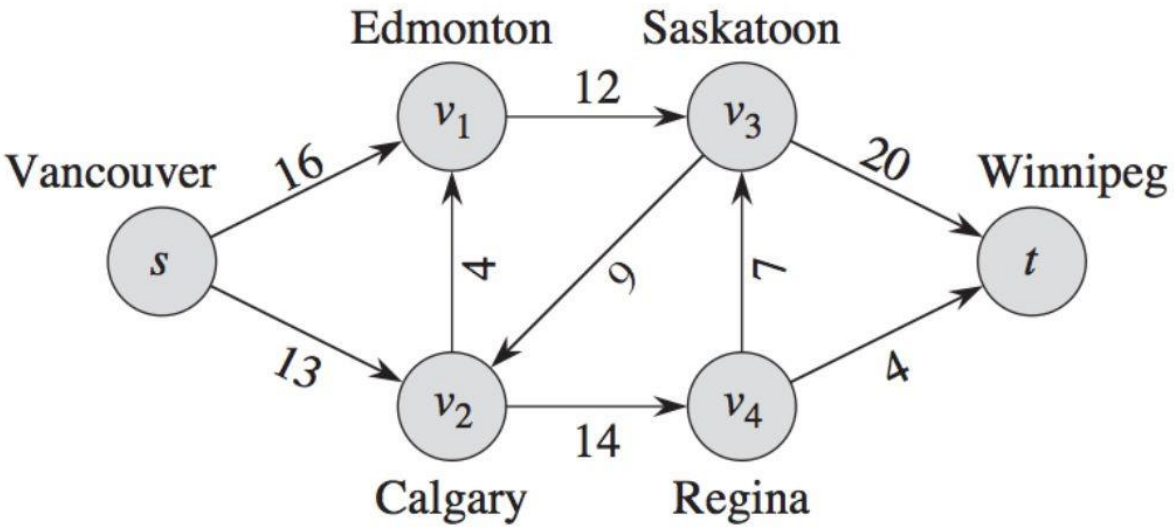
Entregable Backtracking y SDL

Presentado por: - Ramiro Pinchao

- Kenneth Rodriguez

- Brayan Suarez



Existe una conexión entre Saskatoon y Vancouver?

```
conexionEntre( saskatoon, vancouver )
false
```

Con qué nodos está conectado Regina y cual es el costo de cada conexión?

```
nodosConectados( regina, Ciudad, CostoCiudad )
Ciudad = saskatoon,
CostoCiudad = 7
Ciudad = winnipeg,
CostoCiudad = 4
```

Es posible viajar desde Edmonton a Calgary?

true
false

Construir una regla para determinar si un nodo tiene aristas

```
%Regla para saber si un nodo tiene aristas
tieneAristas(X) :- conexion(X, _, _).
```

Construir una regla para determinar cuál es el costo para ir de un nodo X a un Z pasando por Y

```
%Regla para conocer el costo de ir a Y lugar desde X lugar pasando por Z lugar
costoEntreConParada(X, Y, Z, Costo) :-
  costoEntre(X, Z, Costo1),
  costoEntre(Z, Y, Costo2),
  Costo is Costo1 + Costo2.


%Reglas para conocer el costo de viajar desde X ciudad a Y ciudad
costoEntre(X, Y, Costo) :-
  costoEntre(X, Y, [X], 0, Costo).

costoEntre(X, Y, _, Acumulado, Costo) :-
  conexion(X, Y, C),
```

```
    Costo is Acumulado + C.

costoEntre(X, Y, Visitados, Acumulado, Costo) :-
    conexion(X, Z, C),
    \+ member(Z, Visitados),
    NuevoCosto is Acumulado + C,
    costoEntre(Z, Y, [Z|Visitados], NuevoCosto, Costo).
```

Código Entero :

```
conexion(vancouver, edmonton, 16).
conexion(vancouver, calgary, 13).
conexion(edmonton, saskatoon, 12).
conexion(calgary, edmonton, 4).
conexion(calgary, regina, 14).
conexion(saskatoon, winnipeg, 20).
conexion(saskatoon, calgary, 9).
conexion(regina, saskatoon, 7).
conexion(regina, winnipeg, 4).

%Regla para saber si un nodo tiene aristas
tieneAristas(X) :- conexion(X, _, _).

%regla para conocer los nodos conectados a una ciudad junto con su costo
nodosConectados(X,Y, Z) :- conexion(X, Y, Z).
```

```
%reglas para saber si existe una conexion entre X ciudad con Y ciudad
conexionEntre( X, Y ) :- conexionEntreAux(X,Y, []).

conexionEntreAux(X, Y, _ ) :- conexion(X, Y, _).

conexionEntreAux(X, Z, Visitados) :-
   conexion(X, Y, _),
   \+ member(Y, Visitados),
   conexionEntreAux(Y, Z, [X|Visitados]).


%Regla para conocer el costo de ir a Y lugar desde X lugar pasando por Z lugar
costoEntreConParada(X, Y, Z, Costo) :-
   costoEntre(X, Z, Costo1),
   costoEntre(Z, Y, Costo2),
   Costo is Costo1 + Costo2.


%Reglas para conocer el costo de viajar desde X ciudad a Y ciudad
costoEntre(X, Y, Costo) :-
   costoEntre(X, Y, [X], 0, Costo).

costoEntre(X, Y, _, Acumulado, Costo) :-
   conexion(X, Y, C),
   Costo is Acumulado + C.
```

```
costoEntre(X, Y, Visitados, Acumulado, Costo) :-
    conexion(X, Z, C),
    \+ member(Z, Visitados),
    NuevoCosto is Acumulado + C,
    costoEntre(Z, Y, [Z|Visitados], NuevoCosto, Costo).
```