Definición de ciclos

_ciclo.png _ciclo.bb ciclo.png height7.60422ptciclo.png width55.97243ptciclo.pngGraphic file (type bmp)

Figura: Figura con ciclos

Definición de camino

Un camino en un grafo es una lista de nodos de forma que dos nodos consecutivos son adyacentes. Utilizaremos la siguiente notación:

$$v_0 \mapsto v_1 \mapsto \dots v_n.$$

Definición de ciclo

Un ciclo es un camino donde el vértice de inicio y el final son iguales.

£Cómo podemos saber si un grafo tiene ciclos?

Propiedad

Un grafo dirigido tiene ciclos si y solamente si realizando el algoritmo DFS aparecen aristas atrás.

Ciclos

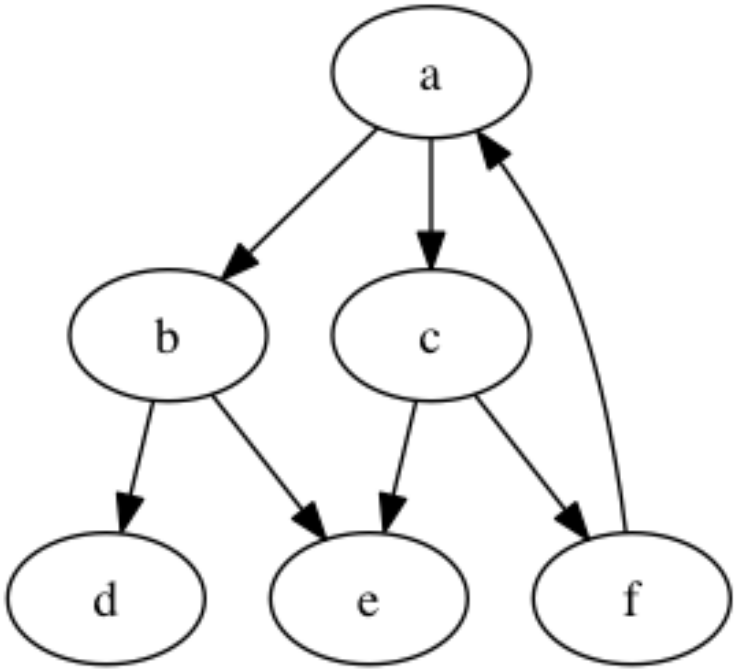


Figura: Figura con ciclos

DAG (Direct Acyclic Graphs)

Los grafos directos acíclicos son muy utilizados, modelan:

- relaciones de causalidad,
- cadenas de trabajos,
- parentescos,
- preguntas y respuestas.

Propiedades de los DAGs

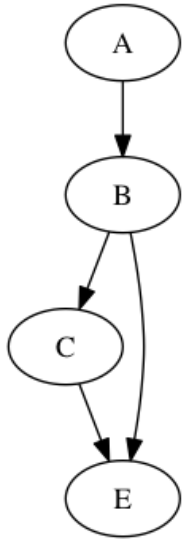


Figura: Ejemplo de DAG

- En los DAGs hay, al menos, una fuente y un sumidero.
- Un DAG admite una linearización.
- En un DAG, cada eje lleva a nodos con menor valor post.

£En que orden hay que ponerse el equipamiento?

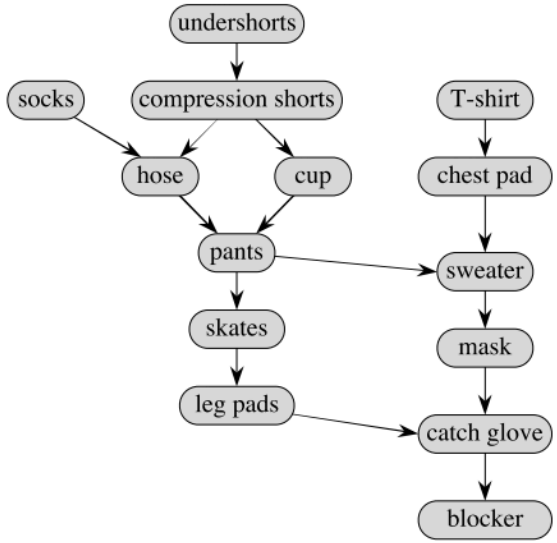


Figura: Tomada del libro **Unlocked algorithms** de T. Cormen (capítulo 5).

Grafos con pesos en las aristas

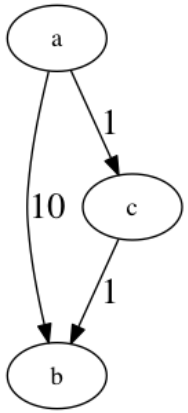
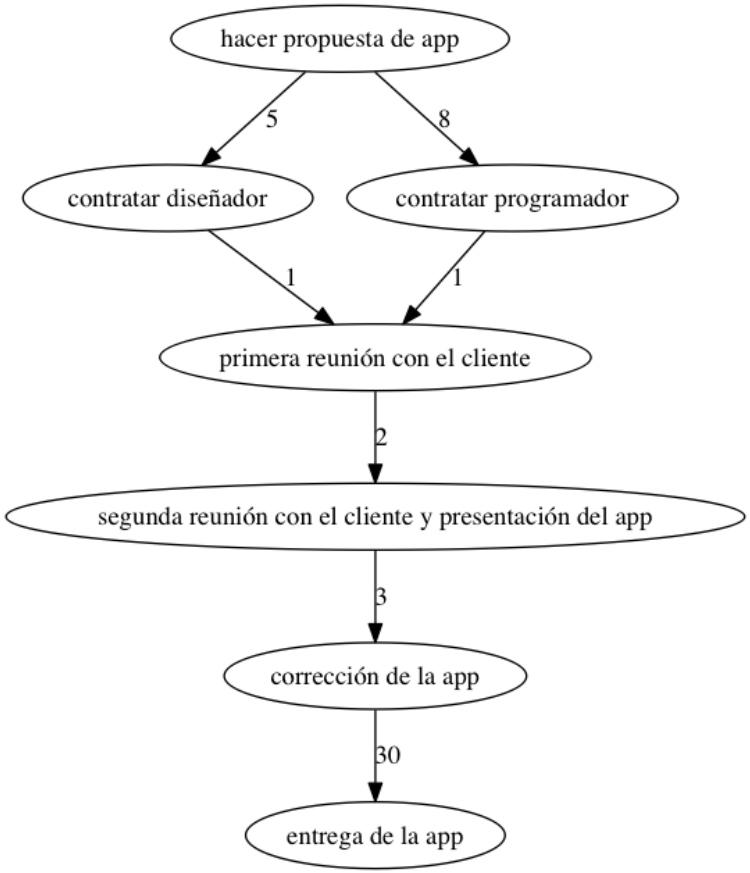
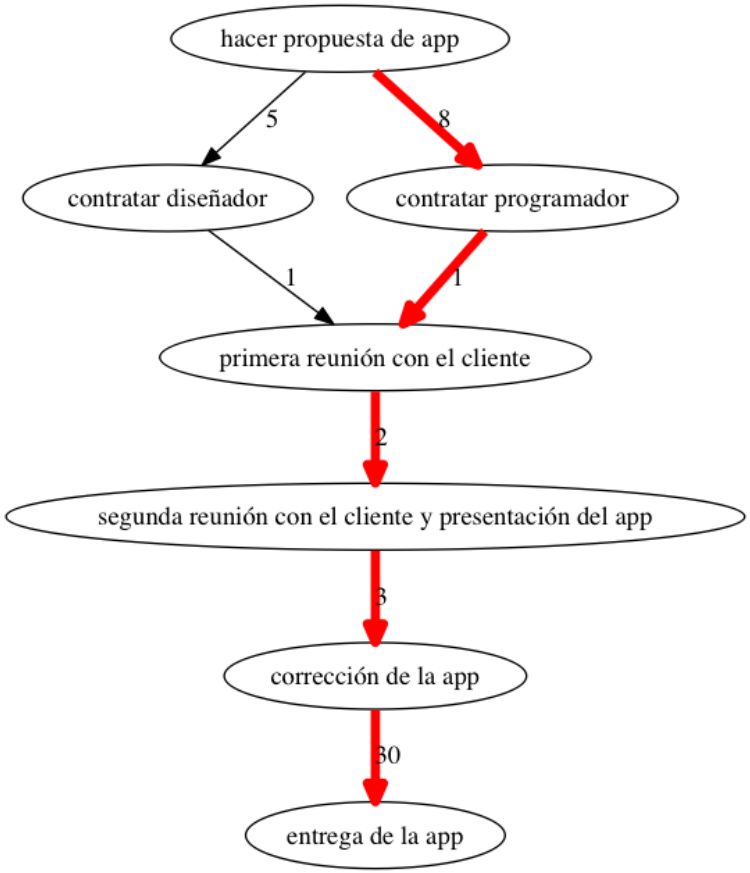


Figura: Grafo con pesos

Otro ejemplo de ordenación lineal



Camino crítico en un grafo



Camino crítico

El camino crítico en un grafo es el camino cuya suma de longitudes de aristas es máxima entre todos los posibles caminos.

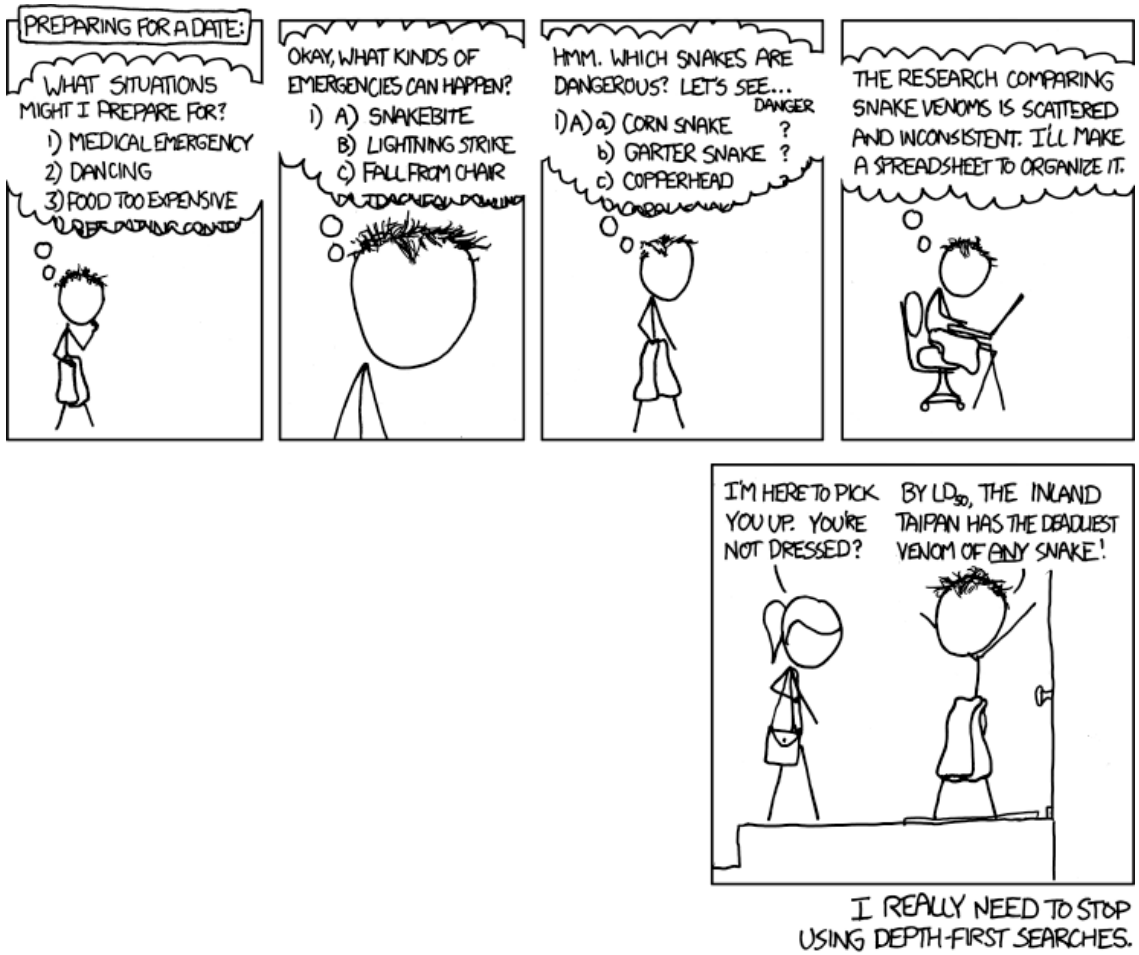
Camino mínimo

El camino mínimo entre dos vértices u,v es cualquier con longitud mínima entre u y v.

Caminos mínimos en DAGs

- Linearizar el DAG.
- Llevar dos arrays distancia y predecesor
- Recorrer los vertices en el orden dado por linearizar e ir actualizando tanto el array de distancias y el predecesor

Problemas de DFS



BFS (Busqueda en anchura)

_fisico.png _fisico.bb fisico.png height7.60422ptfisico.png
width58.55753ptfisico.pngGraphic file (type

bmp)

Imagen tomada de Algorithms (Papadimitriou, C. H. et al)

BFS (Busqueda en anchura)

```
BFS(G,s):
for(int u: G.V):
  distancia[u]=Double.Infinity;
distancia[s] = 0;
Q = Queue(s);
while !Q.isEmpty(){
  u = Q.poll();
  for((u,v): G.E){
    if(distancia[v] == Double.Infinity)
    {
      Q.push(v);
      distancia[v]=distancia[u] + 1;
    }

  }
}
```

BFS con longitudes de aristas enteras

_pesos.png _pesos.bb pesos.png height7.60422ptpesos.png
width60.23045ptpesos.pngGraphic file (type bmp)

Figura: Dos grafos equivalentes para BFS

BFS con longitudes de aristas enteras

1. Elegir un nodo de inicio s.
2. Poner la alarma de ese nodo a 0.
3. Repetir mientras haya alarmas.
    1. Denotemos T el tiempo de la siguiente alarma y nos despertamos en u.
    2. Ponemos la distancia de u a T.
    3. A cada uno de sus vecinos le asignamos el menor valor entre la alarma que da T mas la longitud de la arista que los une o la que tienen.

El algoritmo de Dijkstra

El algoritmo de Dijkstra implementa el sistema de alarmas con una cola de prioridad que tiene las siguientes operaciones:

- Insert
- Decrease-key
- Delete-min
- Make-queue

El algoritmo de Dijkstra

```
update(u,v):
if (dist[v] > dist(u) + l(u,v))
{
    dist[v] = dist(u) + l(u,v);
    prev[v] = u;
    Decrease-key(H,v);
}
```

El algoritmo de Dijkstra

```
dijkstra (G, l, s):
for (v: V)
{
dist[v] = Infinity;
prev[v] = nil;
}
dist[s] = 0;
H = makequeue(s); // Hacer la cola de prioridad
while(!H.empty())
{
  u = H.Delete-min();
  for ((u,v): E)
  {
  update(u,v);
  }

}
```

El algoritmo de Dijkstra con pesos negátivos

_negativos.png _negativos.bb _negativos.bb" negativos.pngGraphic file

(type bmp)

Figura: Grafo con pesos negativos

El algoritmo de Bellman-Ford

```
BellmanFord (G, l, s):
for (v: V)
{
dist[v] = Infinity;
prev[v] = nil;
}
dist[s] = 0;
for(v:V)
{
  for(e:E)
  {
    update(e);
  }
}
```