Grafos: Caminos más cortos

Análisis y Diseño de Algoritmos

Prof. Isabel Besembel Carrera

Caminos mínimos

- Sea un digrafo etiquetado DG={N, A} cuyas aristas tienen pesos no negativos
- Problema: encontrar todos los caminos mínimos desde un nodo origen hacia todos los demás nodos de DG
- Solución: algoritmo de Dijkstra
- Características:
  - Propiedad invariante: N=S∪C, donde S={nodos seleccionados con su distancia mínima desde el origen} y C={resto de los nodos cuya distancia desde el origen es desconocida}
  - En cada paso se selecciona un nodo de C cuya distancia al origen sea mínima

Algoritmo de Dijkstra

- Camino especial: aquel cuyos nodos pertenecen a S
- Vector **distancia** contiene la longitud más corta del camino especial hasta cualquier nodo
- Cuando se desea incluir v en S, el camino especial más corto hasta v es el camino más corto de los caminos posibles hasta v
- Al finalizar, todos los nodos de N se encuentran en S y todos los caminos desde el origen hasta algún otro nodo son especiales
- **distancia** contiene la solución del problema de los caminos mínimos

Algoritmo de Dijkstra

- L[1..n, 1..n] matriz de pesos o longitudes desde el nodo i al nodo j
- D[2..n] vector de distancias
- Algoritmo genérico

dijkstra(Arreglo[1..n, 1..n]de Entero+ L):Arreglo[2..n]de Entero+

1 $C = \{2, 3, \ldots, n\}$

2 [$D_i = L_{1,i}$] $i = 2, n$

3 [v = algún elemento de C que minimiza D_v

$C = C - \{v\}$

para cada $w \in C$

$D_w = \min(D_w, D_v + L_{v,w})$

fpc] $i = 2, n$

4 regrese D

Algoritmo de Dijkstra

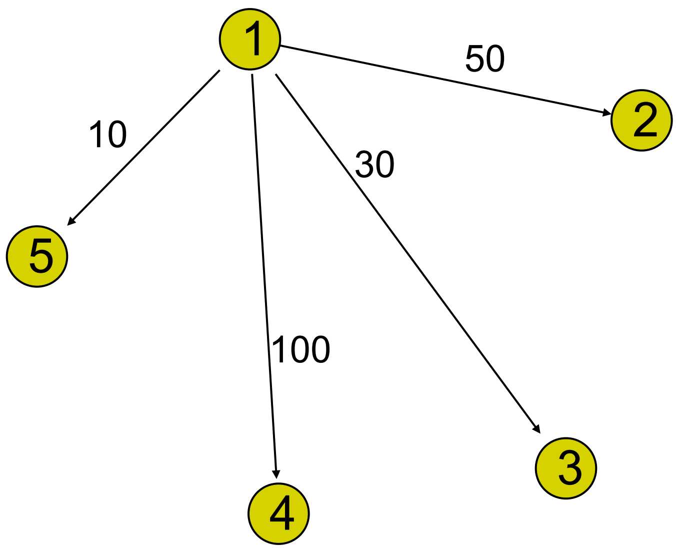
Digrafo etiquetado

Algoritmo de Dijkstra

Origen el nodo 1

Digrafo etiquetado

Algoritmo de Dijkstra

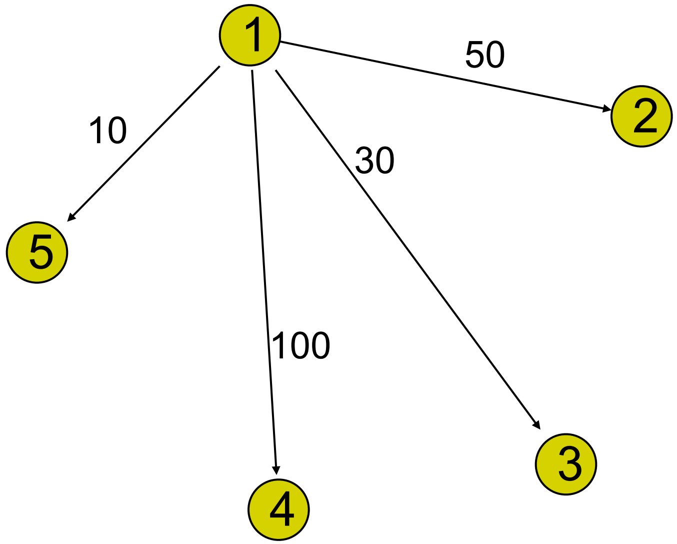


Digrafo etiquetado

Origen el nodo 1

Paso	v	C	D
Inicio	----	{2, 3, 4, 5}	[50, 30, 100, 10]

Algoritmo de Dijkstra

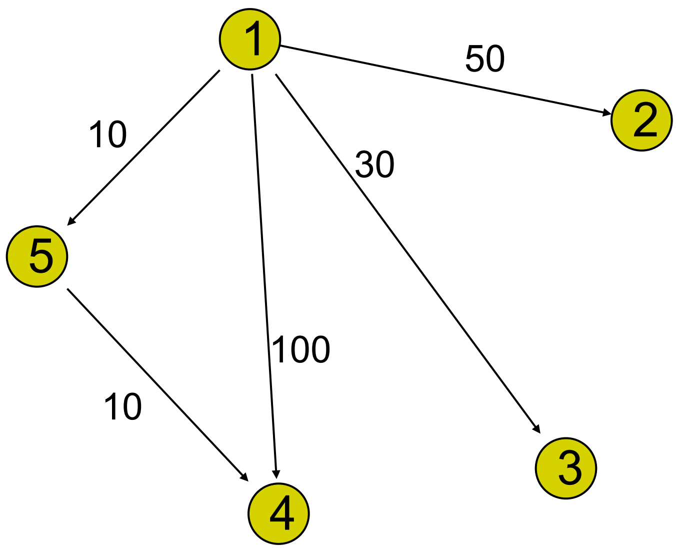


Digrafo etiquetado

Origen el nodo 1

Paso	v	C	D
Inicio	----	{2, 3, 4, 5}	[50, 30, 100, 10]
1	5	{2, 3, 4}	[50, 30, 20, 10]

Algoritmo de Dijkstra

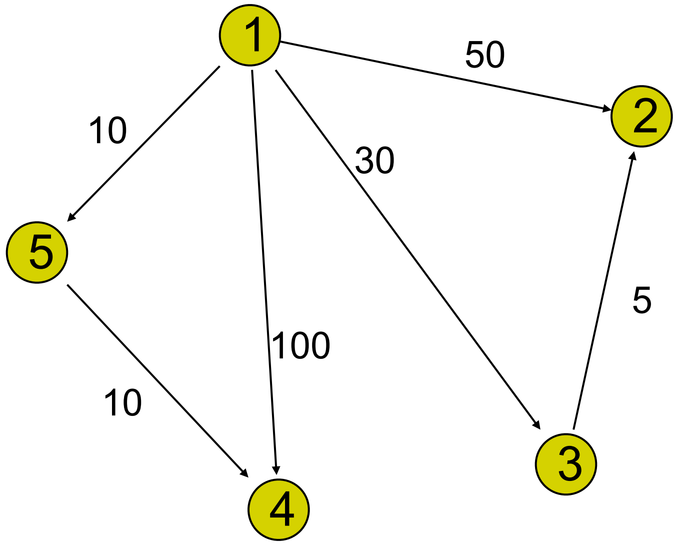


Digrafo etiquetado

Origen el nodo 1

Paso	v	C	D
Inicio	----	{2, 3, 4, 5}	[50, 30, 100, 10]
1	5	{2, 3, 4}	[50, 30, 20, 10]
2	4	{2, 3}	[40, 30, 20, 10]

Algoritmo de Dijkstra

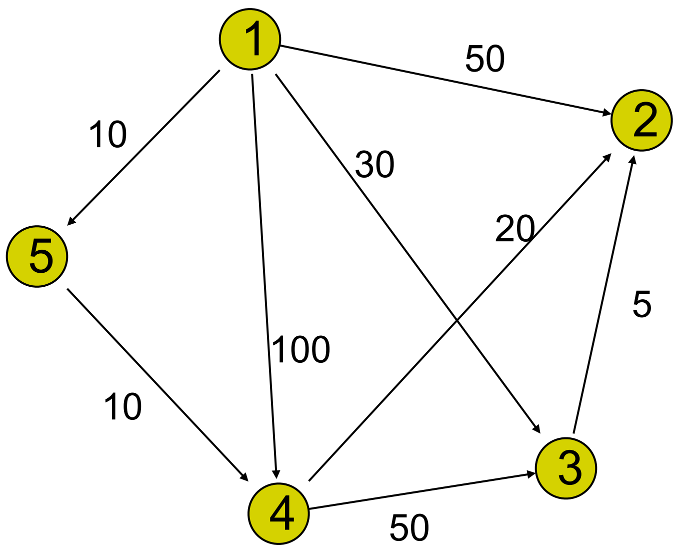


Digrafo etiquetado

Origen el nodo 1

Paso	v	C	D
Inicio	----	{2, 3, 4, 5}	[50, 30, 100, 10]
1	5	{2, 3, 4}	[50, 30, 20, 10]
2	4	{2, 3}	[40, 30, 20, 10]
3	3	{2}	[35, 30, 20, 10]

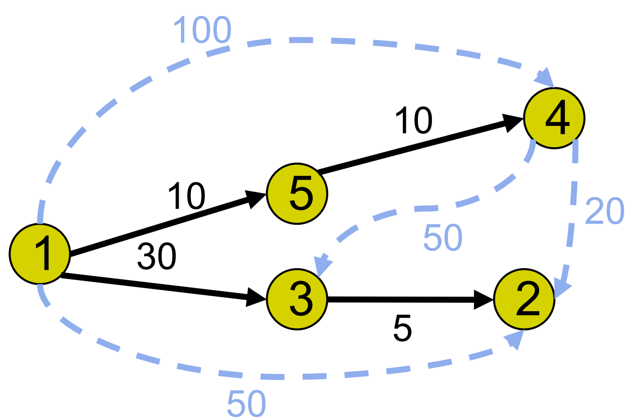
Algoritmo de Dijkstra



Digrafo etiquetado

Origen el nodo 1

Paso	v	C	D
Inicio	----	{2, 3, 4, 5}	[50, 30, 100, 10]
1	5	{2, 3, 4}	[50, 30, 20, 10]
2	4	{2, 3}	[40, 30, 20, 10]
3	3	{2}	[35, 30, 20, 10]



Algoritmo de Dijkstra

- Para saber por donde pasan los caminos mínimos, se añade el vector **padre** (P)
- P[2..n] vector que guarda el nodo predecesor o padre del nodo i
- Algoritmo genérico

dijkstra(Arreglo[1..n, 1..n]de Entero+: L, Arreglo[2..n]de Entero+: D, P)

1 $C = \{2, 3, \dots, n\}$

2 [D_i , $P_i = L_{1,i}$, 1] i = 2, n

3 [v = algún elemento de C que minimiza D_v

$C = C - \{v\}$

para cada $w \in C$

Si ($D_w > D_v + L_{v,w}$) entonces

$D_w = D_v + L_{v,w}$

$P_w = v$

fsi

fpc] i = 2, n

Algoritmo de Dijkstra

Digrafo etiquetado

Algoritmo de Dijkstra

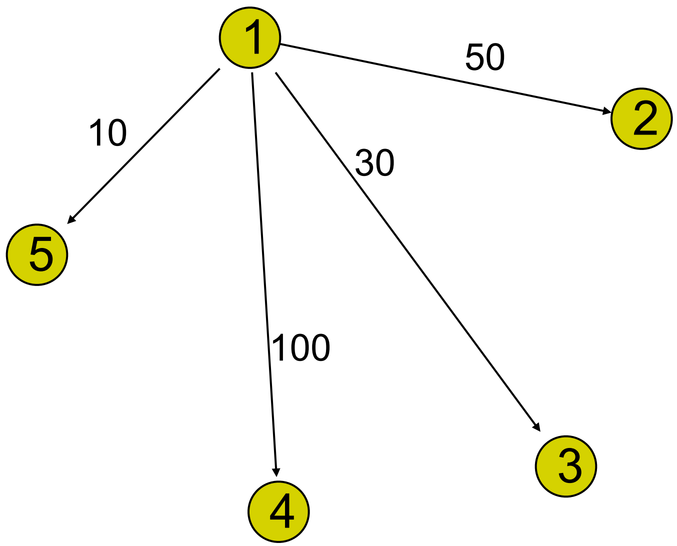
Origen el nodo 1

Digrafo etiquetado

Algoritmo de Dijkstra

Origen el nodo 1

Paso	v	C	D	P
Inicio	----	{2, 3, 4, 5}	[50, 30, 100,10]	[1, 1, 1, 1]

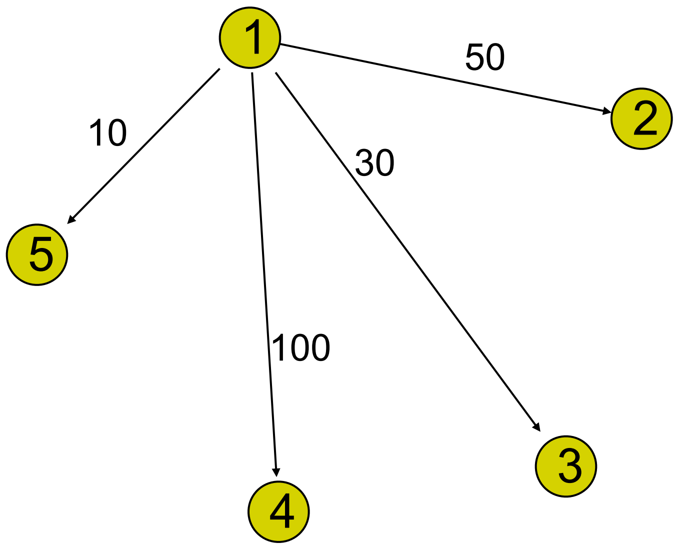


Digrafo etiquetado

Algoritmo de Dijkstra

Origen el nodo 1

Paso	v	C	D	P
Inicio	----	{2, 3, 4, 5}	[50, 30, 100,10]	[1, 1, 1, 1]
1	5	{2, 3, 4}	[50, 30, 20, 10]	[1, 1, 1, 1]

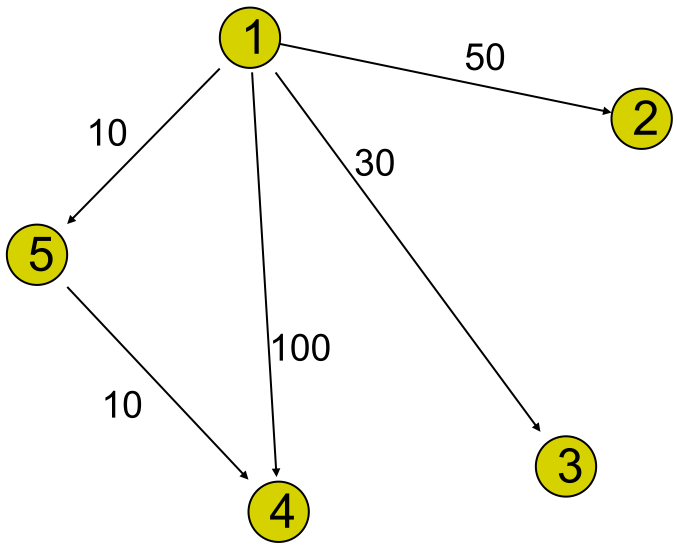


Digrafo etiquetado

Algoritmo de Dijkstra

Origen el nodo 1

Paso	v	C	D	P
Inicio	----	{2, 3, 4, 5}	[50, 30, 100,10]	[1, 1, 1, 1]
1	5	{2, 3, 4}	[50, 30, 20, 10]	[1, 1, 1, 1]
2	4	{2, 3}	[40, 30, 20, 10]	[1, 1, 5, 1]

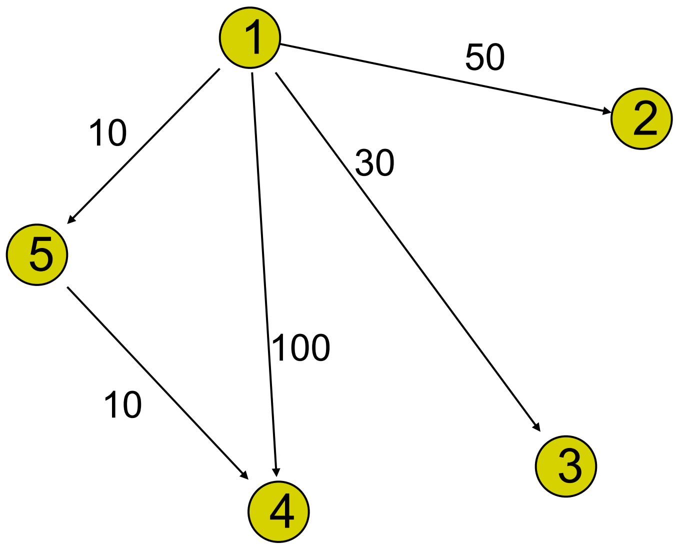


Digrafo etiquetado

Algoritmo de Dijkstra

Origen el nodo 1

Paso	v	C	D	P
Inicio	----	{2, 3, 4, 5}	[50, 30, 100,10]	[1, 1, 1, 1]
1	5	{2, 3, 4}	[50, 30, 20, 10]	[1, 1, 1, 1]
2	4	{2, 3}	[40, 30, 20, 10]	[1, 1, 5, 1]
3	3	{2}	[35, 30, 20, 10]	[1, 1, 5, 1]

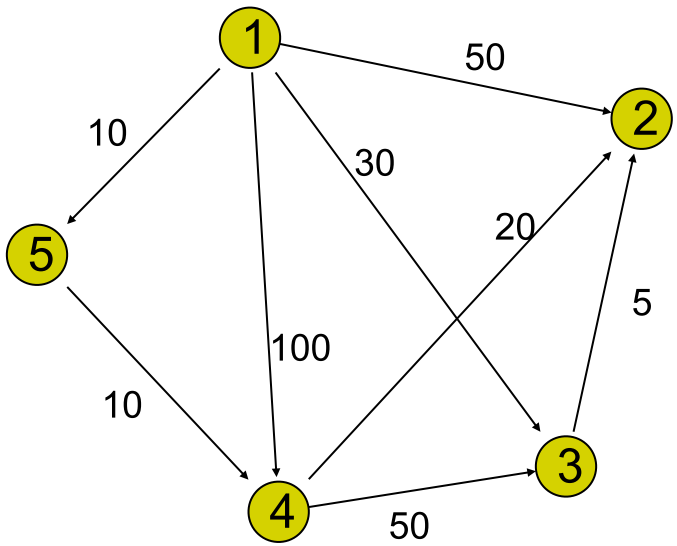


Digrafo etiquetado

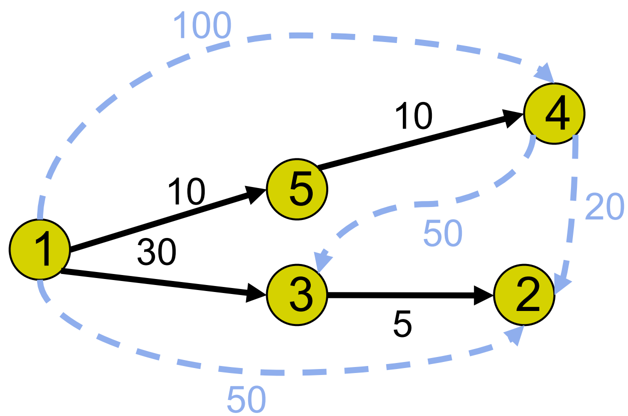
Algoritmo de Dijkstra

Origen el nodo 1

Paso	v	C	D	P
Inicio	----	{2, 3, 4, 5}	[50, 30, 100,10]	[1, 1, 1, 1]
1	5	{2, 3, 4}	[50, 30, 20, 10]	[1, 1, 1, 1]
2	4	{2, 3}	[40, 30, 20, 10]	[1, 1, 5, 1]
3	3	{2}	[35, 30, 20, 10]	[1, 1, 5, 1]
				[3, 1, 5, 1]



Digrafo etiquetado



Teorema

- El algoritmo de Dijkstra halla los caminos más cortos desde un único origen hasta los demás nodos del grafo
- Demostración: por inducción matemática que:
  A. **Si un nodo i ≠ 1 está en S, entonces Di da la longitud del camino más corto desde el origen hasta i**
  B. **Si un nodo i no está en S, entonces Di da la longitud del camino especial más corto desde el origen hasta i**

Base: sólo el nodo 1 (origen) está en S, por tanto A es cierto. Para el resto de los nodos, el único camino desde el origen es el camino directo, por lo tanto B es cierto también

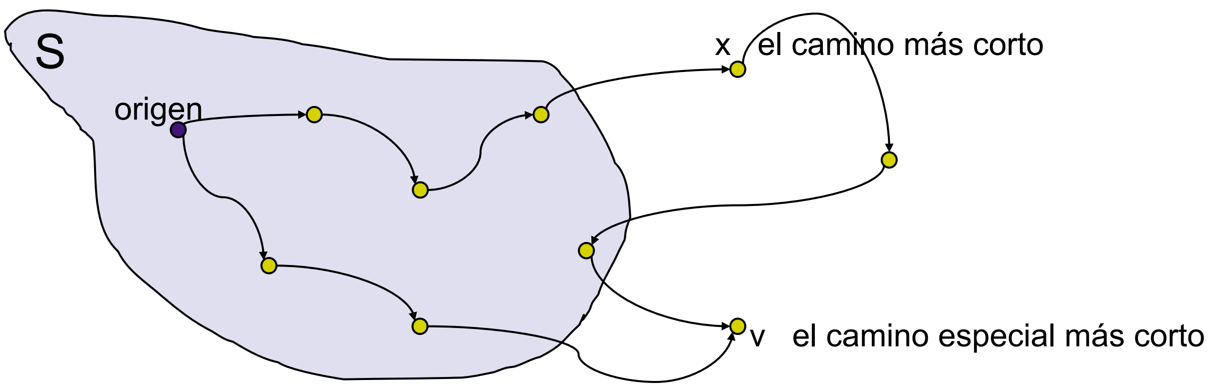
Hipótesis inductiva: tanto A como B son válidos antes de añadir un nodo v a S

Inductiva:

- Caso A: ∀ nodo en S antes de añadir v, no cambia nada, así que A es válido. Antes de añadir v a S, hay que comprobar que D[v] sea el camino más corto del origen a v

Continuación de la demostración

- Por la hipótesis, es la longitud del camino especial más corto, por lo tanto hay que verificar que el camino más corto del origen a v no pase por nodos que no pertenecen a S.
- Suponga lo contrario, sea x ∉ S, el camino del origen a x es un camino especial, D[x] es la distancia por B. Así, la distancia total hasta v a través de x no es más corta que D[x], pues las longitudes son no negativas ⇒ D[x] > D[v], ya que el algoritmo ha seleccionado a v antes de x, por lo tanto la distancia total hasta v a través de x es como mínimo D[v] y el camino a través de x no puede ser más corto que el camino especial que lleva hasta v



Continuación de la demostración

- Por lo cual al añadir v a S, la parte A es válida.
- Caso B: Sea $w \neq v$, $w \notin S$. Cuando v se añade a S, hay dos posibilidades para el camino especial más corto del origen a w; o bien no cambia, o ahora pasa por v. En este último caso, sea x el último nodo de S visitado antes de llegar a w. La longitud de este camino es $D[x] + L[x, w]$, pero ese cálculo ya fue hecho para todo nodo $x \in S$, $x \neq v$, cuando se añadió x a S y por ello, $D[x]$ no ha variado desde entonces. Por lo tanto, el nuevo valor de $D[w]$ se puede calcular comparando el valor anterior con $D[v] + L[v, w]$.
- Ya que el algoritmo lo hace explícitamente, asegura el caso B que es válido cuando se añade v a S.

Análisis del algoritmo de Dijkstra

- Si se implementa tal y como aparece $\Theta(n^2)$, preferible en el caso de digrafos densos
- Si a << n^2, es preferible implementar el digrafo con listas de adyacencia y utilizar un montículo, donde cada nodo tiene el número del nodo del grafo en el conjunto C y su distancia, ordenado por la menor de las D[v], así el elemento v de C que minimiza D[v] siempre se encontrará en la raíz.
  - La iniciación del montículo invertido es $\Theta(n)$
  - La eliminación de la raíz del montículo se hace n-2 veces en $\Theta((A+N) \lg N)$
  - Si el digrafo es conexo, $A \geq N-1$, T(n) está en $\Theta(A \lg N)$

Caminos más cortos desde un nodo dado

- Dado un digrafo G = (N, A) etiquetado y conexo con pesos w: A → $\Re$. Para un camino **p** = ⟨n0, n1, ..., nk⟩ , w(p)= sumatoria de i=1 hasta n de w(ni−1, ni), se define el camino con el peso mínimo de u a v como: δ (u, v) = min { w(p) : longitud del camino p de u a v }, si ∃ p de u a v e ∞ sino
- Los pesos pueden ser cualquier medida como: distancia, tiempo, costos, etc. que se acumulen linealmente a lo largo del camino y que se desea minimizar.
- Variantes:
  - Problema de los caminos más cortos a un único destino: Encontrar **p** dado el destino **d** desde cada nodo **v**. Solución: Se invierten los arcos y se convierte en el problema inicial.
  - Problema de los caminos más cortos para un par de nodos dados: Es el mismo problema inicial. No hay otro algoritmo mejor.
  - Problema de los caminos más cortos para todos los pares de nodos: Se resuelve corriendo el algoritmo del problema inicial para cada nodo. Se puede tratar mejor con otro algoritmo.

Dijkstra con algunos pesos negativos

- Pesos negativos: Si $G = (N, A)$ contiene ciclos con pesos positivos alcanzables desde la fuente **s**, entonces $\forall\, v \in N$, $\delta(s, v)$ está bien definido, así haya algún peso negativo. Si hay algún ciclo con peso negativo en algún camino de **s** a **v** $\Rightarrow \delta(s, v) = -\infty$.
- Representación:
- Con el padre(v) que induce el grafo predecesor $G\pi = (N\pi, A\pi)$ donde:

  $N\pi = \{ v \in N \;/\; \text{padre}(v) \neq \text{Nulo} \} \cup \{ s \}$

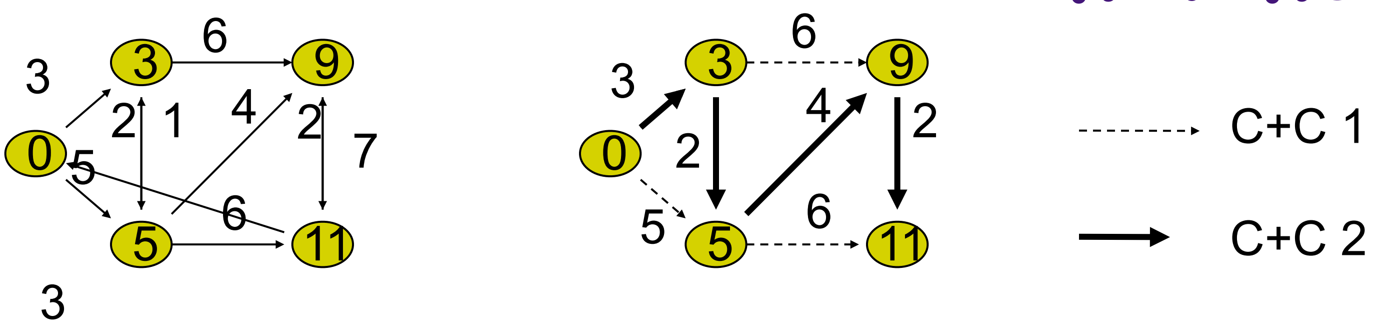
  $A\pi = \{ (\text{padre}(v), v) \in A \;/\; v \in N\pi - \{ s \} \}$
- El árbol del camino más corto (C+C) contiene los C+C desde la fuente, definido en términos de arcos etiquetados con pesos en vez de número de arcos.
- Un árbol del C+C con raíz en **s** es un subgrafo dirigido $G' = (N', A')$ donde $N' \subseteq N$ y $A' \subseteq A$ /

  N' es el conjunto de todos los nodos alcanzables desde **s** en G.

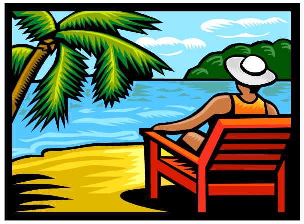
  G' forma un árbol cuya raíz es **s**, y

  $\forall\, v \in N'$ el único camino simple de **s** a **v** en G' es el C+C de **s** a **v** en G.

Propiedades de los caminos mínimos



- Convenciones: Si $a \neq -\infty \Rightarrow$
  $a + \infty = \infty + a = \infty$
  $a + (-\infty) = (-\infty) + a = -\infty$
- Propiedad de los C+C:
  - Un C+C entre dos nodos contiene otros C+Cs dentro.
  - Sea G =(N, A) un digrafo etiquetado con w: $A \to \Re$ y **p** = $\langle$n0, n1, ..., nk$\rangle$, el C+C de n1 a nk, para cualquier i, j / $1 \leq i \leq j \leq k$ pi j = $\langle$ni, ni+1, ..., nj$\rangle$ un subcamino de **p** de ni a nj $\Rightarrow$ pi j es el C+C entre ni y nj.
  - Suponga que **p** de **s** a **v** puede ser descompuesto en p' de s a u $\to$ v para algún **u** y **p'**, entonces el peso del C+C de **s** a **v** $\delta(s, v) = \delta(s, u) + w(u, v)$.
  - Para todo $(u, v) \in A$, se tiene que $\delta(s, v) \leq \delta(s, u) + w(u, v)$.



Relajación

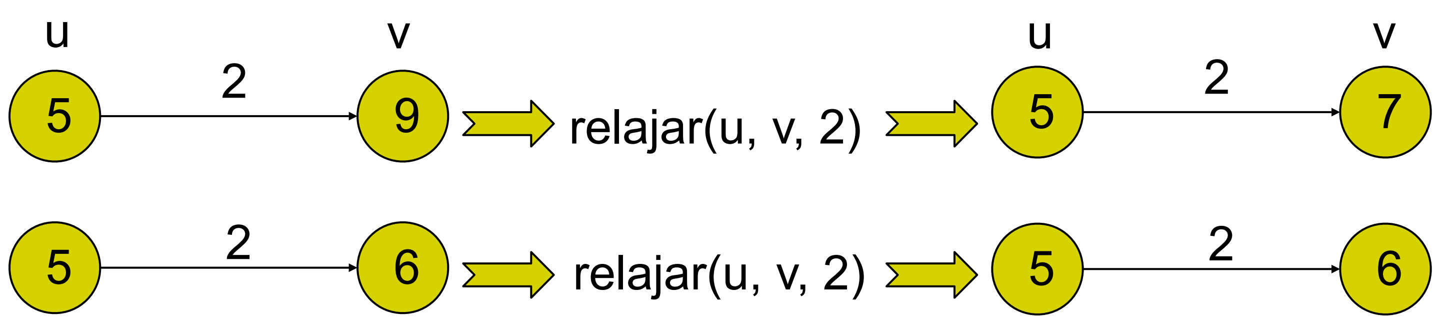
- Relajación: ∀ v ∈ N, se mantiene un atributo d(v) que es el límite superior de los pesos del C+C de **s** a **v**, d(v) se denomina el estimado del peso del C+C y se inicia con

26/11/98 iniciar(Nodo: s) {pre: n > 0 ∧ s ∈ N } {pos: n > 0 }		
1	[v.D(∞) v.Padre(Nulo)] v ∈ N	**-D(), Padre():** Operaciones de la clase Nodo.
2	s.D(0)	
3	regrese	

T(n) = O(n)

Proceso de relajación

- El proceso de relajación de un arco (u, v) consiste en probar si se puede mejorar el C+C encontrado a **v** yendo a través de **u** y si eso es posible, actualizar d(v) y padre(v).
- Ejemplo:



Relajar

26/11/98 relajar(Nodo: u, v, Real: w) {pre: $n > 0 \wedge (u, v) \in A$ } {pos: $n > 0$ }		
1	Si (v.D() > u.D() + w(u, v)) entonces v.D(u.D() + w(u, v)) v.Padre(u) fsi	**-D(), Padre():** Operaciones de la clase Nodo.
2	regrese	

- Propiedades de la relajación: Sea G=(N, A) un digrafo etiquetado con $w: A \rightarrow \Re$
  - Sea $(u, v) \in A$ luego de la relajación del arco (u, v) por medio de relajar( ) se tiene que $d(v) \leq d(u) + w(u, v)$
  - Sea $s \in N$ el nodo fuente y G iniciado con iniciar(s), entonces $d(v) \geq \delta(s, v)\ \forall\ v \in N$ y esta invariante se mantiene sobre cualquier secuencia de relajación de los arcos de G. Mas aún, cuando d(v) alcanza su límite inferior $\delta(s, v)$, este nunca cambia.

Propiedades

- Si no hay camino de $s \in N$ a $v \in N$ entonces luego que G es iniciado con iniciar(s), se tiene $d(v) = \delta(s, v)$ y esta igualdad se mantiene como una invariante sobre cualquier secuencia de relajación sobre los arcos de G.
- Sea $s \in N$ y el camino de s a $u \rightarrow v$ el C+C en G para $u, v \in N$. Suponga G.iniciar(s) y luego una secuencia de relajación que incluya G.relajar(u, v, w) sobre los arcos de G. Si $d(u) = \delta(s, u)$ en cualquier momento antes de la llamada, entonces $d(v) = \delta(s, v)$ en cualquier momento después de la llamada.
- La relajación causa que el estimado del C+C descienda monótonamente hacia el peso del C+C actual.
- Sea $s \in N$ el nodo fuente y asuma que G no contiene ciclos de peso negativo que sean alcanzables desde **s**, entonces luego de G.iniciar(s), el subgrafo predecesor $G\pi$ forma un árbol cuya raíz es **s** y cualquier secuencia de relajación sobre los arcos de G mantiene la propiedad como una invariante.
- Sea $s \in N$ el nodo fuente y asuma que G no contiene ciclos de peso negativo que sean alcanzables desde **s**, entonces luego de G.iniciar(s) y cualquier secuencia de etapas de relajación sobre los arcos de G que produce $d(v) = \delta(s, v)\ \forall\ v \in N$, entonces $G\pi$ es el C+C que es el árbol cuya raíz es **s**.

Algoritmo de Dijkstra

- Todos los $w(u, v) \geq 0 \; \forall \; u, v \in N$ y $(u, v) \in A$.
- El algoritmo mantiene un conjunto S de nodos cuyos pesos finales del C+C desde **s** han sido determinados.
- Esto es, $\forall \; v \in S$ se tiene $d(v) = \delta(s, v)$.
- El algoritmo repetidamente selecciona un nodo $u \in N - S$ con el mínimo C+C estimado, inserta **u** en S y relaja todos los arcos que salen de **u**.
- La cola por prioridad C contiene todos los nodos en N – S cuya clave es d(u).
- G está implantado con listas de adyacencia.
- Es un algoritmo incremental de tipo II.

Algoritmo de Dijkstra

- Teorema: Si se ejecuta el algoritmo de Dijkstra en un digrafo etiquetado con $w \geq 0$ y un nodo fuente **s**, al terminar $d(u) = \delta(s, u)\ \forall\ u \in N$.
- **Corolario**: y $G\pi$ es el árbol del C+C cuya raíz es **s**.

Dijkstra

26/11/98

C+Cdijkstra(Nodo: s): Conjunto

{pre: $n > 0 \wedge s \in N$ }

{pos: $n > 0$ }

```
1  iniciar(s)
2  [ C.entrar(u) ] u ∈ N
3  (¬C.vacíaColaP() ) [ u = C.min( )
                        C.salir( )
                        S = S U { u }
                        [relajar(u, v, w) ] v ∈ listaAdy(u) ]
4  regrese  S
```

- **-iniciar(), relajar():** Operaciones de la clase Grafo.
- **-C:** ColaPrioridad. Cola de prioridad que contiene los nodos que aún no han sido pasados al C+C.
- **-S:** Conjunto. Contiene el C+C.
- **-entrar(), vacíaColaP(), min(), salir().** Operaciones de la clase ColaPrioridad.

Análisis

- Si C se implanta como un vector, entonces $T(n) = O(N^2 + A) = O(N^2)$
- Si C se implanta como un montículo binario, entonces $T(n) = O(A \lg N)$
- Si C se implanta como un montículo binomial, entonces $T(n) = O(N \lg N + A)$

Caminos más cortos para grafos implementados con el método secuencial

Marzo, 2005

caminos+cortos(TipoEle: n):Arreglo[100] de Salida[TipoEle]

{pre: g (i, j) ≥ 0 ∀ i,j} {pos: g (i, j) ≥ 0 ∀ i,j}

```
1   ele = t.conTabla(n)
2   Si (ele.Clave() = n)  entonces
3     [cam( i ).dis(maximo)
       cam( i ).padre({TipoNoDef}) ]  i = 1, 100
      j = ele.Resto()
      cam[j].dis(0)
      c.enCola(j)
      ( ¬c.vacíaCola() )
          [ i = c.primero()
           c.desenCola()
           Si (g (i, j)  = 1) ∧ (cam(j).distancia() = maximo) entonces
                cam(j).distancia(cam(i).distancia()+1)
                cam(j).padre(dir(i))
                c.enCola(j)
             fsi  ] j = 1, pos
    fsi
4   regresa cam
```

-t, g: Definido en DigrafoMat.
-maximo: Máximo valor que se puede almacenar como distancia.
-cam: Arreglo[100] de Salida[TipoEle]. arreglo con la salida de los caminos mas cortos a partir del nodo n.
-ele: TipoEleTab. Definido en DigrafoMat.
-i, j, k: Entero: Variable con la posicion de los nodos en la matriz
-c: Cola[Entero]. Cola de los nodos por procesar.

$$T(n) = O(N + A)$$

Caminos más cortos para grafos implementados con el método enlazado

Marzo, 2005

caminos+cortos(Entero: s): Arreglo[n]De [Entero]

{pre: $n > 0 \wedge s \in \{N\}$}

{pos: $n > 0 \wedge G' = G \wedge$ padre $\supset$ árbol en amplitud de s}

```
1   [ Si ( i ≠ s ) entonces
          color(i), distancia(i), padre(i) = 'blanco', MV, 0
      sino
          color(s), distancia(s), padre(s)  = 'gris', 0, 0
      fsi  ] i = 1, n
2   c.enCola(s)
3   (¬ c.vaciaCola()) [ v = c.primero( )
                        [ Si ( color(k) = 'blanco')  entonces
                              color(k) = 'gris'
                              distancia(k) = distancia(k) + 1
                              padre(k) = v
                              c.enCola(k)
                         fsi  ] ∀ k ∈ v.listaAdya
                        c.desenCola( )
                        color(v) = 'negro'  ]
4   regrese  padre
```

-i, v, k: Entero. Variables auxiliares para recorrer los nodos.

-color, distancia, padre: Arreglo[n] De [Entero]. Variables adicionales para controlar el número de veces que se ha tratado el nodo, su distancia a s y su nodo predecesor inmediato, respectivamente.

-enCola(), desenCola(), primero(), vaciaCola(). Funciones de la clase Cola[X].

-c: Cola[Entero]. Cola que mantiene los nodos que tienen color gris, es decir que ya han sido tocados, pero aún no han sido procesados.

-MV. Entero. Máximo valor.

$$T(n) = O(N + A)$$

Problema del morral (v1)

- Se tienen n objetos y 1 morral.
- Cada objeto i tiene un peso w_i y un valor positivo v_i
- Capacidad máxima del morral W
- Objetivo: llenar el morral para maximizar el valor de los objetos transportados, respetando la capacidad del mismo
- Suposición: los objetos se pueden picar en fracciones más pequeñas, fracción x_i del objeto i, con $0 \leq x_i \leq 1$.
- El objeto i contribuye en $x_i w_i$ al peso total del morral y en $x_i v_i$ al valor total de la carga.

Problema del morral (v1)

- Problema:

$$máx \sum_{i=1}^{n} x_i v_i \text{ con la restricción } \sum_{i=1}^{n} x_i v_i \leq W$$

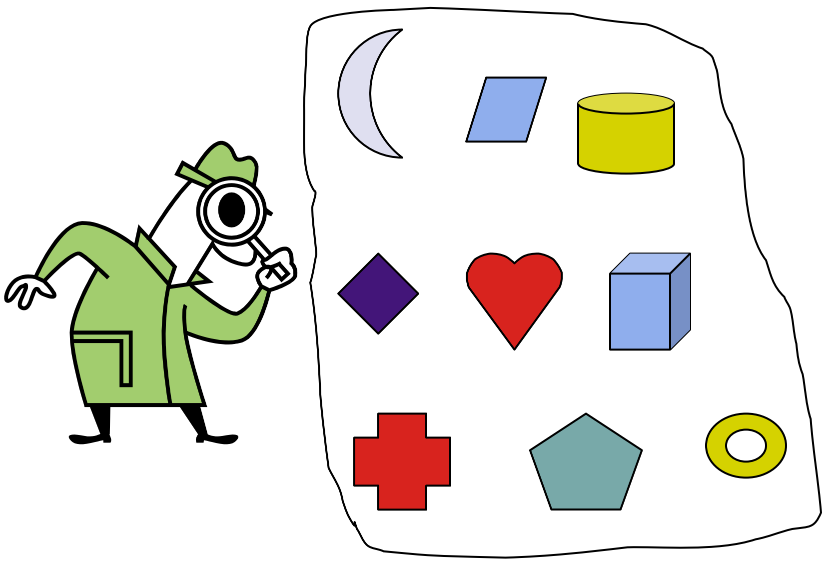
Donde $v_i > 0$, $w_i > 0$ y $0 \leq x_i \leq 1$ para $1 \leq i \leq n$.

Problema del morral (v1)

- Problema:

$$m\acute{a}x\sum_{i=1}^{n} x_i v_i \text{ con la restricción } \sum_{i=1}^{n} x_i v_i \leq W$$

Donde $v_i > 0$, $w_i > 0$ y $0 \leq x_i \leq 1$ para $1 \leq i \leq n$.

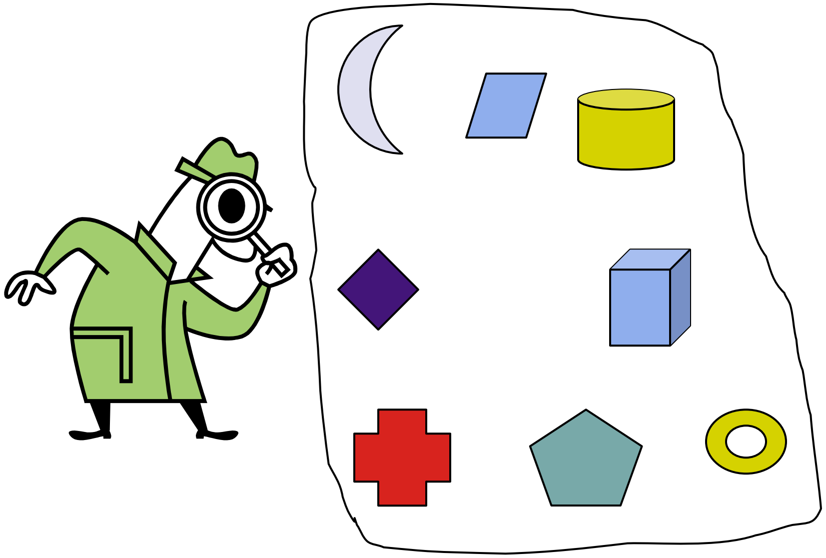


Problema del morral (v1)

- Problema:

$$m\acute{a}x \sum_{i=1}^{n} x_i v_i \text{ con la restricción } \sum_{i=1}^{n} x_i v_i \leq W$$

Donde $v_i > 0$, $w_i > 0$ y $0 \leq x_i \leq 1$ para $1 \leq i \leq n$.

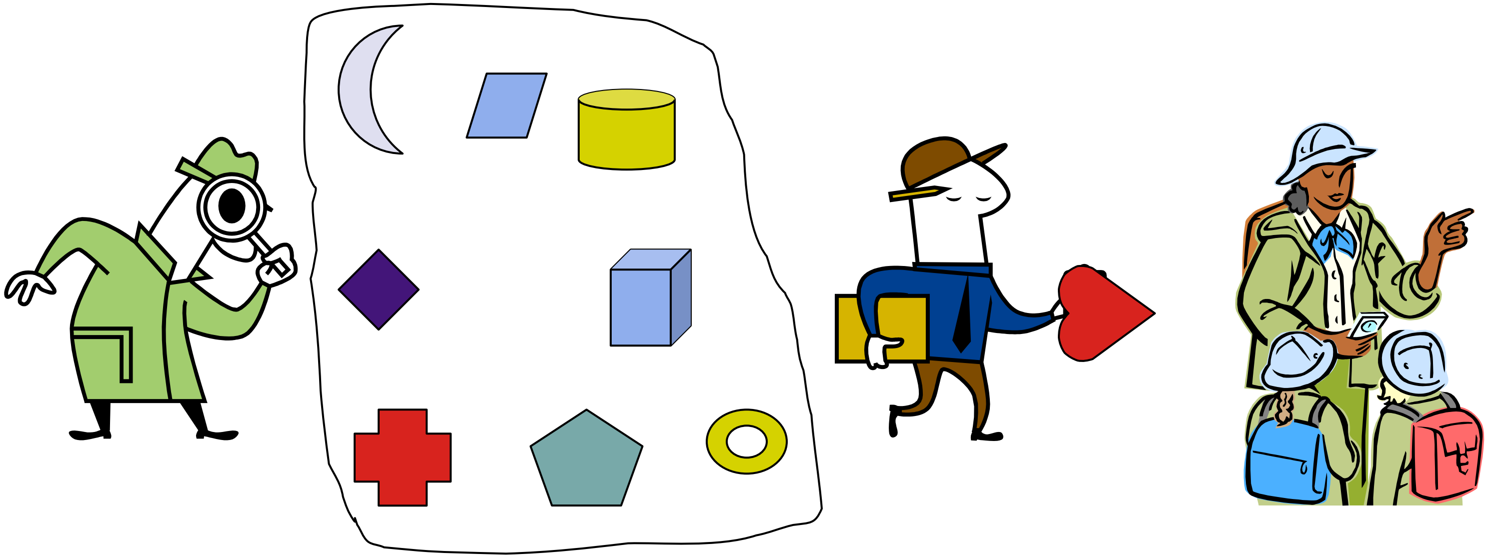


Problema del morral (v1)

- Problema:

$$\textit{máx}\sum_{i=1}^{n} x_i v_i \text{ con la restricción } \sum_{i=1}^{n} x_i v_i \leq W$$

Donde $v_i > 0$, $w_i > 0$ y $0 \leq x_i \leq 1$ para $1 \leq i \leq n$.

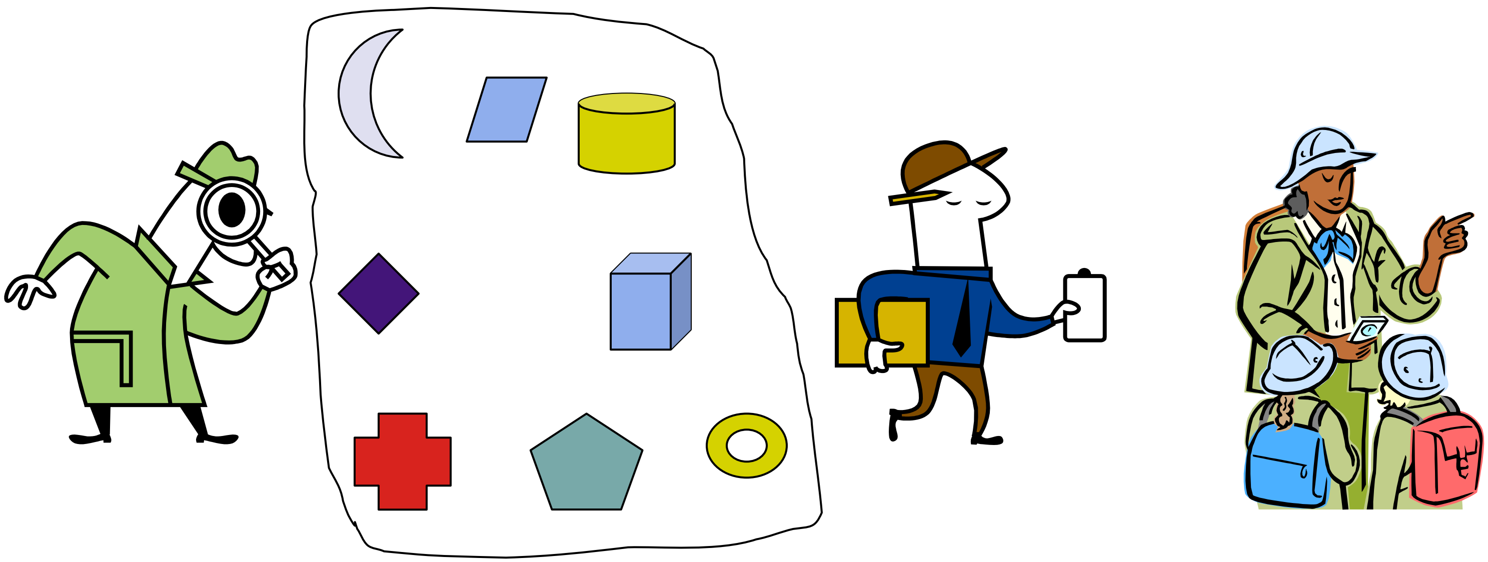


Problema del morral (v1)

- Problema:

$$m\acute{a}x \sum_{i=1}^{n} x_i v_i \text{ con la restricción } \sum_{i=1}^{n} x_i v_i \leq W$$

Donde $v_i > 0$, $w_i > 0$ y $0 \leq x_i \leq 1$ para $1 \leq i \leq n$.

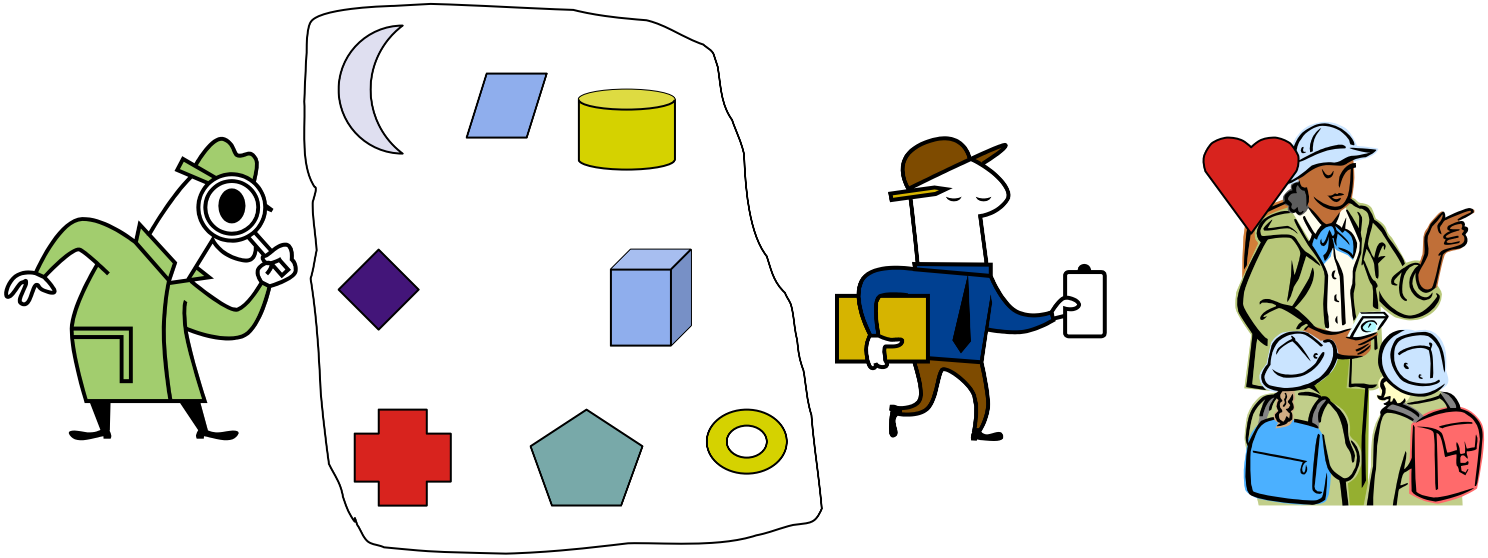


Problema del morral (v1)

- Problema:

$$m\acute{a}x \sum_{i=1}^{n} x_i v_i \text{ con la restricción } \sum_{i=1}^{n} x_i v_i \leq W$$

Donde $v_i > 0$, $w_i > 0$ y $0 \leq x_i \leq 1$ para $1 \leq i \leq n$.

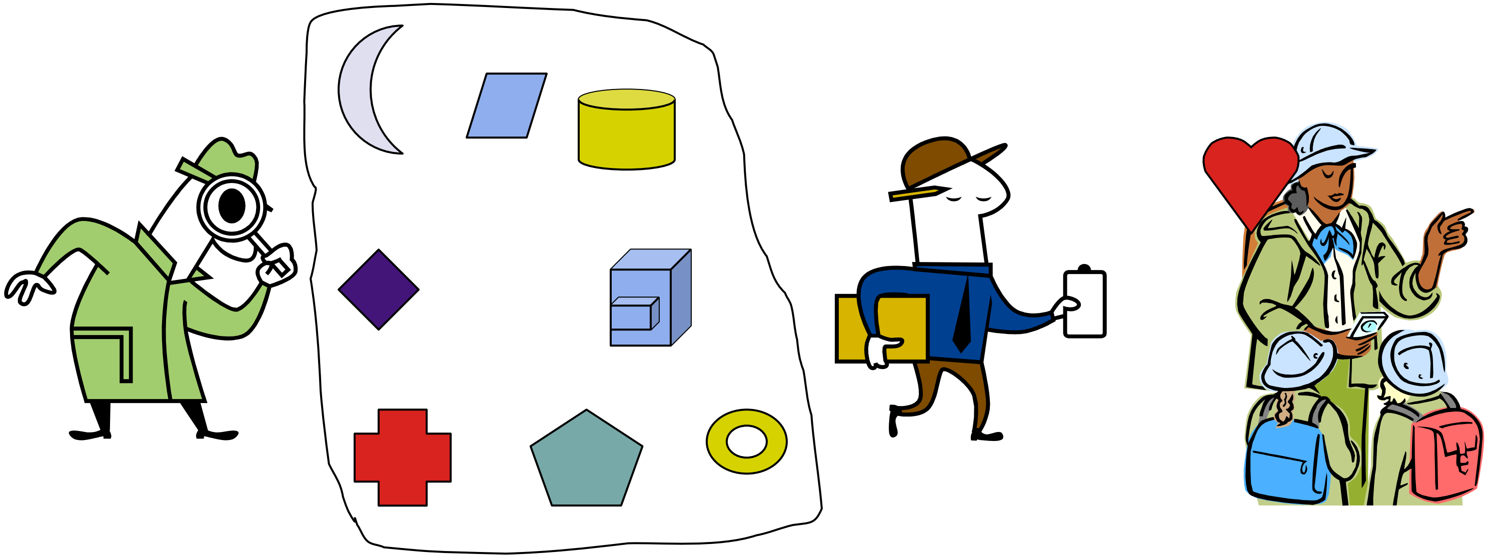


Problema del morral (v1)

- Problema:

$$m\acute{a}x \sum_{i=1}^{n} x_i v_i \text{ con la restricción } \sum_{i=1}^{n} x_i v_i \leq W$$

Donde $v_i > 0$, $w_i > 0$ y $0 \leq x_i \leq 1$ para $1 \leq i \leq n$.

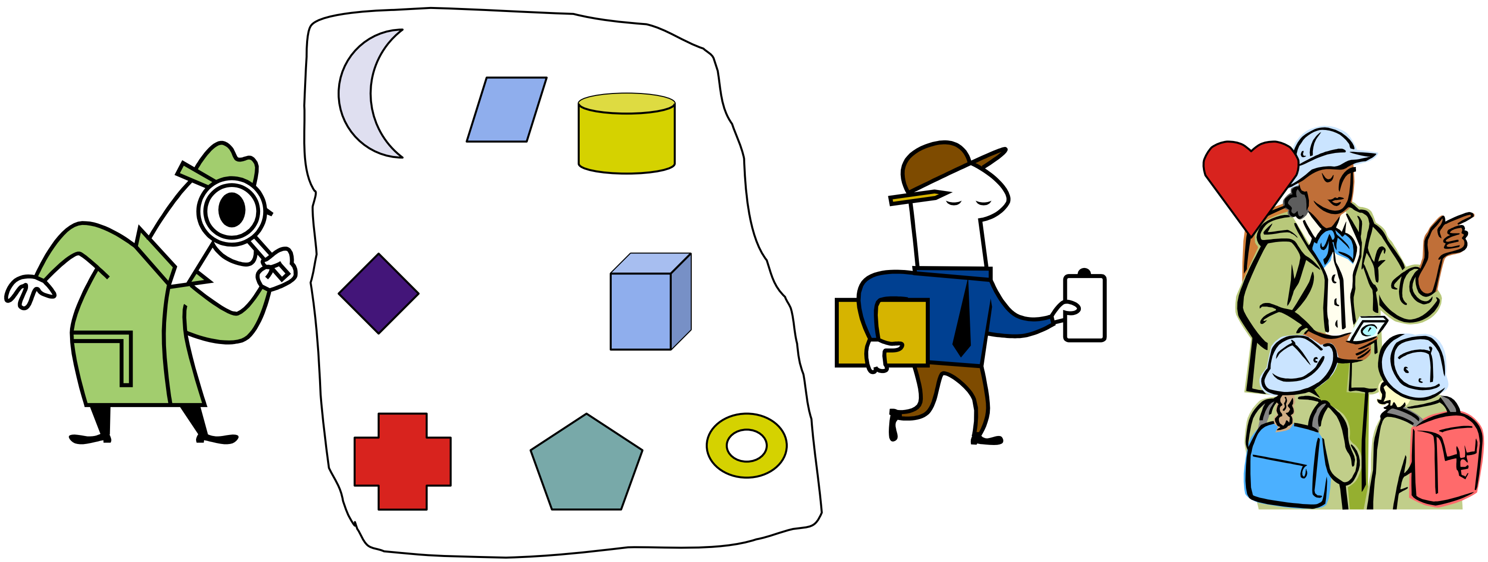


Problema del morral (v1)

- Problema:

$$máx \sum_{i=1}^{n} x_i v_i \text{ con la restricción } \sum_{i=1}^{n} x_i v_i \leq W$$

Donde $v_i > 0$, $w_i > 0$ y $0 \leq x_i \leq 1$ para $1 \leq i \leq n$.

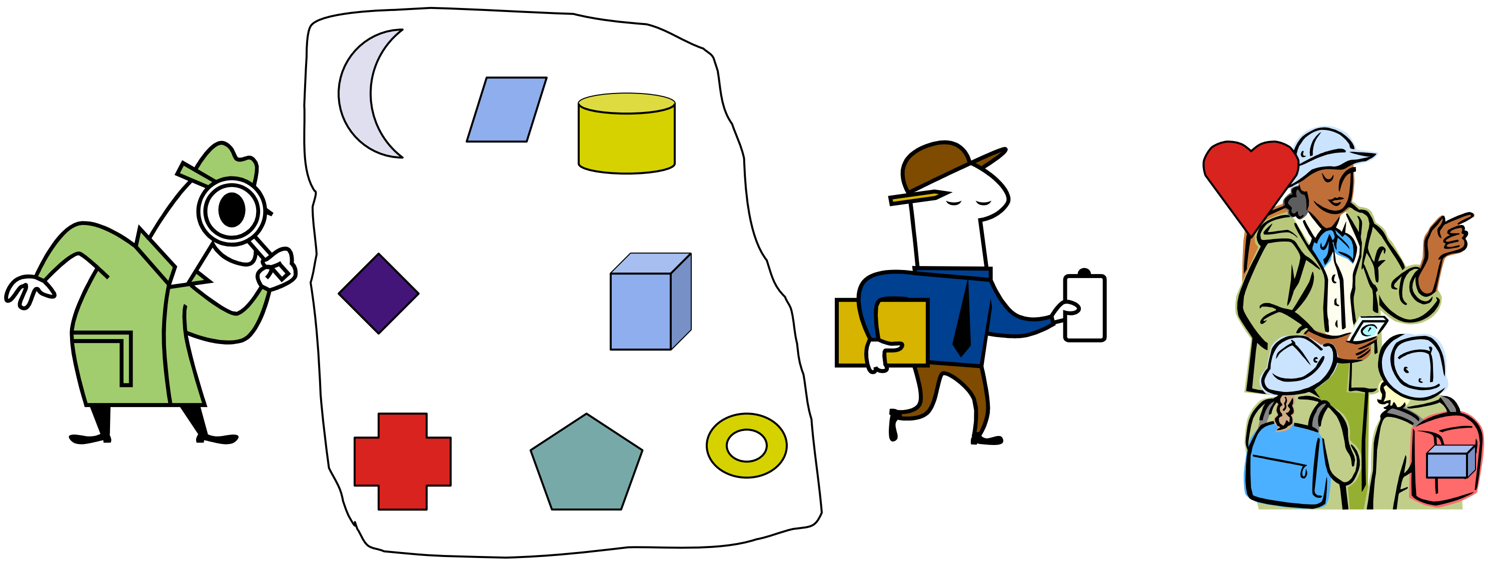


Problema del morral (v1)

- Problema:

$$\text{máx} \sum_{i=1}^{n} x_i v_i \text{ con la restricción } \sum_{i=1}^{n} x_i v_i \leq W$$

Donde $v_i > 0$, $w_i > 0$ y $0 \leq x_i \leq 1$ para $1 \leq i \leq n$.

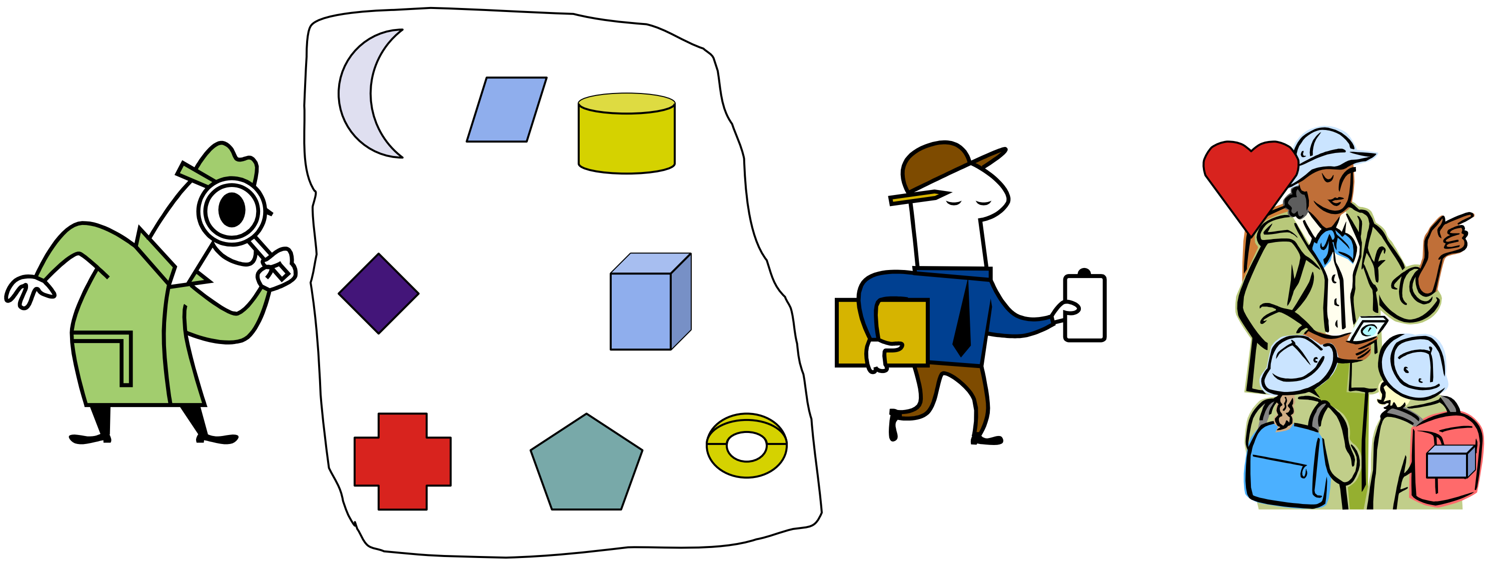


Problema del morral (v1)

- Problema:

$$m\acute{a}x \sum_{i=1}^{n} x_i v_i \text{ con la restricción } \sum_{i=1}^{n} x_i v_i \leq W$$

Donde $v_i > 0$, $w_i > 0$ y $0 \leq x_i \leq 1$ para $1 \leq i \leq n$.

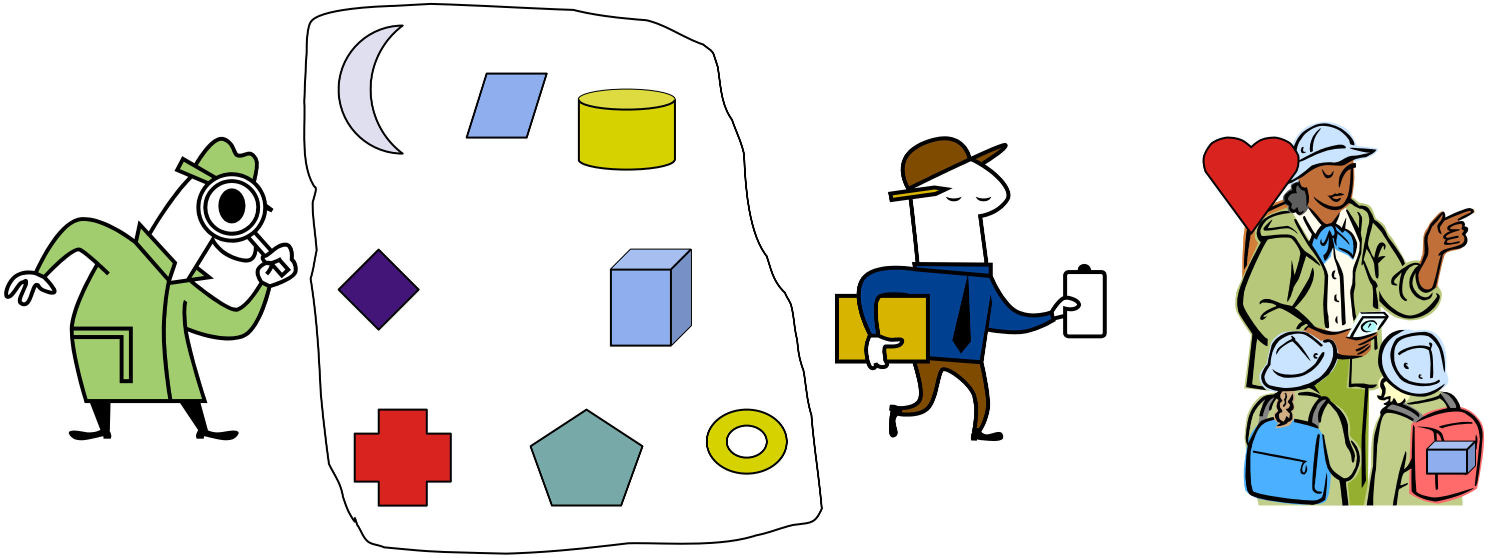


Problema del morral (v1)

- Problema:

$$m\acute{a}x \sum_{i=1}^{n} x_i v_i \text{ con la restricción } \sum_{i=1}^{n} x_i v_i \leq W$$

Donde $v_i > 0$, $w_i > 0$ y $0 \leq x_i \leq 1$ para $1 \leq i \leq n$.

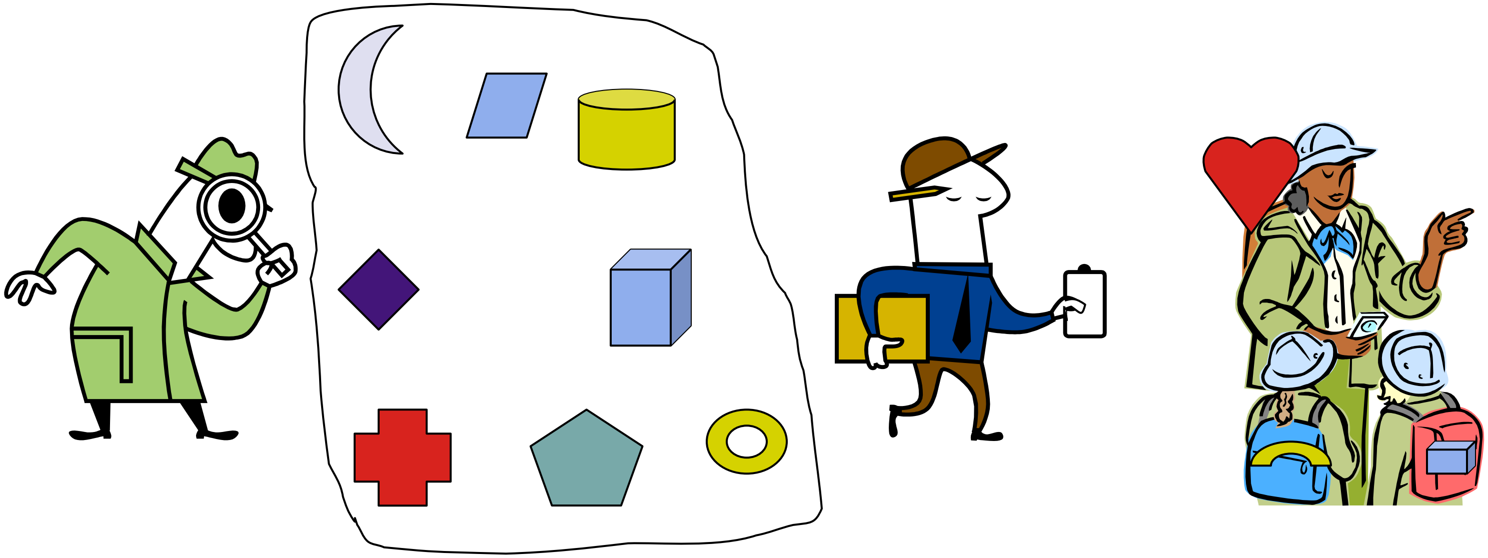


Problema del morral (v1)

- Problema:

$$\text{máx} \sum_{i=1}^{n} x_i v_i \text{ con la restricción } \sum_{i=1}^{n} x_i v_i \leq W$$

Donde $v_i > 0$, $w_i > 0$ y $0 \leq x_i \leq 1$ para $1 \leq i \leq n$.



Algoritmo (v1)

- Algoritmo genérico

morral1(Arreglo[1..n]de Entero+: w, Entero+: W, Arreglo[1..n]de Entero+: v):Arreglo[1..n]de Real

1 [x[i] = 0] i = 1, n

2 peso=0

3 (peso < W)[i = el mejor objeto restante

Si (peso + w[i] ≤ W) entonces

x[i], peso = 1, peso + w[i]

sino

x[i], peso = (W – peso)/w[i], W

fsi]

4 Regrese x

Funciones de selección:

1. **El objeto más valioso, para incrementar el valor de la carga**
2. **El objeto más pequeño, para llenarlo lentamente**
3. **El objeto cuyo valor por unidad de peso sea el mayor**

Prueba del algoritmo (v1)

n = 5			W = 100		
w	10	20	30	40	50
v	20	30	66	40	60
v/w	2.0	1.5	2.2	1.0	1.2

Selecció	x[i]					valor
Máx v[i]	0	0	1	0.5	1	146
Mín w[i]	1	1	1	1	0	156
Máx v[i]/ w[i]	1	1	1	0	0.8	164

- Teorema: si se seleccionan los objetos en orden decreciente de vi/wi, el algoritmo del morral (v1) encuentra una solución óptima

Demostración

- Suponga que los objetos están ordenados por vi/wi

  v1/w1 ≥ v2/w2 ≥ … ≥ vn/wn

  - Sea X=(x1, x2, …, xn) la solución hallada por el algoritmo
  - Si todos los xi son 1, entonces esta solución es la óptima
  - Si no, suponga que j denota el menor índice tal que xj < 1. Según el algoritmo, xi = 1 si i < j, xi = 0 si i > j, en cualquier otro caso $W = \sum_{i=1}^{n} x_i w_i$

    Sea $V(x) = \sum_{i=1}^{n} x_i w_i$ el valor de la solución X

  - Sea Y=(y1, y2, …, yn) cualquier solución factible.
  - Como Y es factible $\sum_{i=1}^{n} y_i w_i \leq W$ y por tanto $\sum_{i=1}^{n} (x_i - y_i) v_i \geq 0$

Demostración

Sea $V(y) = \sum_{i=1}^{n} y_i v_i$ el valor de la solución Y

$$V(x) - V(y) = \sum_{i=1}^{n} (x_i - y_i) v_i = \sum_{i=1}^{n} (x_i - y_i) w_i \frac{v_i}{w_i}$$

- Cuando i<j, xi=1 y (xi–yi)≥0, mientras que vi/wi≥vj/wj cuando i>j, xi=0 y (xi–yi)≤0, mientras que vi/wi≤vj/wj, y cuando i=j, vi/wi=vj/wj. Por lo tanto, en todos los casos se tiene que (xi–yi)(vi/wi) ≥(xi–yi)(vj/wj)

$$V(x) - V(y) \geq (v_j / w_j) \sum_{i=1}^{n} (x_i - y_i) w_i \geq 0$$

- Con lo cual se demuestra que ninguna solución factible puede tener un valor mayor que V(x) y ella es óptima

UNIVERSIDAD DE LOS ANDES

Análisis

- Si los objetos están ordenados por orden decreciente de vi/wi, entonces T(n) está en O (n)
- T(n) total está en O(n lg n)
- Si se usa un montículo con el mayor valor de vi/wi en la raíz, entonces T(n) está en O(lg n)

Algoritmo de Bellman-Ford

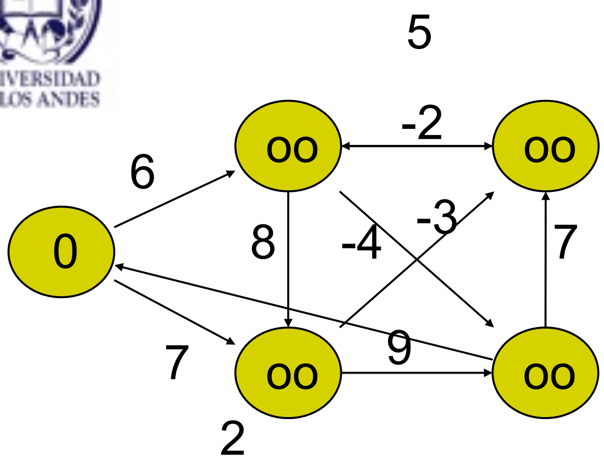
- Resuelve el problema de los caminos más cortos en el caso general donde los pesos pueden ser negativos
- Usa programación dinámica.
- Regresa un valor lógico para indicar:
  - Verdadero, no hay ciclos con peso negativo alcanzables desde **s** y produce los caminos mínimos con sus pesos
  - Falso, si los hay, por lo cual no hay solución.

Algoritmo de Bellman-Ford

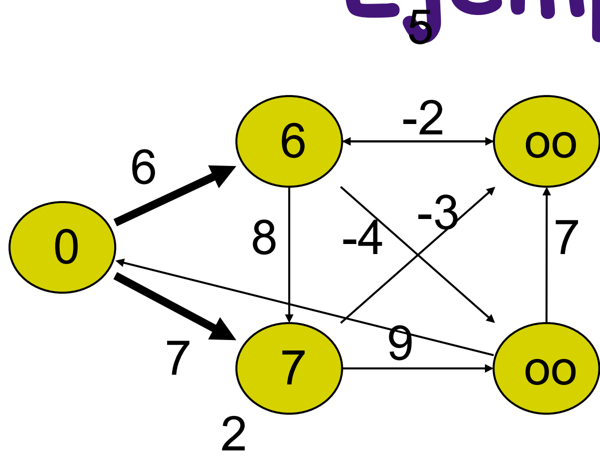
26/11/98 C+CBellmanFord(Nodo: s): Lógico {pre: $n > 0 \wedge s \in N$ } {pos: $n > 0$ }		
1 2 3 4	iniciar(s) O(N) [[relajar(u, v, w)] $(u, v) \in A$] $i = 1, N - 1$ $\Theta(A)$ [Si (v.D() > u.D() + w(u, v)) entonces regresar Falso fsi] $(u, v) \in A$ O(A) regrese Verdadero	**-iniciar(), relajar():** Operaciones de la clase Grafo. **-i:** Entero. Subíndice. **-D(), Padre().** Operaciones de la clase Nodo.

$$T(n) = O(N\,A)$$

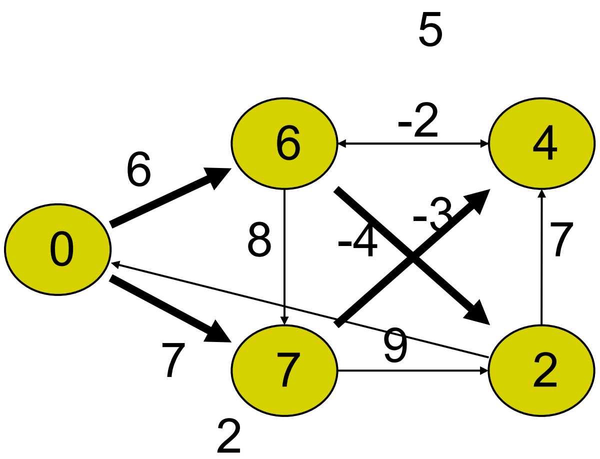
Ejemplo



Inicio

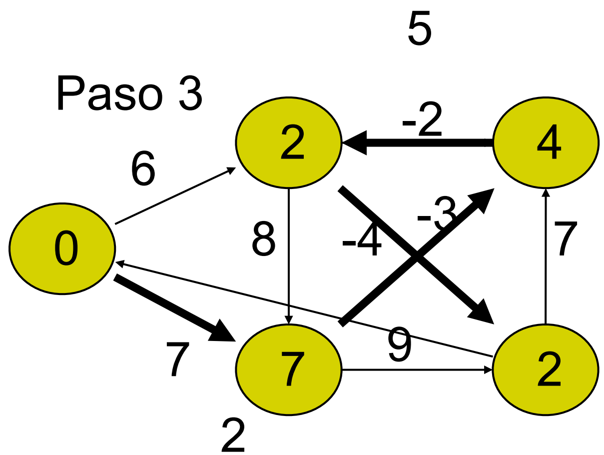


Paso 1

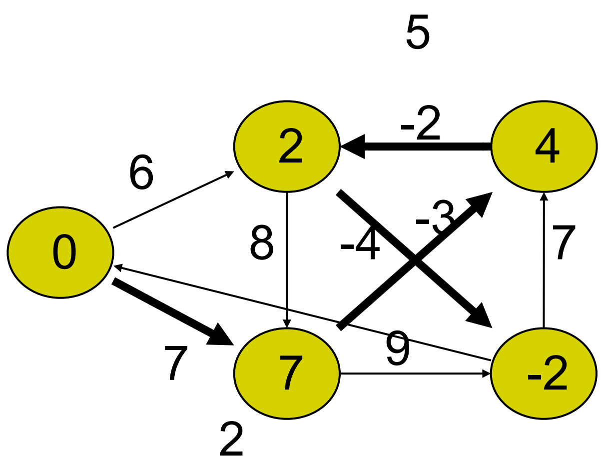


Paso 2

Paso 3



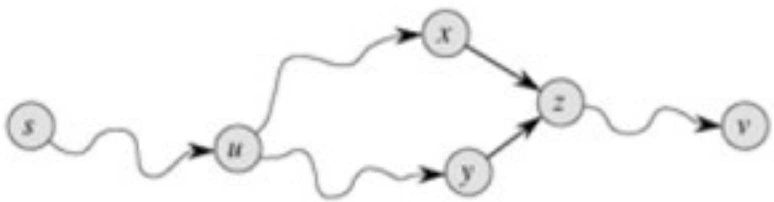
Ejemplo



Fin y regresa Verdadero

- Lema 16: Sea G=(N,A) un digrafo etiquetado y asuma que G contiene ciclos no negativos alcanzables desde s. Luego de |N|−1 iteraciones en el lazo 2 del algoritmo C+CBellmanFord, se tiene d[v] = δ(s, v) ∀ v alcanzable desde s

Prueba del lema 16: Usando la propiedad de la relajación.



Corrección del algoritmo de Bellman-Ford

- **Corolario 16**: Para cada $v \in N$ hay un camino de **s** a **v** si y solo si, el algoritmo de Bellman-Ford termina con $d(v) < \infty$ cuando se ejecuta en G.
- **Teorema 16**: Sea G=(N,A) un digrafo etiquetado con w:A->R, con origen s.
  - Si G no tiene ciclos negativos alcanzables desde s, el algoritmo C+CBellmanFord regresa Verdadero y se tiene $d[v]=\delta(s, v)\ \forall\ v \in N$, con el árbol de los caminos mínimos desde **s**.
  - Si G contiene un ciclo negativo alcanzable desde s, entonces regresa Falso.

Demostración del teorema 16

- Suponga que G no contiene ciclos negativos alcanzables desde s.
  - Al finalizar el algoritmo $d[v] = \delta(s, v)\ \forall\ v \in N$.
    - Si el nodo v es alcanzables desde s, entonces el lema 16 lo prueba.
    - Si v no es alcanzable desde s, entonces no hay camino desde s a v.
    - Ninguna de las condiciones del paso 3 hace que el algoritmo regrese Falso, por lo tanto regresa Verdadero
    - $D[v] = \delta(s, v) \leq \delta(s, u) + w(u, v) = d[u] + w(u, v)$

Demostración (continuación)

- Suponga que G contiene un ciclo negativo alcanzable desde s, sea ese ciclo c=<n_0, n_1, … n_k>, donde $n_0=n_k$, entonces

(1) $$\sum_{i=1}^{k} w(n_{i-1}, n_i) < 0$$

- Asuma por contradicción que el algoritmo regresa Falso.
- Así, d[ni]≤d[ni−1]+w(ni−1, ni) para i=1, 2, …, k

$$\sum_{i=1}^{k} d[n_i] \le \sum_{i=1}^{k} (d[n_{i-1}] + w(n_{i-1}, n_i)) = \sum_{i=1}^{k} d[n_{i-1}] + \sum_{i=1}^{k} w(n_{i-1}, n_i)$$

- Como $n_0=n_k$, cada nodo aparece exactamente una vez en cada suma y

$$\sum_{i=1}^{k} d[n_i] = \sum_{i=1}^{k} d[n_{i-1}] \text{ y por el corolario 16 } d[n_i] \text{ es finito y}$$

$$0 \le \sum_{i=1}^{k} w(n_{i-1}, n_i)$$

Que contradice (1). q.e.d

Camino más corto desde un nodo fuente en un dag etiquetado

- El C+C está siempre bien definido por ser un dag, aún si tiene pesos negativos.
- Se relajan los nodos en base a un ordenamiento topológico de los mismos, calculando el C+C desde **s** en $\Theta(N + A)$.
- Si hay un camino desde **u** hasta **v** en G, entonces **u** precede a **v** en el orden topológico.

Algoritmo para caminos mínimos en dag etiquetado

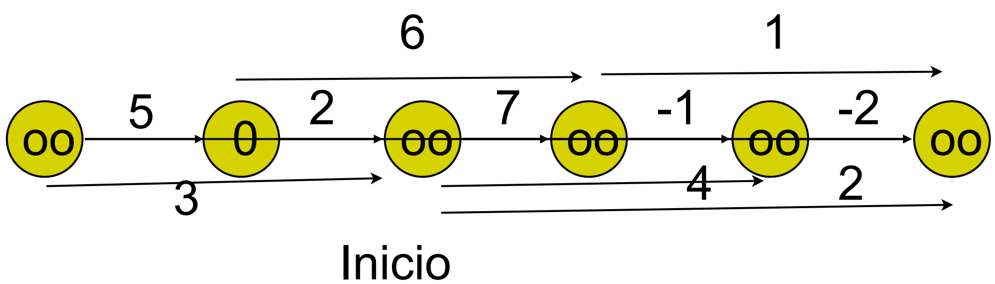
26/11/98

C+Cdag(Nodo: s)

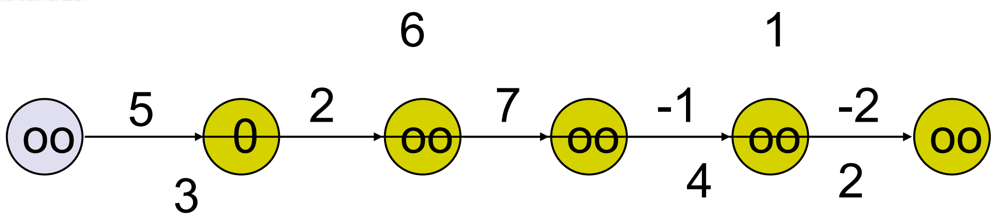
{pre: $n > 0 \wedge s \in N$ } {pos: $n > 0$ }

Instrucción	Orden	Descripción
lot = ordenTopologico( )	$\Theta(N+A)$	**-iniciar(), relajar(), ordenTopologico():** Operaciones de la clase Grafo.
iniciar(s)	$\Theta(N)$	**-lot:** ListaDe [Entero]. Lista de nodos en orden topológico.
[[relajar(u, v, w)] $v \in$ listaAdy(u)] $u \in$ lot		

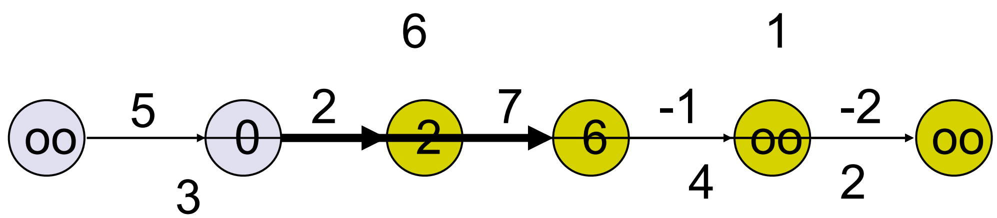
$T(n) = \Theta(N + A)$.



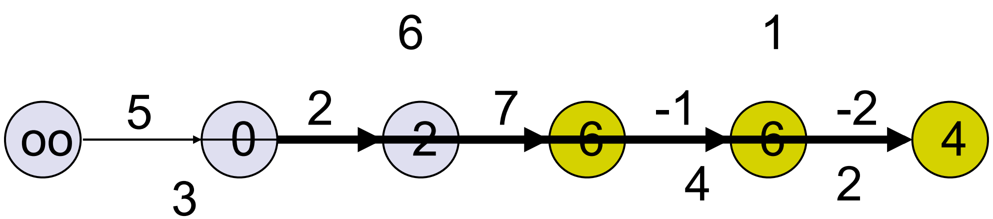
Ejemplo



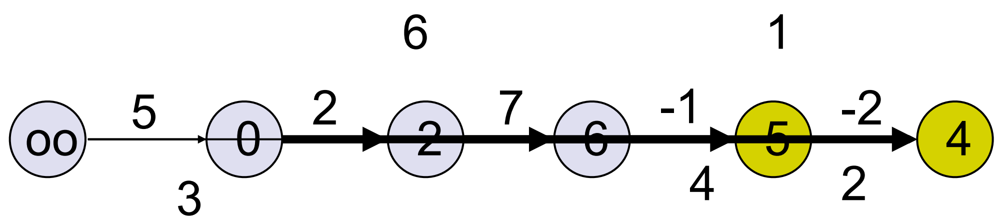
Paso 1



Paso 2

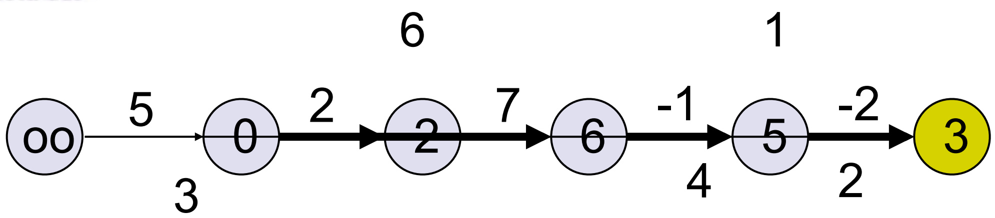


Paso 3

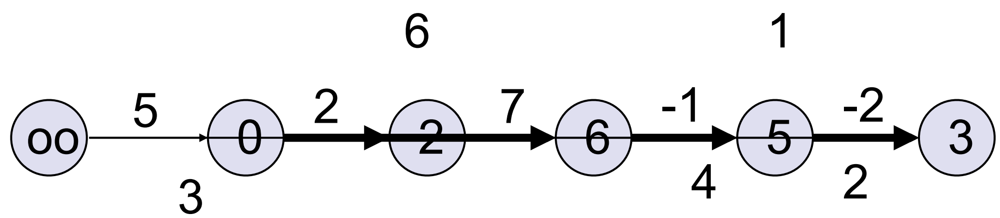


Paso 4

Ejemplo



Paso 5



Paso final

Los nodos se ordenan topológicamente de izquierda a derecha, desde el nodo fuente s.
Los valores de d se muestran dentro de los nodos y los arcos resaltados forman el subgrafo predecesor almacenado en el vector padre

Corrección del algoritmo C +Cdag

- **Teorema**: Si a un dag etiquetado $G = (N, A)$ con un nodo fuente **s** y sin ciclos se le ejecuta el procedimiento C+Cdag(s) entonces $d(v) = \delta(s, v)\ \forall v \in N$ y $G\pi$ es el árbol del C+C cuya raíz es **s**.
- **Aplicación:** PERT donde un camino crítico es el camino más largo de G. Se usa el mismo algoritmo pero:
  - negando los pesos de G, o
  - reemplazando ∞ con $-\infty$ en iniciar(s) y ">" con "<" en relajar(u, v, w)