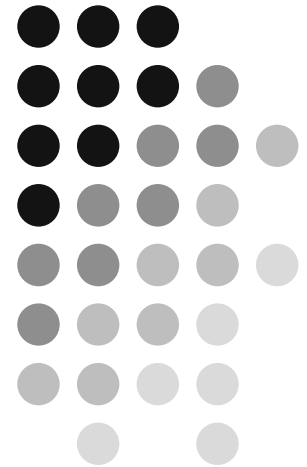


Grafos: Caminos más cortos

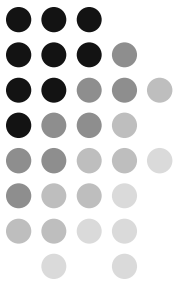


UNIVERSIDAD
DE LOS ANDES

Diseño y Análisis de Algoritmos
Cátedra de Programación
Carrera de Ingeniería de Sistemas
Prof. Isabel Besembel Carrera

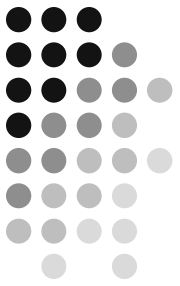


Caminos mínimos



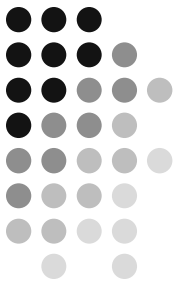
- Sea un digrafo etiquetado $DG = \{N, A\}$ cuyas aristas tienen pesos no negativos
- Problema: encontrar todos los caminos mínimos desde un nodo origen hacia todos los demás nodos de DG
- Solución: algoritmo de Dijkstra
- Características:
 - ❖ Propiedad invariante: $N = S \cup C$, donde $S = \{\text{nodos seleccionados con su distancia mínima desde el origen}\}$ y $C = \{\text{resto de los nodos cuya distancia desde el origen es desconocida}\}$
 - ❖ 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 $C = \{2, 3, \dots, n\}$

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

3 [$v = \text{algún elemento de } C \text{ que minimiza } D_v$

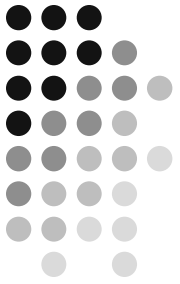
$C = C - \{v\}$

para cada $w \in C$

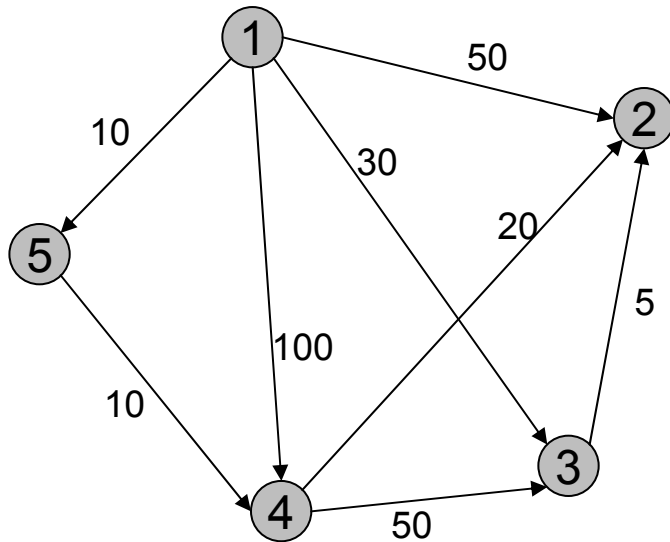
$D_w = \min(D_w, D_v + L_{v,w})$

fpc] $i = 2, n$

4 regrese D



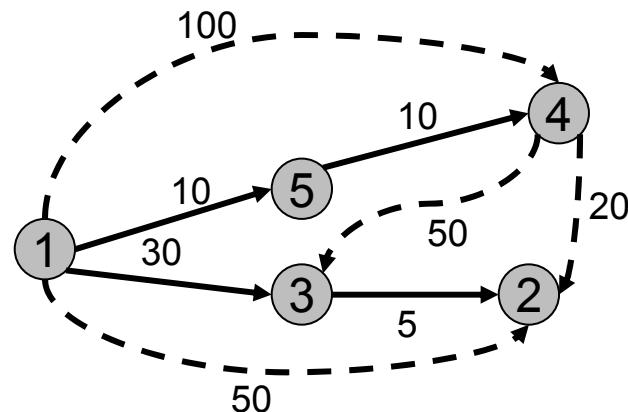
Algoritmo de Dijkstra



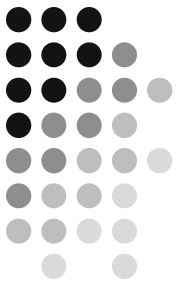
Digrafo etiquetado

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

3 $[v = \text{algún elemento de } C \text{ que minimiza } D_v$

$C = C - \{v\}$

para cada $w \in C$

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

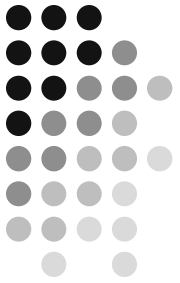
$D_w = D_v + L_{v,w}$

$P_w = v$

fsi

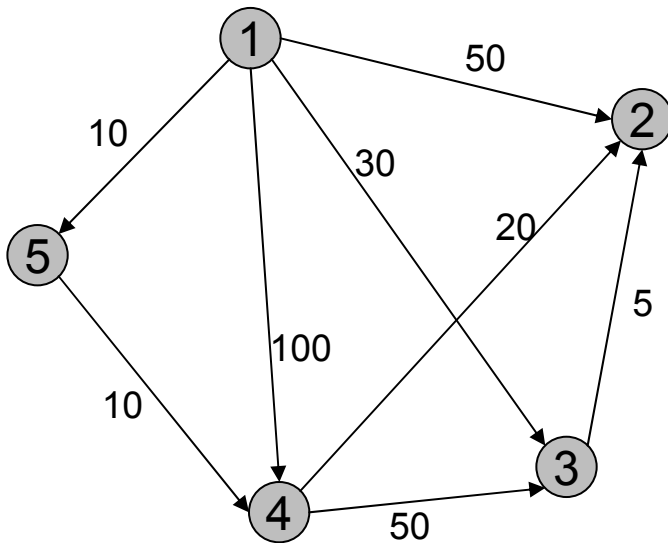
fpc $] \quad i = 2, n$

Algoritmo de Dijkstra

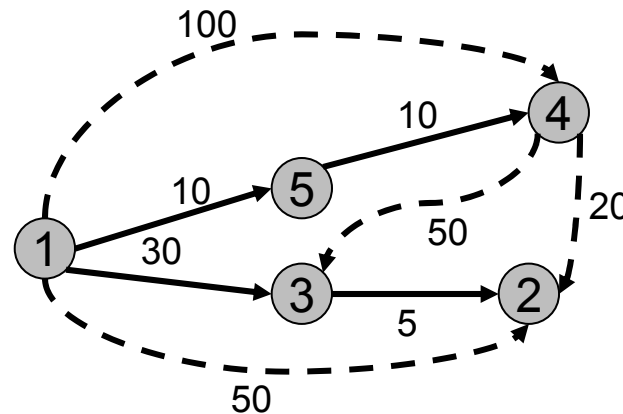


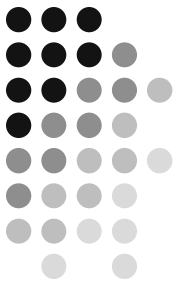
Origen el nodo 1

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





- 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 $i \neq 1$ está en S , entonces D_i da la longitud del camino más corto desde el origen hasta i
 - B. Si un nodo i no está en S , entonces D_i 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.

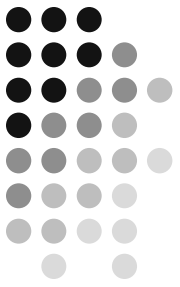
Para el resto de los nodos, el único camino desde el origen es el camino directo, por lo tanto B es cierto también

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

Inductiva:

- Caso A: \forall 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

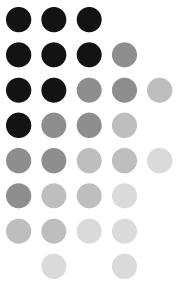
Continuación de la demostración



- 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 $x \notin S$, 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 $\Rightarrow D[x] > D[v]$, 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

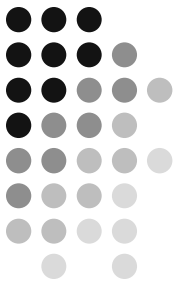


Continuación de la demostración



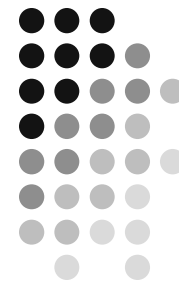
- Por lo cual al añadir v a S , la parte A es válida.
- Caso B: Sea $w \neq v$, $w \notin S$. Cuando v se añade a S , hay dos posibilidades para el camino especial más corto del origen a w ; o bien no cambia, o ahora pasa por v . En este último caso, sea x el último nodo de S visitado antes de llegar 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



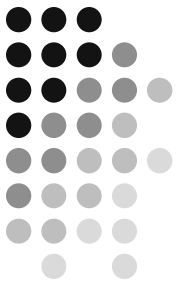
- Si se implementa tal y como aparece $\Theta(n^2)$, preferible en el caso de digrafos densos
- Si $a \ll n^2$, es preferible implementar el digrafo con listas de adyacencia y utilizar un montículo, donde cada nodo tiene el número del nodo del grafo en el conjunto C y su distancia, ordenado por la menor 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



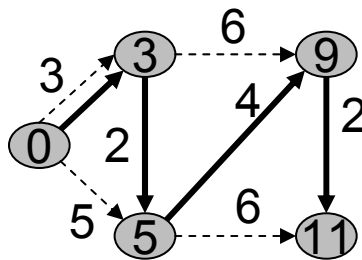
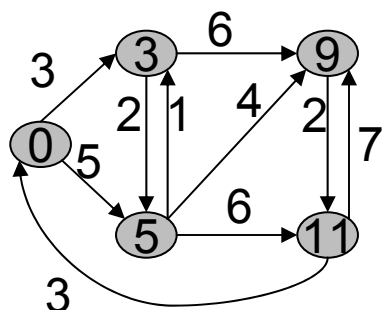
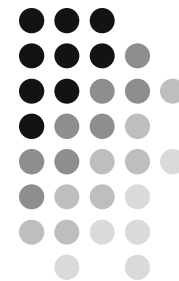
- Dado un digrafo $G = (N, A)$ etiquetado y conexo con pesos $w: A \rightarrow \mathcal{R}$. Para un camino $\mathbf{p} = \langle n_0, n_1, \dots, n_k \rangle$, $w(\mathbf{p}) =$ sumatoria de $i=1$ hasta n de $w(n_{i-1}, n_i)$, se define el camino con el peso mínimo de u a v como: $\delta(u, v) = \min \{ w(\mathbf{p}) : \text{longitud del camino } \mathbf{p} \text{ de } u \text{ a } v \}$, si $\exists \mathbf{p}$ 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 \mathbf{p} dado el destino \mathbf{d} desde cada nodo \mathbf{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



-----> C+C 1

-----> C+C 2

➤ Convenciones: Si $a \neq -\infty \Rightarrow$

$$a + \infty = \infty + a = \infty$$

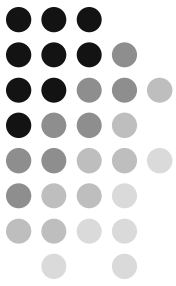
$$a + (-\infty) = (-\infty) + a = -\infty$$

➤ Propiedad de los C+C:

- ❖ Un C+C entre dos nodos contiene otros C+Cs dentro.
- ❖ Sea $G = (N, A)$ un digrafo etiquetado con $w: A \rightarrow \mathfrak{R}$ y $\mathbf{p} = \langle n_0, n_1, \dots, n_k \rangle$, el C+C de n_1 a n_k , para cualquier $i, j / 1 \leq i \leq j \leq k$ $\mathbf{p}_{i:j} = \langle n_i, n_{i+1}, \dots, n_j \rangle$ un subcamino de \mathbf{p} de n_i a $n_j \Rightarrow \mathbf{p}_{i:j}$ es el C+C entre n_i y n_j .
- ❖ Suponga que \mathbf{p} de s a v puede ser descompuesto en \mathbf{p}' de s a $u \rightarrow v$ para algún u y \mathbf{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

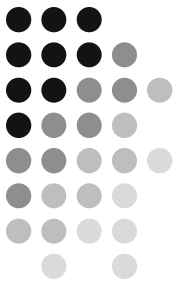


- *Relajación*: $\forall v \in N$, se mantiene un atributo $d(v)$ que es el límite superior de los pesos del C+C de s a v , $d(v)$ se denomina el estimado del peso del C+C y se inicia con

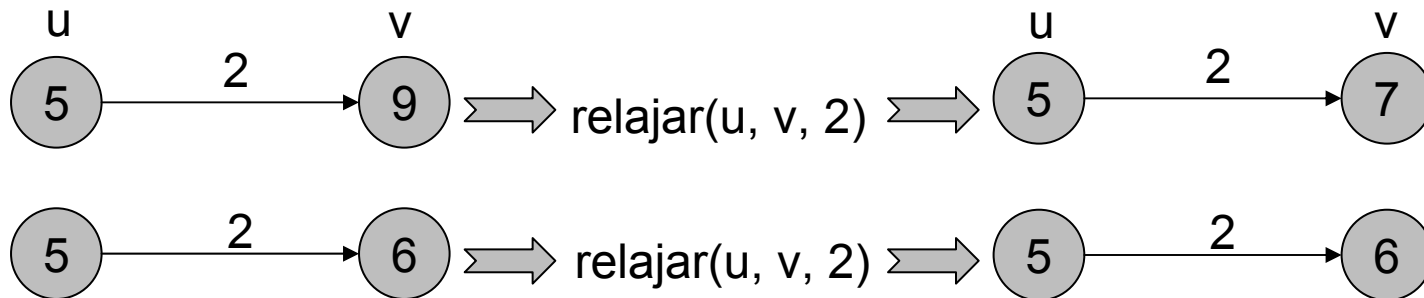
26/11/98		iniciar(Nodo: s)
{pre: $n > 0 \wedge s \in N$ }		{pos: $n > 0$ }
1	[$v.D(\infty)$ $v.Padre(Nulo)$] $v \in N$	-D(), Padre(): Operaciones de la clase Nodo.
2	$s.D(0)$	
3	regrese	

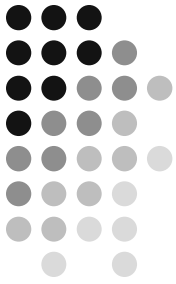
$$T(n) = O(n)$$

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 $\text{padre}(v)$.
- Ejemplo:





26/11/98

relajar(Nodo: u, v, Real: w)

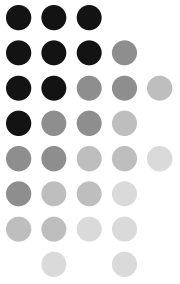
{pre: $n > 0 \wedge (u, v) \in A$ }

{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	

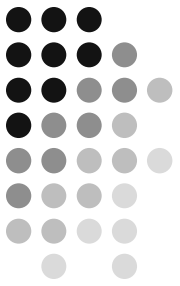
- Propiedades de la relajación: Sea $G=(N, A)$ un digrafo etiquetado con $w: A \rightarrow \mathcal{R}$
 - Sea $(u, v) \in A$ luego de la relajación del arco (u, v) por medio de $relajar()$ se tiene que $d(v) \leq d(u) + w(u, v)$
 - Sea $s \in N$ el nodo fuente y G iniciado con $iniciar(s)$, entonces $d(v) \geq \delta(s, v) \forall v \in N$ 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



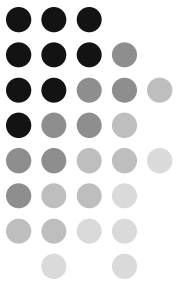
- ❖ Si no hay camino de $s \in N$ a $v \in N$ entonces luego que G es iniciado con $\text{iniciar}(s)$, se tiene $d(v) = \delta(s, v)$ y esta igualdad se mantiene como una invariante sobre cualquier secuencia de relajación sobre los arcos de G .
- ❖ Sea $s \in N$ y el camino de s a $u \rightarrow v$ el C+C en G para $u, v \in N$. Suponga $G.\text{iniciar}(s)$ y luego una secuencia de relajación que incluya $G.\text{relajar}(u, v, w)$ 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 $G.\text{iniciar}(s)$, 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 $G.\text{iniciar}(s)$ 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

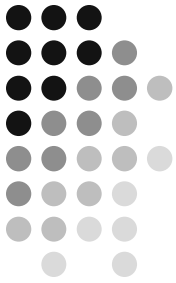


- Todos los $w(u, v) \geq 0 \forall u, v \in N$ y $(u, v) \in A$.
- El algoritmo mantiene un conjunto S de nodos cuyos pesos finales del C+C desde s han sido determinados.
- Esto es, $\forall v \in S$ se tiene $d(v) = \delta(s, v)$.
- El algoritmo repetidamente selecciona un nodo $u \in N - S$ con el 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:** G_π es el árbol del C+C cuya raíz es s .



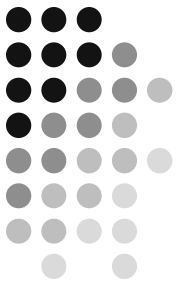
26/11/98

C+Cdijkstra(Nodo: s): Conjunto

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

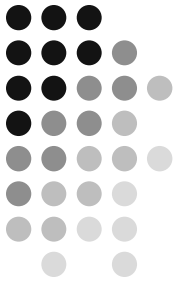
{pos: $n > 0$ }

<p>1 iniciar(s) 2 [C.entrar(u)] $u \in N$ 3 (\negC.vacíaColaP()) [u = C.min() C.salir() $S = S \cup \{ u \}$ [relajar(u, v, w)] $v \in \text{listaAdy}(u)$ 4 regrese S</p>	<p>-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.</p>
---	--



- 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: $g(i, j) \geq 0 \quad \forall i, j$ }

{pos: $g(i, j) \geq 0 \quad \forall i, j$ }

```

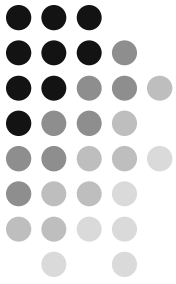
1  ele = t.conTabla(n)
2  Si (ele.Clave() = n) entonces
3  [cam( i ).dis(maximo)
   cam( i ).padre({TipoNoDef}) ] i = 1, 100
   j = ele.Resto()
   cam[j].dis(0)
   c.enCola(j)
   ( -c.vaciaCola() )
   [ i = c.primer()
   c.desenCola()
   Si ( g ( i, j ) = 1 ) ^ ( cam(j).distancia() = maximo) entonces
       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: $n > 0 \wedge G' = G \wedge \text{padre} \supset \text{árbol en amplitud de s}$ }

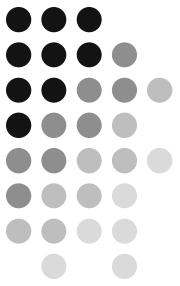
```

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.primer()
    [ 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.

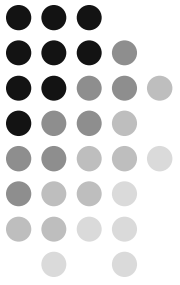
$$T(n) = O(N + A)$$

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.

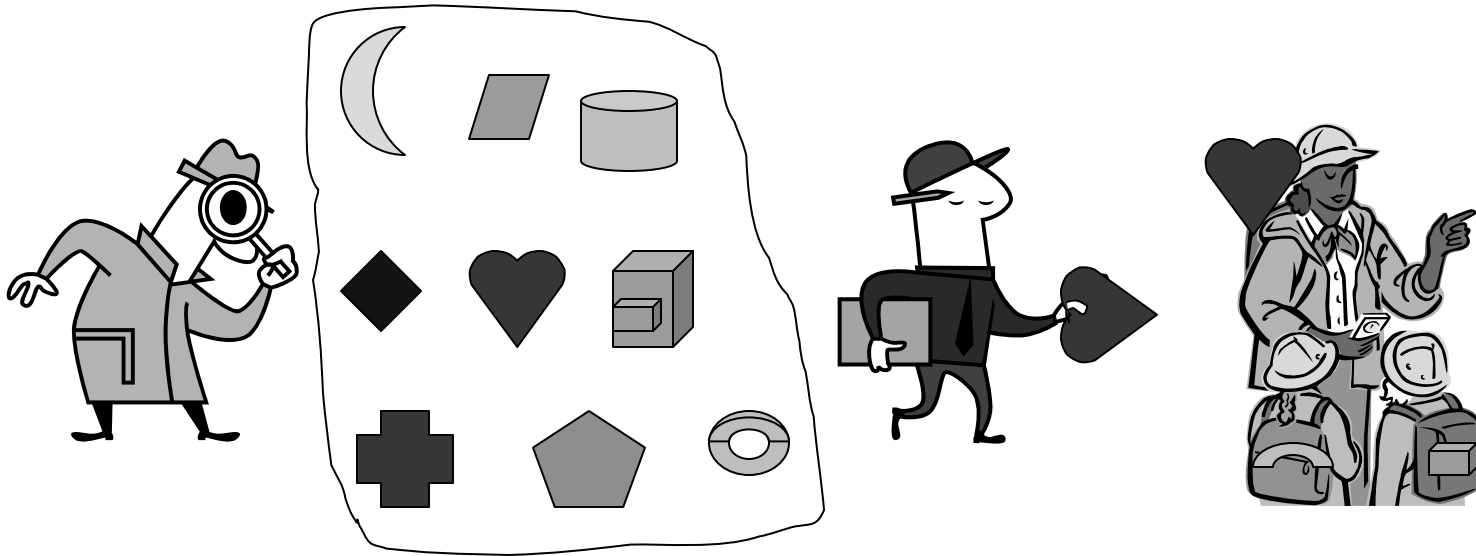
Problema del morral (v1)



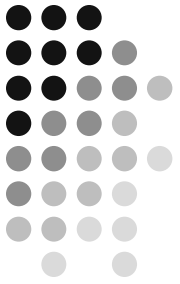
➤ Problema:

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

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

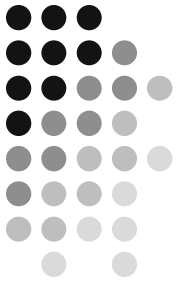
fsi]

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)

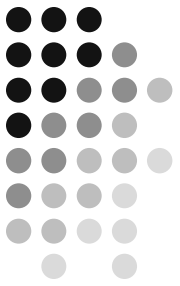


	n = 5		W = 100		
w	10	20	30	40	50
v	20	30	66	40	60
v/w	2.0	1.5	2.2	1.0	1.2

Selección	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 v_i/w_i , el algoritmo del morral (v1) encuentra una solución óptima

Demostración



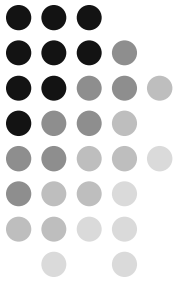
- Suponga que los objetos están ordenados por v_i/w_i
- $$v_1/w_1 \geq v_2/w_2 \geq \dots \geq v_n/w_n$$
- ❖ Sea $X=(x_1, x_2, \dots, x_n)$ la solución hallada por el algoritmo
 - ❖ Si todos los x_i son 1, entonces esta solución es la óptima
 - ❖ Si no, suponga que j denota el menor índice tal que $x_j < 1$. Según el algoritmo, $x_i = 1$ si $i < j$, $x_i = 0$ 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 $Y=(y_1, y_2, \dots, y_n)$ 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



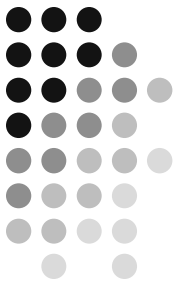
Sea $V(y) = \sum_{i=1}^n y_i v_i$ el valor de la solución Y

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

- Cuando $i < j$, $x_i = 1$ y $(x_i - y_i) \geq 0$, mientras que $v_i / w_i \geq v_j / w_j$ cuando $i > j$, $x_i = 0$ y $(x_i - y_i) \leq 0$, mientras que $v_i / w_i \leq v_j / w_j$, y cuando $i = j$, $v_i / w_i = v_j / w_j$. Por lo tanto, en todos los casos se tiene que $(x_i - y_i)(v_i / w_i) \geq (x_i - y_i)(v_j / w_j)$

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

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



- Si los objetos están ordenados por orden decreciente de v_i/w_i , 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 v_i/w_i en la raíz, entonces $T(n)$ está en $O(\lg n)$