.obtener(actual) // O(1)

  SI verticeActual <> nulo ENTONCES
    PARA CADA adyacencia EN verticeActual.getAdyacentes() HACER // O(v)
      destino <- adyacencia.getDestino().getEtiqueta() // O(1)
      SI destino NO está en visitados ENTONCES
        bpfRecursivo(destino, visitados)
      FIN SI
    FIN PARA
  FIN SI
FIN
```

Orden de tiempo de ejecución

Tiene un orden $O(n)$ porque recorre cada vértice una sóla vez.

Recorrida desde Montevideo

Output:

```
Desde Montevideo (BPF): [Curitiba, Santos, BuenosAires, Asuncion, SanPablo,
Montevideo, RioDeJaneiro, PortoAlegre]
```

Obtener Caminos

Lenguaje natural

Este algoritmo, toma como parámetro el vértice de inicio y el destino. Recorre los vértices del grafo de manera recursiva, pasando por cada vértice adyacente de un vértice (w). En caso de que de que el vértice adyacente no sea el destino, se suma el camino hasta ese vértice adyacente al camino actual, en caso de que dicho vértice sí sea el nodo destino, guarda el valor del camino hasta dicho vértice. Para que esto se lleve a cabo, cada nodo tiene un atributo bool que verifica si ya fue visitado o no para que no se visite el mismo nodo 2 veces. Al final el nodo por el que se empezó del camino completo.

Precondiciones

- El grafo dirigido debe ser válido
- TCaminos es una colección de TCamino

Postcondiciones

- obtenerCamino agrega todos los caminos TCamino posibles hasta el vertice destino a la colección TCaminos

Pseudo

```
obtenerCamino(destino: TVertice, elCamino: TCamino, losCaminos: TCaminos)
COM
  this.visitado <- VERDADERO // O(1)
  elCamino.agregar(this) // O(1)

  SI w == destino ENTONCES
      losCaminos.agregar(elCamino + destino) // O(1)
  FIN SI

  PARA CADA ady EN adyacentes HACER // O(n^2)
    w <- ady.destino // O(1)
    SI w.visitado() == FALSO ENTONCES
      obtenerCamino(destino, elCamino, losCaminos)
    FIN SI
  FIN PARA CADA

  elCamino.quitar(this) // O(1)
  this.visitado <- FALSO // O(1)
FIN
```

Resultado

```
[Montevideo, BuenosAires, Asuncion, Curitiba, RioDeJaneiro]
[Montevideo, BuenosAires, Asuncion, PortoAlegre, SanPablo, Curitiba,
RioDeJaneiro]
[Montevideo, BuenosAires, Asuncion, PortoAlegre, SanPablo, RioDeJaneiro]
[Montevideo, BuenosAires, Asuncion, PortoAlegre, Santos, SanPablo, Curitiba,
RioDeJaneiro]
[Montevideo, BuenosAires, Asuncion, PortoAlegre, Santos, SanPablo,
RioDeJaneiro]
[Montevideo, BuenosAires, PortoAlegre, SanPablo, Asuncion, Curitiba,
RioDeJaneiro]
[Montevideo, BuenosAires, PortoAlegre, SanPablo, Curitiba, RioDeJaneiro]
[Montevideo, BuenosAires, PortoAlegre, SanPablo, RioDeJaneiro]
[Montevideo, BuenosAires, PortoAlegre, Santos, SanPablo, Asuncion, Curitiba,
RioDeJaneiro]
[Montevideo, BuenosAires, PortoAlegre, Santos, SanPablo, Curitiba,
RioDeJaneiro]
[Montevideo, BuenosAires, PortoAlegre, Santos, SanPablo, RioDeJaneiro]
[Montevideo, PortoAlegre, SanPablo, Asuncion, Curitiba, RioDeJaneiro]
[Montevideo, PortoAlegre, SanPablo, Curitiba, RioDeJaneiro]
[Montevideo, PortoAlegre, SanPablo, RioDeJaneiro]
[Montevideo, PortoAlegre, Santos, SanPablo, Asuncion, Curitiba, RioDeJaneiro]
[Montevideo, PortoAlegre, Santos, SanPablo, Curitiba, RioDeJaneiro]
[Montevideo, PortoAlegre, Santos, SanPablo, RioDeJaneiro]
[Montevideo, SanPablo, Asuncion, Curitiba, RioDeJaneiro]
[Montevideo, SanPablo, Curitiba, RioDeJaneiro]
[Montevideo, SanPablo, RioDeJaneiro]
```