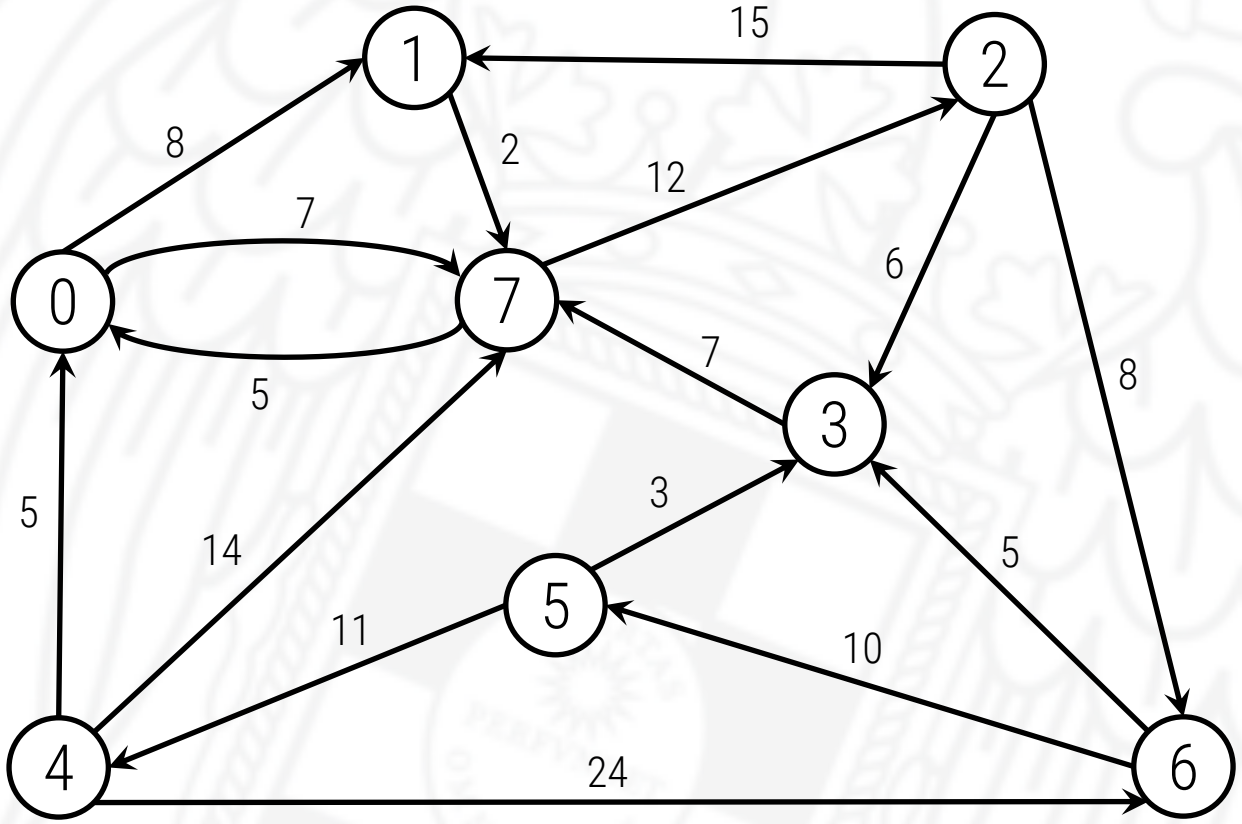
DIGRAFOS VALORADOS

UNIVERSIDAD COMPLUTENSE MADRID

ALBERTO VERDEJO

Digrafos valorados

- Grafos con aristas orientadas que tienen asociado un valor (peso, coste).



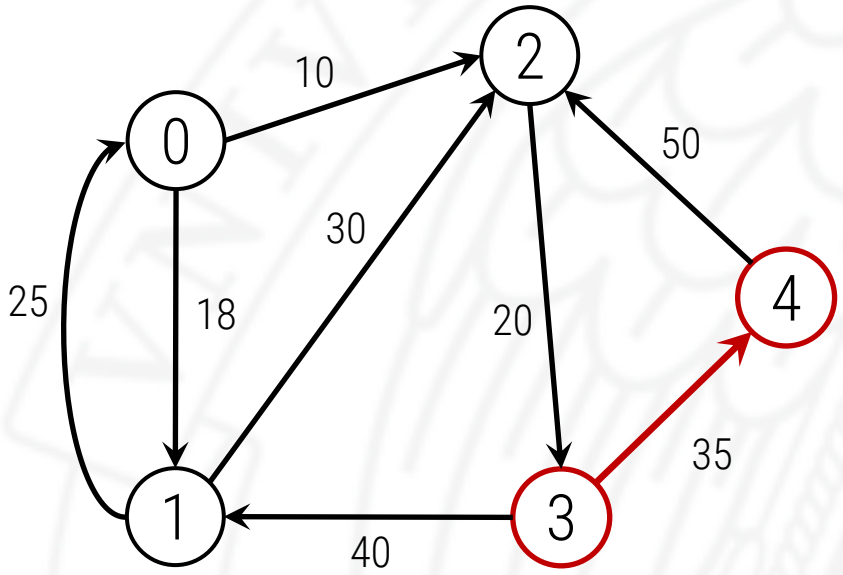
Aristas dirigidas

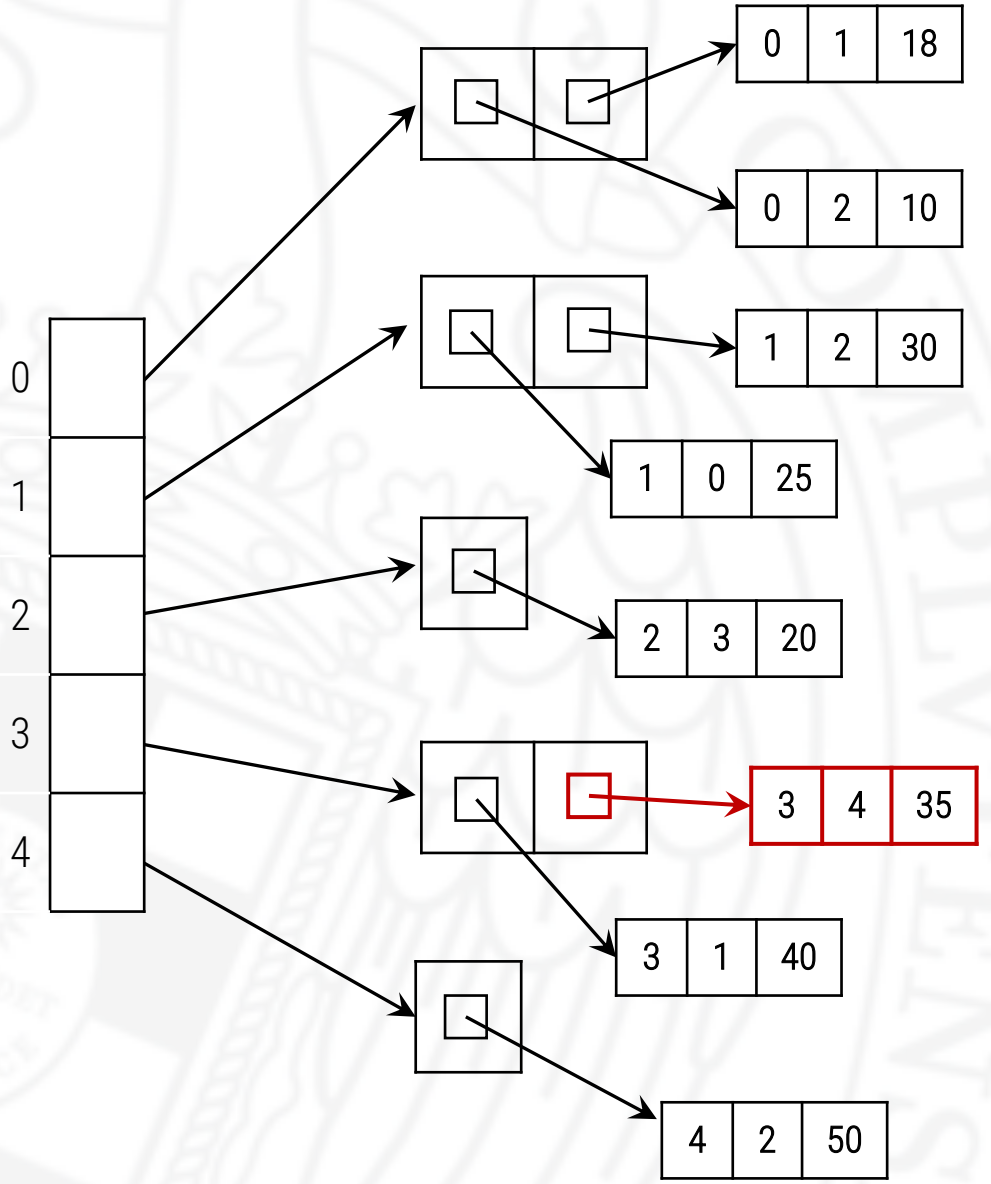
DigrafoValorado.h

```
template <typename Valor>
class AristaDirigida {
public:
   AristaDirigida(int v, int w, Valor valor);
   int desde() const;
   int hasta() const;
   Valor valor() const;
   bool operator<(AristaDirigida<Valor> const& b) const;
   bool operator>(AristaDirigida<Valor> const& b) const;
};
```

Listas de adyacentes

DigrafoValorado.h





Digrafo valorado

DigrafoValorado.h

```
template <typename Valor>
class DigrafoValorado {
public:
   DigrafoValorado(int V);
   void ponArista(AristaDirigida<Valor> arista);
   int V() const;
   int A() const;
   AdysDirVal<Valor> const& ady(int v) const;
   DigrafoValorado<Valor> inverso() const;
};
```

Digrafo valorado

DigrafoValorado.h

```
void ponArista(AristaDirigida<Valor> arista) {
    int v = arista.desde(), w = arista.hasta();
    if (v < 0 || v >= _V || w < 0 || w >= _V)
        throw std::invalid_argument("Vertice inexistente");
    ++_A;
    _ady[v].push_back(arista);
}
```

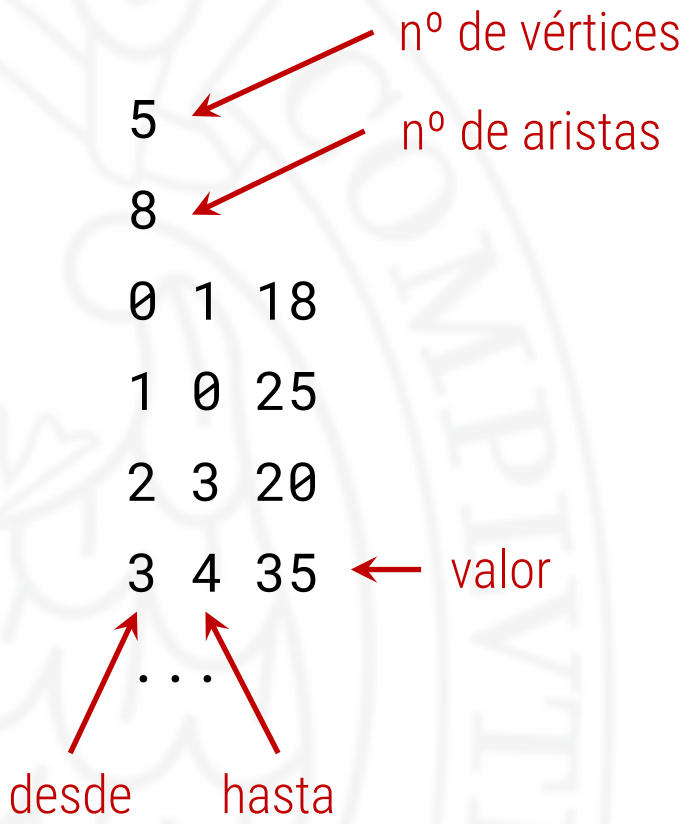
Digrafo valorado

DigrafoValorado.h

```
DigrafoValorado<Valor> inverso() const {
    DigrafoValorado<Valor> inv(_V);
    for (auto v = 0; v < _V; ++v) {
        for (auto a : _ady[v]) {
            inv.ponArista({a.hasta(), a.desde(), a.valor()});
        }
    }
    return inv;
}
```

Construcción de un digrafo valorado

```
bool resuelveCaso() {
    int V, A;
    cin >> V >> A;
    DigrafoValorado<int> digrafo(V);
    int v, w, valor;
    for (int i = 0; i < A; ++i) {
        cin >> v >> w >> valor;
        digrafo.ponArista({v, w, valor});
    }
    ...
```



Recorrido en profundidad de un digrafo

```
// visita los vértices alcanzables desde v respetando el umbral
void dfs(DigrafoValorado<int> const& g, int v, int ancho) {
    visit[v] = true;
    for (auto a : g.ady(v)) {
        if (a.valor() > ancho) {
            int w = a.hasta();
            if (!visit[w])
                dfs(g, w, ancho);
        }
    }
}
```