.UBAfiuba
FACULTAD DE INGENIERÍA

Greedy: Seam carving y camino mínimo

Teoría de Algoritmos I (75.29 / 95.06)

Ing. Víctor Daniel Podberezski
vpodberezski@fi.uba.ar

Adecuación inteligente de imágenes

Los contenidos web modernos

están preparados para verse en diferentes dispositivos.

Cada dispositivo tiene tamaños y escalas de pantallas diferentes

el contenido “responsive” se acomoda a la relación de aspecto y dimensiones

Una misma imagen debe verse bien en cada situación

Se realiza redimensión y recorte para ajustar al contenedor

A veces esto no arroja resultados satisfactorios.

Resultados insatisfactorios

Imagen original

Escalado

Recorte

Seam Carving

Algoritmo para la manipulación de la imagen

Creado por Shai Avidan y Ariel Shamir en 2007

Paper “Seam Carving for content-aware image Resizing”

Enlace: *https://dl.acm.org/doi/pdf/10.1145/1275808.1276390*

Analiza la imagen recortando los pixels de menor importancia

Encuentra una secuencia de pixel horizontal o vertical

Al retirarla tiene el menor impacto sobre la visualizacion

Seam: veta / Carving: tallado

Retira tantas vetas como sea necesario para llegar al tamaño requerido

Ejemplo

Imagen original

Escalado

Recorte

Seam carving

Pixels poco importantes

Existen diferentes métodos

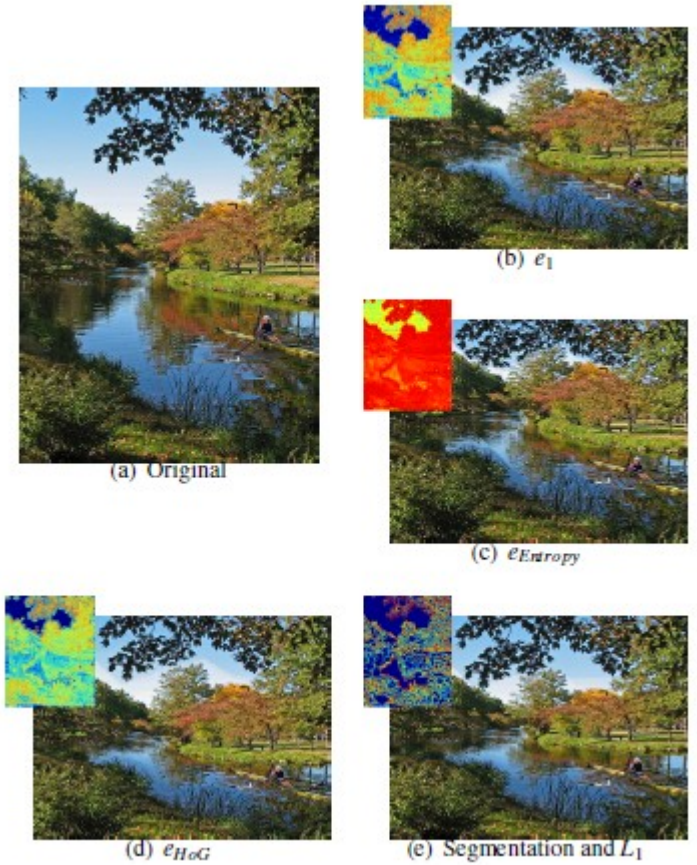
Por ejemplo: entropía, diferencia cromática, movimiento de la mirada, etc

Diferencia cromática

Pixel rodeados de pixels similares → poca importancia (ej: cielo azul)

Pixel rodeado de pixels diferentes → muchas importancia (ej: cara de una persona)

Inicialmente debo calcular la importancia de cada pixel



Vetas

Pueden ser

- Horizontales
- Verticales

Buscaremos una veta

- Que inicie en extremo superior (izquierdo)
- Finalice en el extremo inferior (derecho)
- Cuya suma de importancia sea la mejor posible

Camino mínimo

Trataremos la imagen

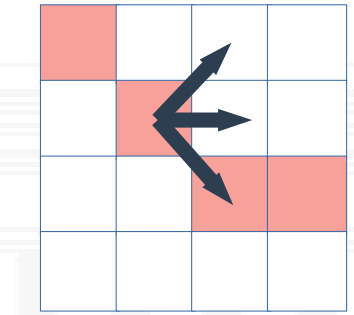
Como una grilla de pixels inter comunicados

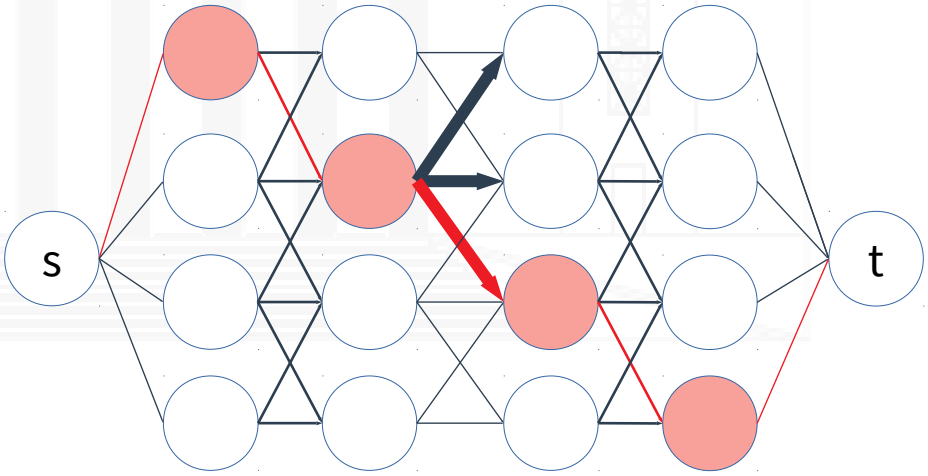
Lo representaremos como un grafo

Los pixels son nodos

Los ejes son los posibles caminos de la veta

Calcularemos el camino mínimo entre s y t





Dijkstra

Propuesto por Edsger Dijkstra en 1959

Es Greedy

Funciona para un grafo G

Dirigido y ponderado (con costos positivos)

de N nodos no aislados (en nuestro caso es la cantidad de pixels) y M ejes

Dados dos nodos

“s” inicial

“t” final

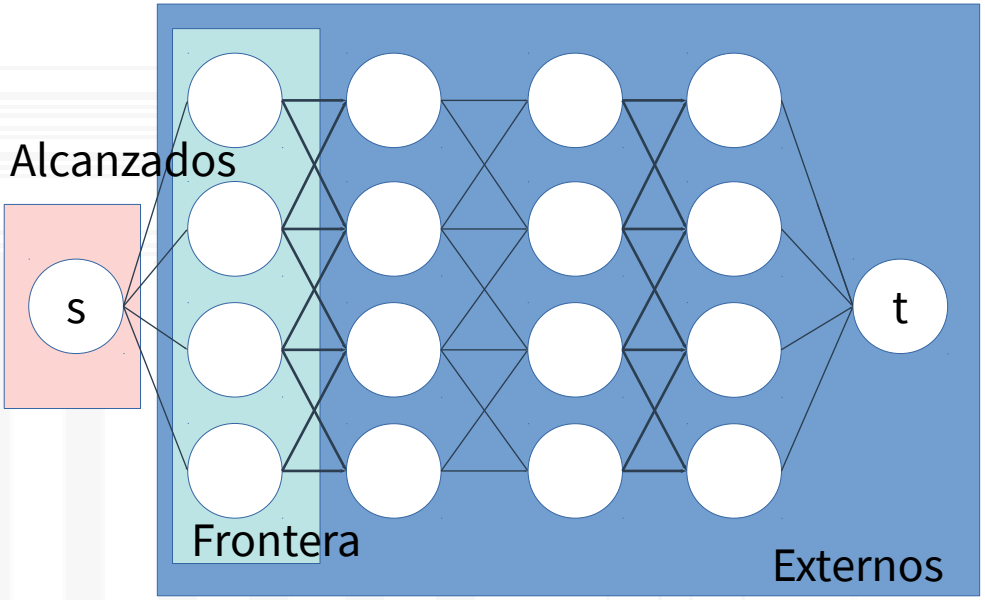
Encuentra el camino mínimo que los une

Funcionamiento

Es un algoritmo iterativo

Divide los nodos en 3 conjuntos

- “alcanzados”: inicialmente solo el nodo “s”
- “externos”: inicialmente todos los nodos menos “s”.
- “frontera”: pertenecientes a externos que están “conectados” a algún nodo de los "alcanzados”



Funcionamiento (cont.)

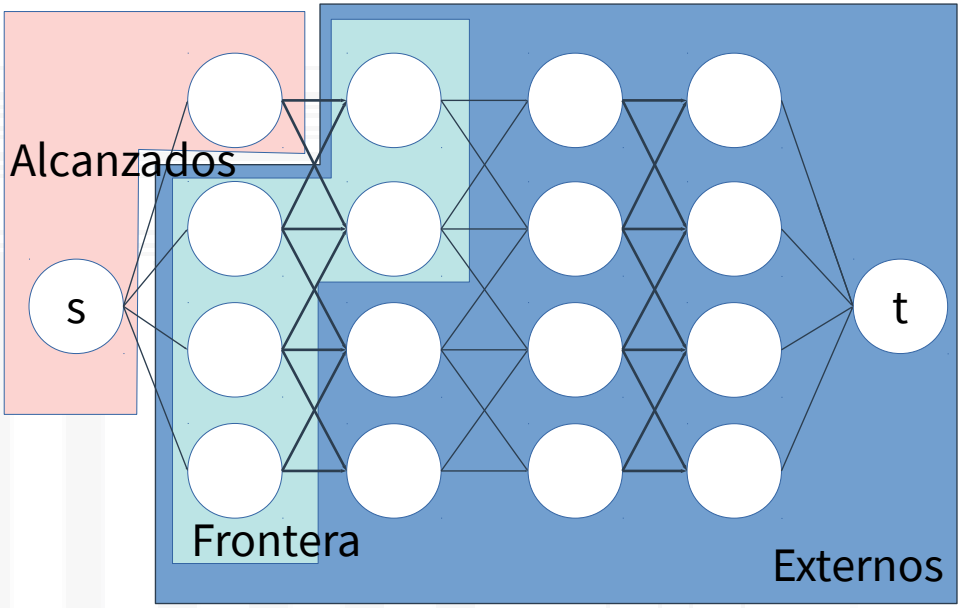
En cada iteración

Se obtiene aquel nodo x de la frontera cuyo costo de llegada desde “s” sea el menor posible (elección greedy)

Se agrega x con su costo al conjunto de “alcanzados” (se registra desde que nodo se llegó)

Se pasan los nodos”externos” con conexión a x a la “frontera” con su cálculo de costo de llegada

Se actualizan los costos de los nodos de la frontera con conexión a x si el nuevo costo es menor.



Funcionamiento (cont.)

Finalización

El algoritmo finaliza cuando no quedan nodos en la frontera

(o cuando se llega al nodo “t”)

Costos de llegada al nodo

Cuando un nodo se encuentra en los alcanzados su costo no puede modificarse y es el mínimo.

Cuando se encuentra en la frontera, es el mínimo costo entre todos los costos de sus nodos vecinos alcanzados mas el costo de llegar a él desde ellos

Cuando es externo su costo es infinito

Pseudocódigo

```
Alcanzados = {(s,0,_)}
frontera = ∅
Por cada nodo x vecino a S
    Agregar a frontera x con costo Cx=Csx y predecesor(x)=s

Mientras la frontera ≠ ∅
    Sea x nodo en frontera con menor costo
    Quitar x de frontera
    Alcanzados = Alcanzados ∪ {(x,Cx,predecesor(x))}
    Por cada y vecino de x
        cy' = Cx + Cxy
        Si y ∈ frontera y cy' < cy
            predecesor(y) = x
            actualizar en frontera y con costo cy'
        Si y ∉ frontera
            predecesor(y) = x
            agregar en frontera y con costo cy'
```

Implementación

Frontera

Heap de mínimos con actualización de clave

Alcanzados

Vector de tamaño “n”

Vecinos de cada nodo

Lista de adyacencias por nodo.

```
Alcanzados = {(s,0,_)}
frontera = ∅
Por cada nodo x vecino a S
     Agregar a frontera x con costo Cx=Csx
                              y predecesor(x)=s

Mientras la frontera ≠ ∅
     Sea x nodo en frontera con menor
                                    costo
     Quitar x de frontera
     Alcanzados = Alcanzados ∪
                  {(x,Cx,predecesor(x))}
     Por cada y vecino de x
          cy' = Cx + Cxy
          Si y ∈ frontera y cy' < cy
               predecesor(y) = x
               actualizar en frontera y con
                                        costo cy'
          Si y ∉ frontera
               predecesor(y) = x
               agregar en frontera y con
                                    costo cy'
```

Complejidad algorítmica

El loop se ejecuta n-1 veces

Cada vez obtengo el nodo con menor en O(logn) ← extract_min

En la frontera se inserta n-1 veces

con costo O(logn) ← insert

En la frontera se actualiza (en el peor de los casos) m veces

con costo O(logn) ← dec_key

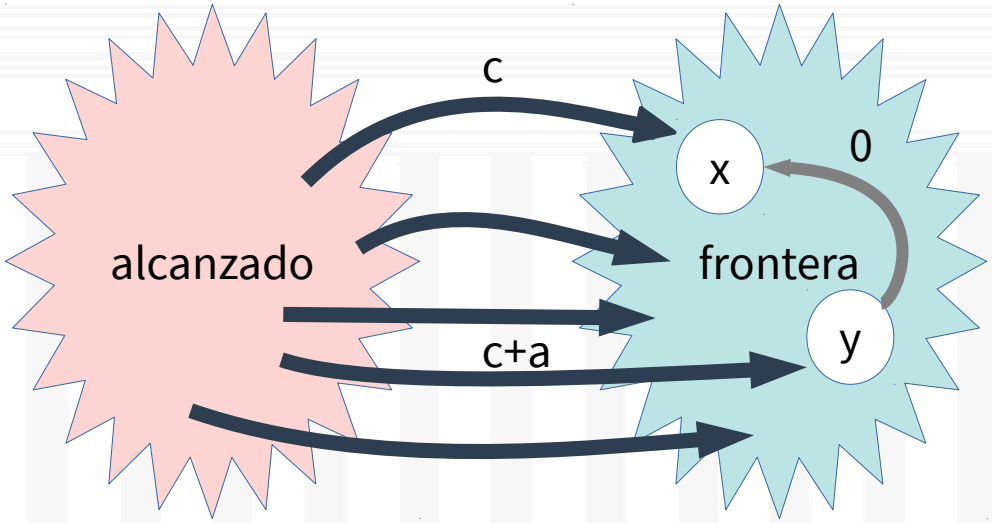
La complejidad es O([n+m]logn)

```
Alcanzados = {(s,0,_)}
frontera = ∅
Por cada nodo x vecino a S
     Agregar a frontera x con costo Cx=Csx
                              y predecesor(x)=s

Mientras la frontera ≠ ∅
     Sea x nodo en frontera con menor
                                      costo

     Quitar x de frontera
     Alcanzados = Alcanzados ∪
                  {(x,Cx,predecesor(x))}
     Por cada y vecino de x
          cy' = Cx + Cxy
          Si y ∈ frontera y cy' < cy
               predecesor(y) = x
               actualizar en frontera y con
                                      costo cy'
          Si y ∉ frontera
               predecesor(y) = x
               agregar en frontera y con
                                  costo cy'
```

Elección Greedy

En cada iteración selecciona el menor costo disponible

Puede esto resultar contraproducente?



En el “mejor de los casos” puedo llegar a nodo x otro nodo en la frontera con costo 0

El costo acumulado de y es superior al camino directo a x en “a”

La elección greedy es correcta!

.UBAfiuba

FACULTAD DE INGENIERÍA

Presentación realizada en Abril de 2020