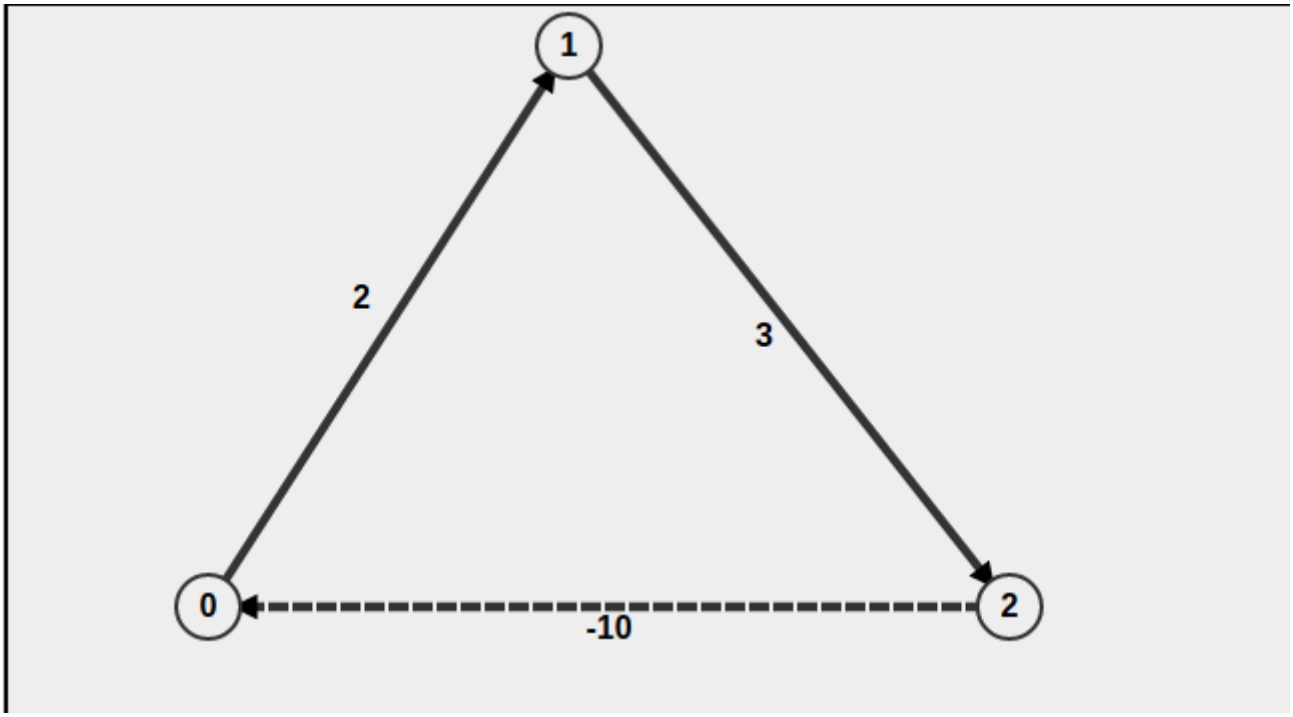
Detectando ciclos negativos en grafos.

Cuando estudiamos los caminos mínimos en grafos desde un solo destino, vimos que para aplicar el algoritmo de dijkstra no debemos tener ciclos negativos, veamos porque no queremos ciclos negativos,



Si corremos algún algoritmos para conseguir las distancias mínimas desde el vértice cero ¿cuales serían?

Notemos que inicialmente tenemos $d[0] = 0$ por lo que podríamos pensar que $d[1] = 2$. Definamos $\delta_i(j)$ como el camino mínimo a `j` desde `i`. Queremos que eventualemte avancemos en el algoritmos $d[j] \to \delta_0(j)$ pero notemos que si tomamos el camino $0 \to 1 \to 2 \to 0 \to 1$ obtenemos `d[1] = -3` y si seguimos caminando en el ciclo notaremos que $\delta_0[i] \to -\infty$ por lo que realmente no tenemos una distancia mínima, y el camino mínimo no estaría definido. Por lo que solo nos queda pdoer detectar si es que existe uno.

Detectar ciclos negativos por medio de Bellman-Ford

Recordemos que del algoritmo de Bellman-Ford iteraremos `V - 1` para terminar y después que eso termine todos los caminos mínimos deben estar encontrados.(si es que existen.) El algoritmo de BF es:

```
vector<int> dist(V, INF);
dist[s] = 0;
for(int i = 0; i < V - 1; i ++){
    for(int u=0; u<V; u++){
        for(int j=0; j<(int)AdjList[u].size(); j++){
            pair<int,int> v = AdjList[u][j];
            dist[v.first] = min(dist[v.first], dist[u] + v.second);
        }
    }
}
```

Se puede probar que después de esto si los caminos mínimos están definidos entonces deben ser hallados. ¿pero qué pasa si no?. Si existe algun Ciclo por más iteraciones que demos `dist[i]`seguirá decreciendo para `i` en el ciclo.

Por lo que para verificr si existe algun ciclo negativo basta hacer una iteración más y checar si hay algún `dist[i]` que se puede hacer más pequeño. Si esto es posible existe un ciclo negativo pues si no existiences `dist[i]` ya hubiera convergido $\delta_s(i)$ que es el costo del camino mínimo. Entonces para detectar un ciclo usando BF podemos correr BF en $\mathbb{O}(VE)$ y después buscar poder relajar un eje más de la forma siguiente.

```
bool HayCicloNegativo = false;
for(int u = 0; u < V; u++)
    for(int j = 0; j<(int)AdjList[u].size(); j++){
        pair<int> AdjList[u][j];
        if(dist[u] + v.first < dist[v.first]){
            HayCicloNegativo = true;
        }
    }
```

Detectar ciclos negativos por medio de SPFA.

En camino mínimo vieron el algoritmo Shortest Path Faster Algorithm, ahora usemoslo para detectar la presenciade ciclos negativos. SPFA es un algoritmo que optimiza BF aunque en el peor de los casos tiene la misma complejidad, aun así en promedio se comporta mejor.

Sabemos que un vértice que pertenece a un ciclo negativo siempre se podrá seguir encontrando un camino menor a él, por lo que si un eje entra `V - 1` a la cola siginifica hay un ciclo negativo, el código queda así.

```
void SPFA(int source){
        dist[source] = 0;
        queue<int> cola; cola.push(source);
        vi encola(V,0); encola[source] = 1;
        vi visit(V,0);
        while(!cola.empty()){
                int u = cola.front(); cola.pop();
                encola[u] = 0;
                visit[u]++;
                if(visit[u] < V){
                        for(int j=0 ; j<(int)AdjList[u].size(); j++){
                                ii v = AdjList[u][j];
                                if(dist[u] + v.second < dist[v.first]){
                                        dist[v.first] = dist[u] + v.second;
                                        if(!encola[v.first]){
                                                cola.push(v.first);
                                                encola[v.first] = 1;
                                        }
                                }
                        }
```

```
                }else {tieneCiclo = true;}
            }
        }
```