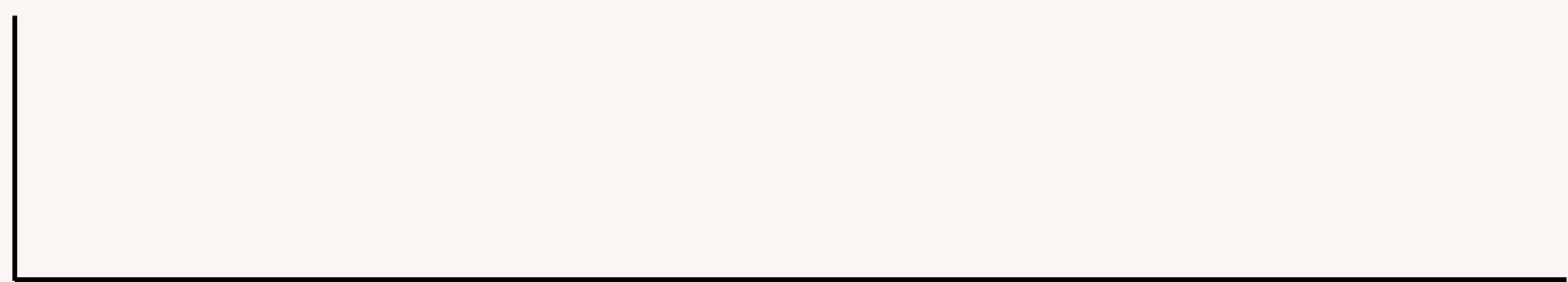




# BUSCA DE HAZARDS

Hazards search

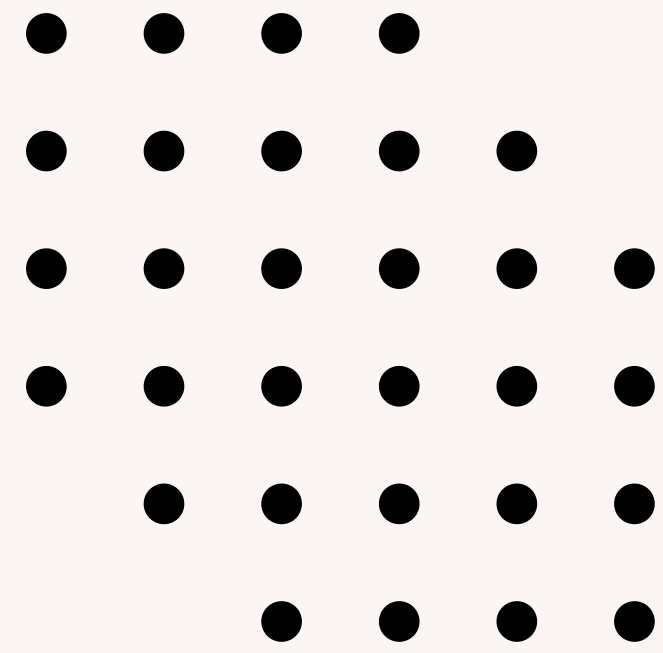


# POTENCIALIDADES

- Otimização de recursos
- Continuação das operações

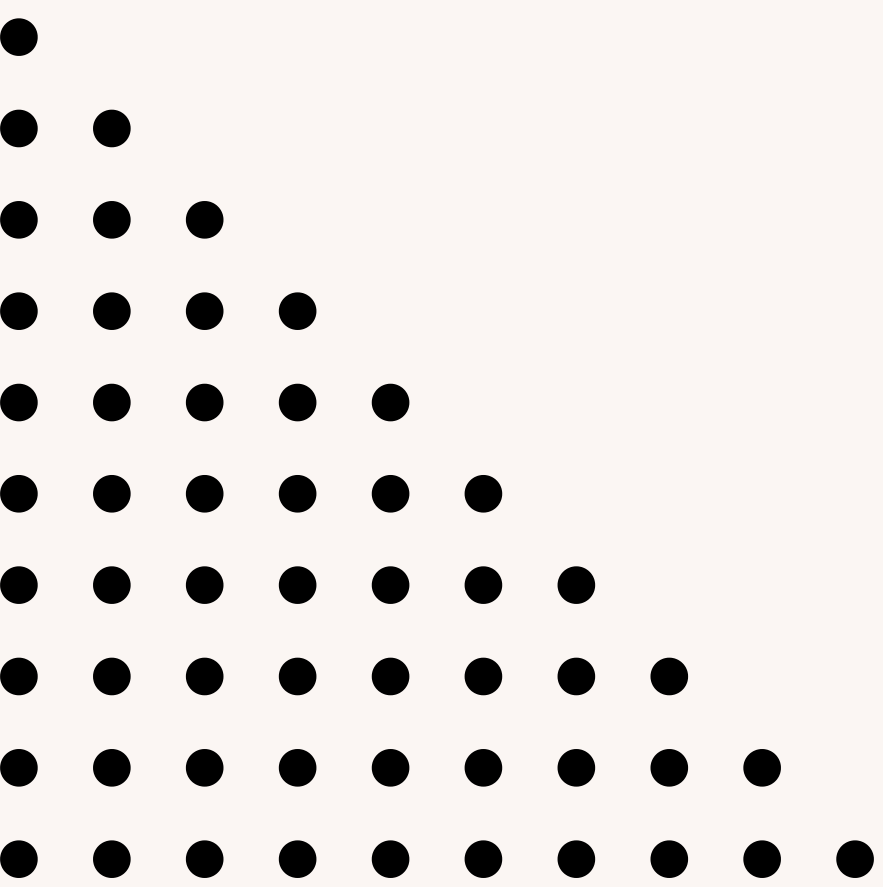
# FRAGILIDADES

- Incompletude
- Falsos positivos



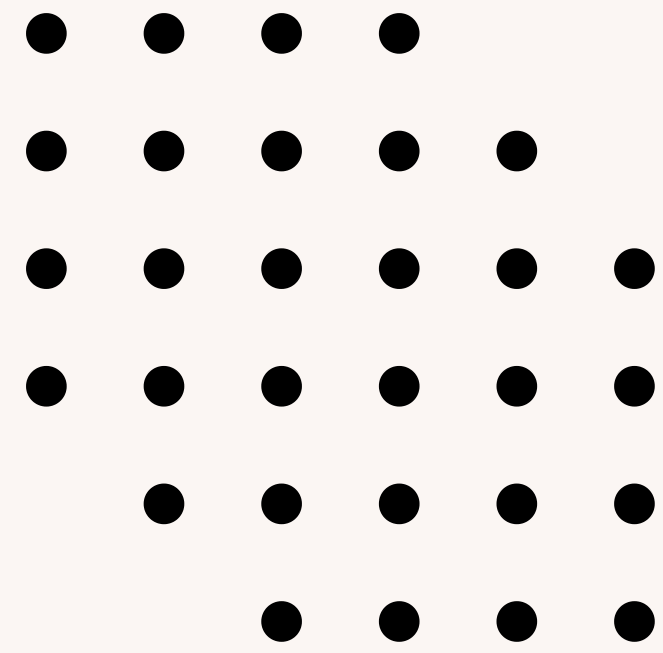


# COMO FUNCIONA?

- 
- 1 definição do escopo
  - 2 Identificação de hazards
  - 3 Análise de riscos
  - 4 avaliação de riscos
  - 5 Controle de riscos
  - 6 Monitoramento e revisão

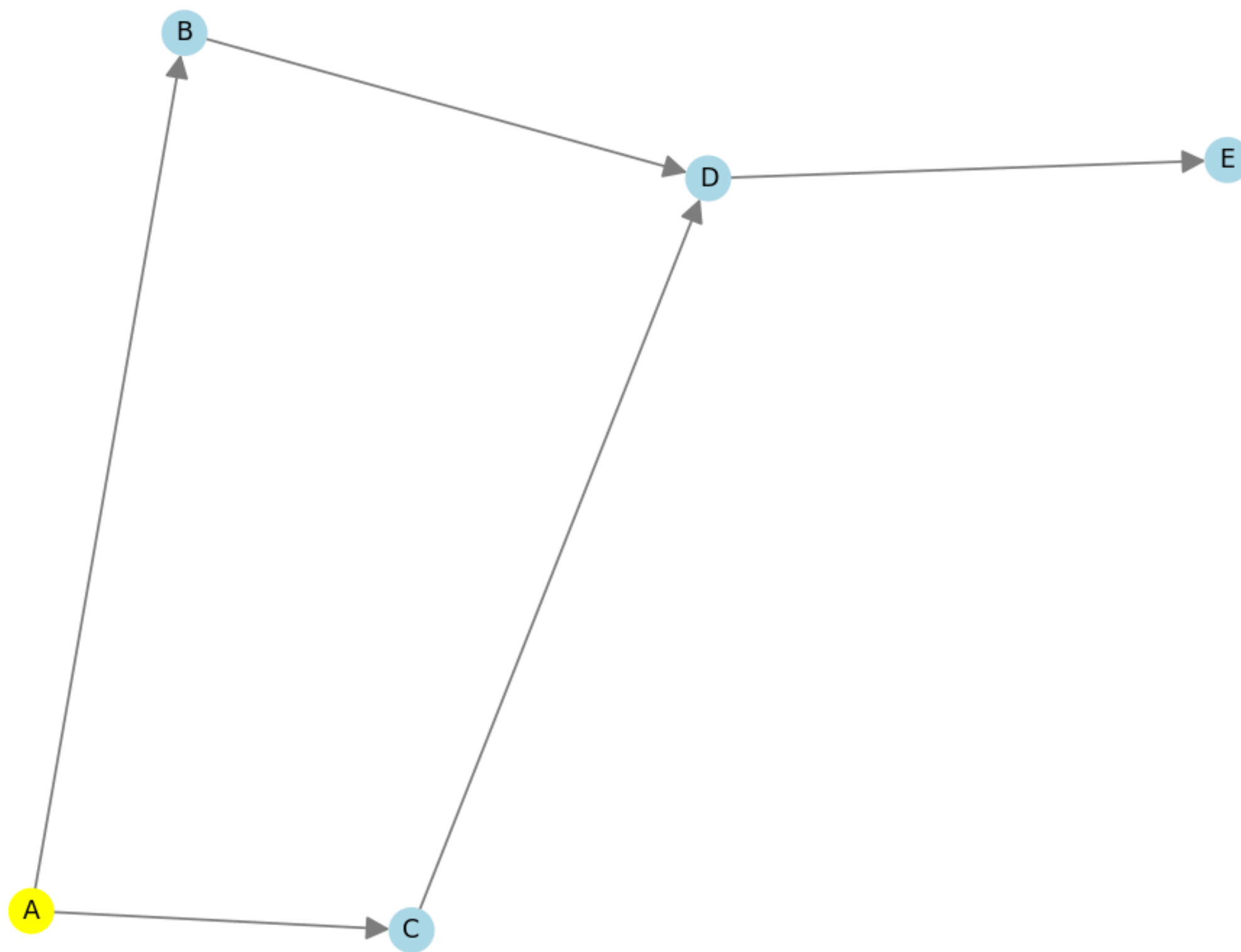
# ANÁLISE UTILIZANDO GRAFOS

- **Tipo de hazard:** hazard de dado
- **Tipo de grafo:** direcionado acíclico
  - **Nó:** Tarefas
  - **Aresta:** Dependência entre as tarefas

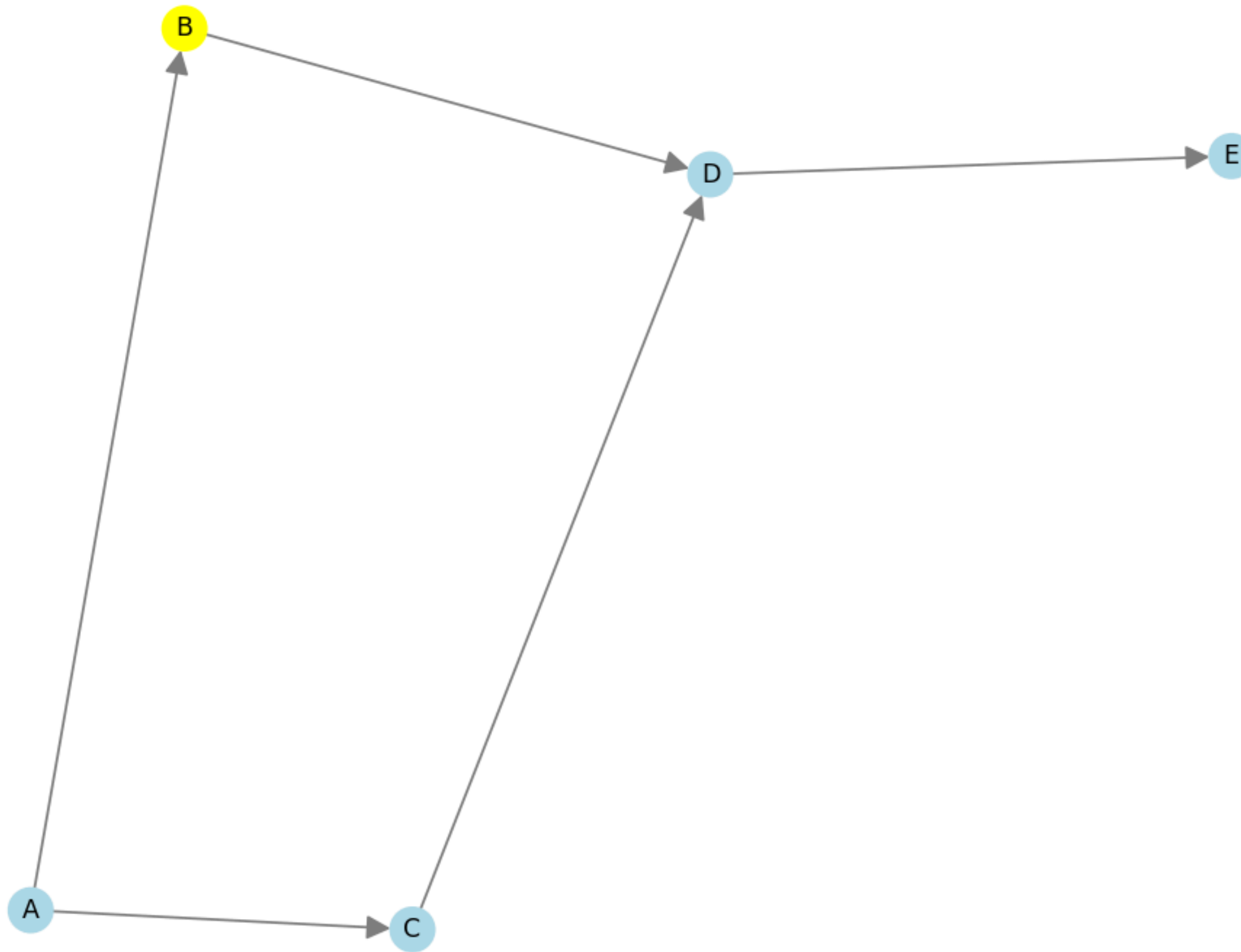


# ALGORITMO

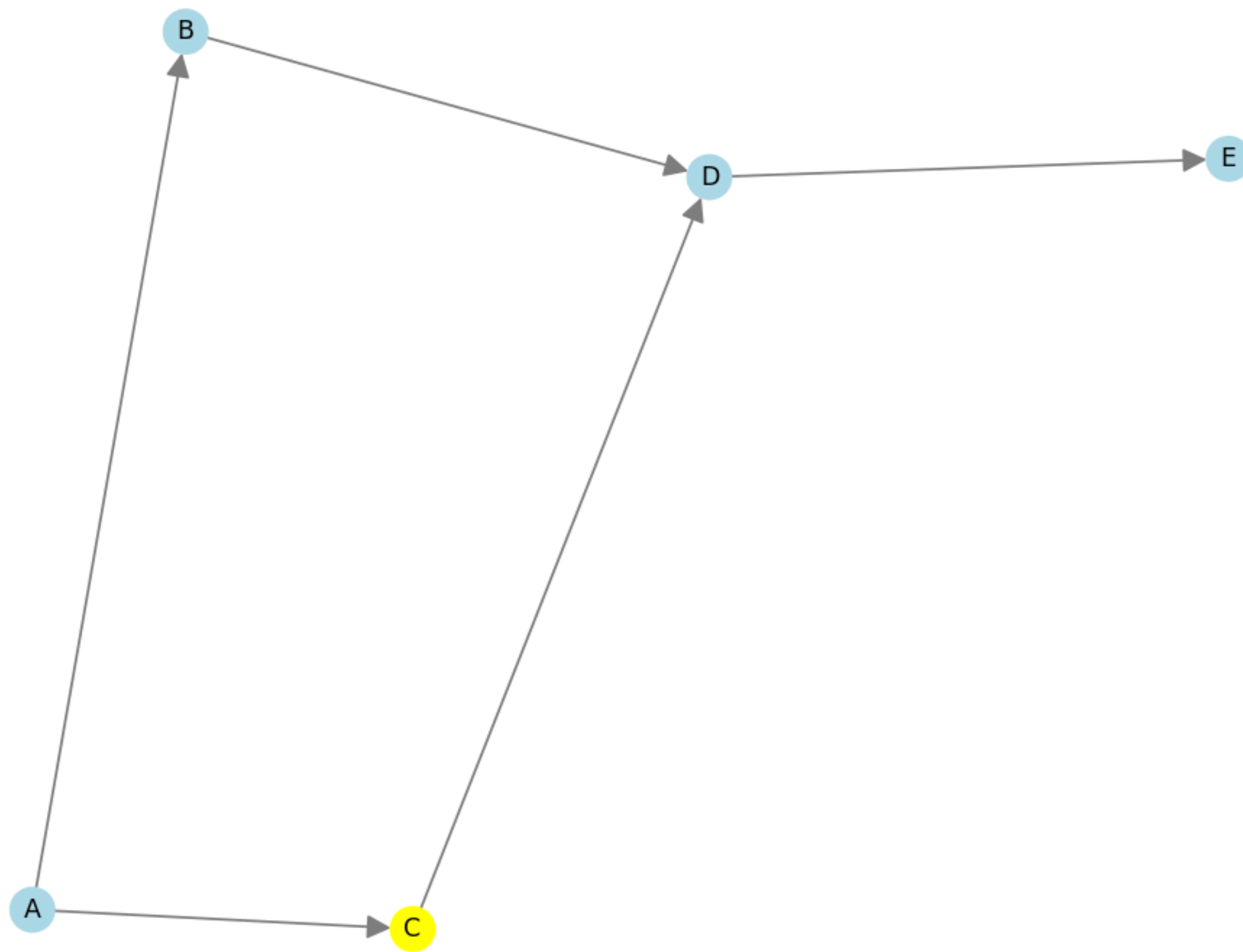
```
def buscar_hazards(grafo):  
    hazards = []  
    for node in grafo.nodes:  
        predecesores = list(grafo.predecessors(node))  
        if len(predecesores) > 1:  
            for i in range(len(predecesores)):  
                for j in range(i + 1, len(predecesores)):  
                    pre1, pre2 = predecesores[i], predecesores[j]  
                    if pre1 in grafo and pre2 in grafo:  
                        hazards.append((pre1, pre2, node))  
    return hazards
```



Nó 'A' tem predecessores: []  
Nenhum hazard encontrado  
para o nó 'A'.

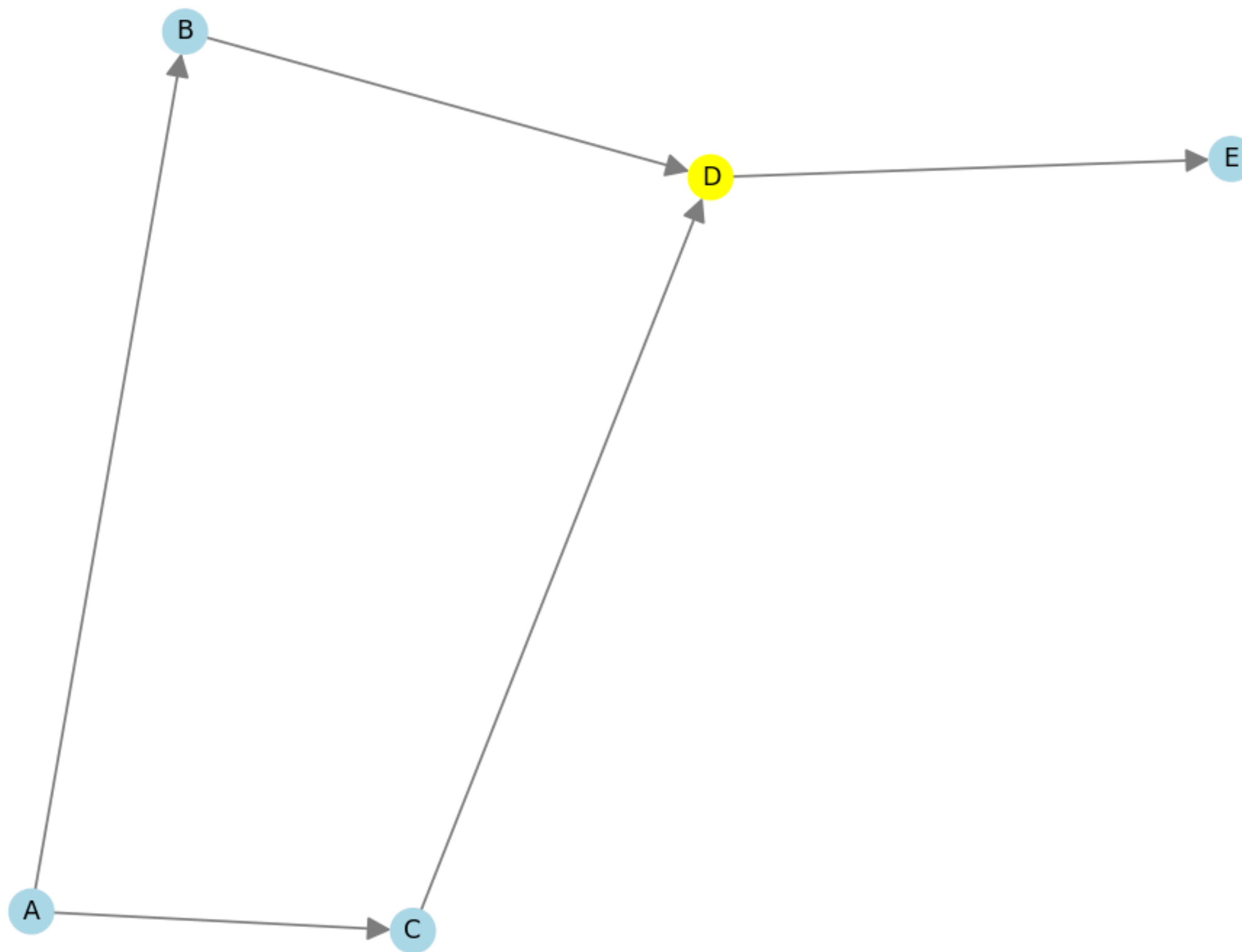


Nó 'B' tem predecessores: ['A']  
Nenhum hazard encontrado  
para o nó 'B'.

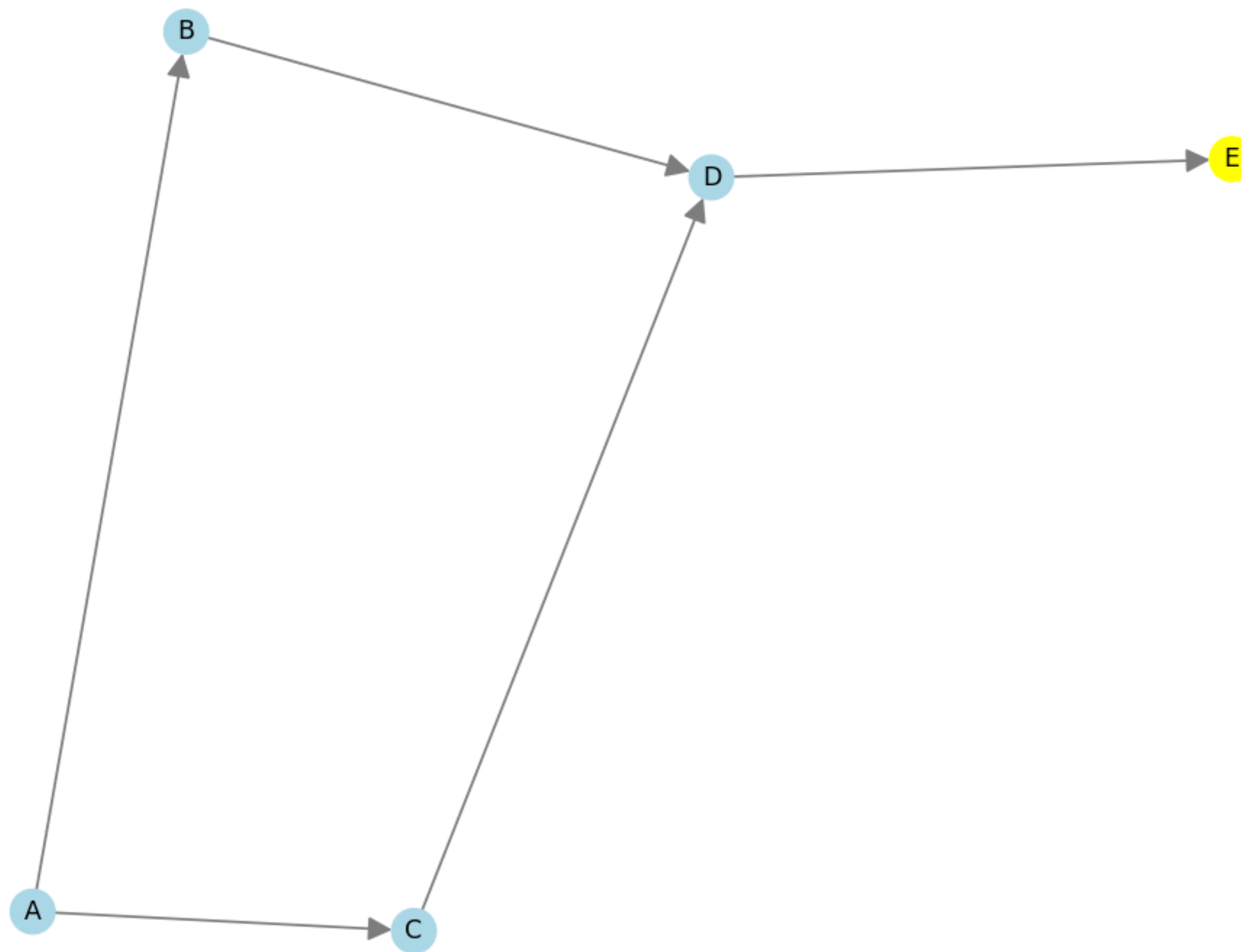


Nó 'C' tem predecessores: ['A']  
Nenhum hazard encontrado  
para o nó 'C'.

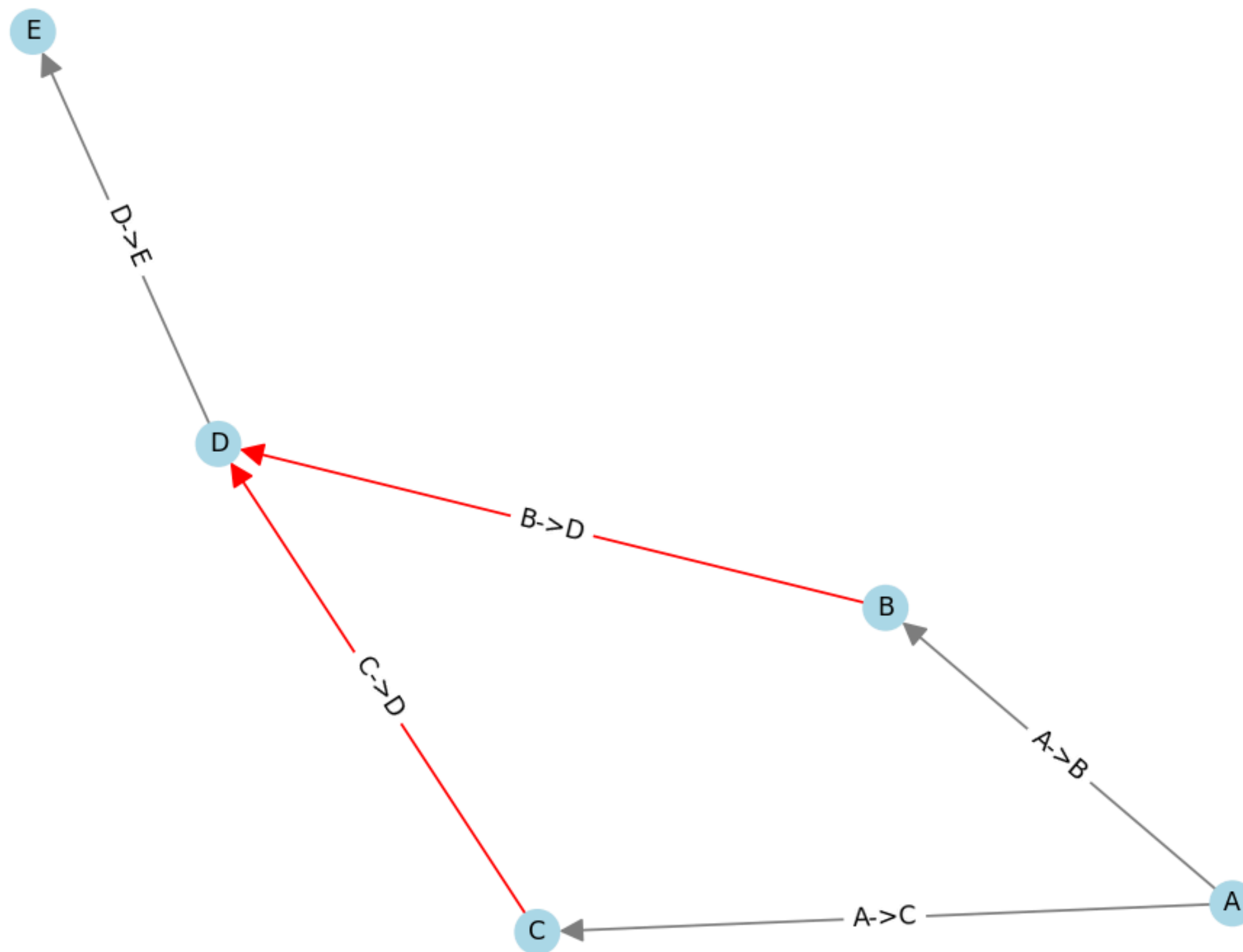




Nó 'D' tem predecessores: ['B', 'C']  
Possível hazard identificado: B, C -  
> D



Nó 'E' tem predecessores: ['D']  
Nenhum hazard encontrado para o  
nó 'E'.



Grafo:

('A', 'B')

('A', 'C')

('B', 'D')

('C', 'D')

('D', 'E')

Hazards encontrados:

Hazard entre B e C no nó D

# POTENCIALIDADES

- Facilidade de extensão
- Simplicidade e clareza

# FRAGILIDADES

- Falta de validação dos dados
- Redundância na verificação de hazards

