

Diseño y Análisis de Algoritmos

TEMA 3. ALGORITMOS VORACES



Universidad
Rey Juan Carlos

Tema 3. Algoritmos Voraces

Introducción

- Árboles de recubrimiento (expansión/generador) mínimo
 - Algoritmo de Prim
 - Algoritmo de Kruskal
- Caminos mínimos
 - Algoritmo de Dijkstra
- Heurísticas voraces
 - Coloreado de grafos
 - Viajante de comercio

Tema 3. Algoritmos Voraces

Árboles de recubrimiento mínimo

• Problema

Dado un grafo $G = (V,E)$ no dirigido y ponderado con pesos positivos, calcular el subgrafo conexo $T \subseteq G$, que conecte todos los vértices del grafo G y que la suma de aristas seleccionadas sea mínima

• Solución

Si el grafo es conexo, el subgrafo resultante es necesariamente un árbol

Tema 3. Algoritmos Voraces

Árboles de recubrimiento mínimo

Estrategias para resolver el problema

- Seleccionar la arista **más corta** en cada iteración
 - Algoritmo de Kruskal
- Seleccionar un **vértice al azar** y construir el árbol a partir de él, añadiendo las **arista de menor** peso que tenga un extremo en la solución y otro no
 - Algoritmo de Prim

Tema 3. Algoritmos Voraces

Árboles de recubrimiento mínimo

Aplicaciones

- Diseño de redes: telefónicas, eléctricas, hidráulicas, ordenadores, carreteras
 - Construcción de redes de mínimo coste
 - Refuerzo de líneas críticas
 - Identificación de cuellos de botella
 - Enrutamiento (evitar ciclos)
- Soluciones aproximadas a problemas NP
- Algoritmos de agrupamiento (*clustering*)
- ...

Tema 3. Algoritmos Voraces

Árboles de recubrimiento mínimo

Datos relevantes

- **Candidatos:** conjunto de aristas E
- **Solución:** se ha construido un árbol con $|V|$ vértices y $|V|-1$ aristas (extensible a grafos no conexos)
- **Factibilidad:** no existe ningún ciclo
- **Objetivo:** minimizar la suma de los pesos de las aristas seleccionadas.
- **Selección?** Depende del algoritmo

Tema 3. Algoritmos Voraces

Árboles de recubrimiento mínimo

Algoritmo de Kruskal

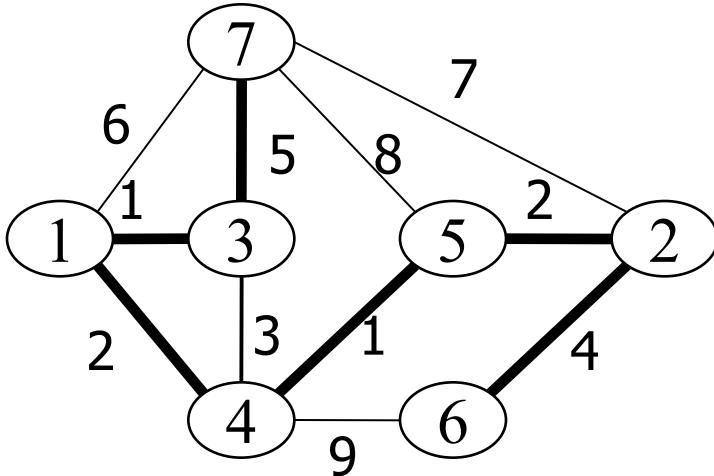
- Ordena las aristas de **mayor** a **menor** peso
- El árbol se construye a partir de **varias** componentes **conexas**
- Sólo se incluye una arista si une **dos** componentes **conexas**
- El algoritmo termina cuando sólo hay **una** componente **conexa**

Encuentra la **solución óptima** al problema

Tema 3. Algoritmos Voraces

Árboles de recubrimiento mínimo

Ejemplo algoritmo de Kruskal



Conjunto de candidatos ordenado:

~~{1,3}~~, ~~{4,5}~~, ~~{1,4}~~, ~~{2,5}~~,
~~{3,4}~~, ~~{2,6}~~, ~~{3,7}~~, ~~{1,7}~~,
~~{2,7}~~, ~~{5,7}~~, ~~{4,6}~~

Paso	Arista	Comp.conexas
Inicial.	-----	{1}{2}{3}{4}{5}{6}{7}
1	{1,3}	{1,3}{2}{4}{5}{6}{7}
2	{4,5}	{1,3}{4,5}{2}{6}{7}
3	{1,4}	{1,3,4,5}{2}{6}{7}
4	{2,5}	{1,3,2,4,5}{6}{7}
5	{3,4}	Rechazada
6	{2,6}	{1,3,2,4,5,6}{7}
7	{3,7}	{1,3,2,4,5,6,7}

Tema 3. Algoritmos Voraces

Esquema de la técnica

```
funcion Kruskal (G = <V,E>: Grafo) :Conjunto
    C  $\leftarrow$  ordenar(E)
    S  $\leftarrow$   $\emptyset$ 
    mientras C  $\neq \emptyset$  y no solucion(S) hacer
        (u,v)  $\leftarrow$  seleccionarPrimero(C)
        C  $\leftarrow$  C \ (u,v)
        si factible(S  $\cup$  (u,v)) entonces
            S  $\leftarrow$  S  $\cup$  (u,v)
        fin_si
    fin_mientras
    si solucion(S) entonces
        devolver S
    si_no
        devolver  $\emptyset$ 
    fin_si
fin_funcion
```

Tema 3. Algoritmos Voraces

Árboles de recubrimiento mínimo

Algoritmo de Kruskal. Detalles de implementación

- Complejidad: $O(|V||E|)$
- Fusión de componentes conexas con la estructura de datos *union-find*
 - Si se copia el menor en el mayor, el análisis amortizado nos dice que la fusión se puede hacer en $O(\log |V|)$

Tema 3. Algoritmos Voraces

Árboles de recubrimiento mínimo

Algoritmo de Prim

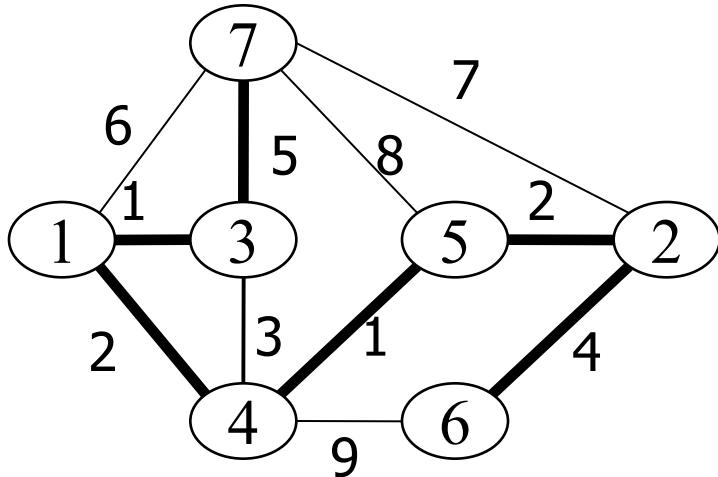
- El árbol se construye a partir de una **raíz**
- En cada iteración se añade **una nueva rama** (arista) al árbol
- Sólo se incluye una arista si no **genera ciclos**
- El algoritmo termina cuando sólo se han añadido **todos** los vértices

Encuentra la **solución óptima** al problema

Tema 3. Algoritmos Voraces

Árboles de recubrimiento mínimo

Ejemplo algoritmo de Prim



Conjunto de aristas ordenado:

~~{1,3}~~, ~~{4,5}~~, ~~{1,4}~~, ~~{2,5}~~,
~~{3,4}~~, ~~{2,6}~~, ~~{3,7}~~, ~~{1,7}~~,
~~{2,7}~~, ~~{5,7}~~, ~~{4,6}~~

Paso	Arista	Nodos que pertenecen al grafo
Inicial.	-----	{1}
1	{1,3}	{1,3}
2	{1,4}	{1,3,4}
3	{4,5}	{1,3,4,5}
4	{2,5}	{1,3,2,4,5}
5	{3,4}	Rechazada
6	{2,6}	{1,3,2,4,5,6}
7	{3,7}	{1,3,2,4,5,6,7}

Tema 3. Algoritmos Voraces

Esquema de la técnica

```
funcion Prim (G = (V, E) : Grafo) : Arbol
    S  $\leftarrow \emptyset$ 
    C  $\leftarrow \text{random}(V)$ 
    mientras C  $\neq V$  hacer
        (u, v)  $\leftarrow \text{seleccionar}(E)$  {u  $\in$  C, v  $\in$  V\ C}
        S  $\leftarrow S \cup (u, v)$  {La selección garantiza}
        C  $\leftarrow C \cup \{v\}$  {la factibilidad}
    fin mientras
    si solucion(S) entonces
        devolver S
    si_no
        devolver  $\emptyset$ 
    fin_si
fin_funcion
```

Tema 3. Algoritmos Voraces

Árboles de recubrimiento mínimo

Algoritmo de Prim. Detalles de implementación

- Complejidad: $O(|V||E|)$
- Utilizar una cola de prioridad ordenada por el coste de las aristas
 - Un extremo en la solución parcial y otro fuera
- La implementación eficiente garantiza

$$O(|A| \log |V|)$$

Tema 3. Algoritmos Voraces

Caminos mínimos

• Problema

Dado un grafo conexo $G = (V, E)$, dirigido y ponderado con pesos positivos, se toma uno, v , de los vértices como origen. El problema consiste en determinar la longitud mínima del camino que empieza en v hasta el resto

Tema 3. Algoritmos Voraces

Caminos mínimos

Estrategias para resolver el problema

- Mantener un conjunto de nodos ya explorados para los cuales ya se ha determinado el camino más corto desde v
 - Algoritmo de Dijkstra
- Propiedades de los caminos mínimos
 - Si $d(v, u)$ es la longitud del camino mínimo para ir desde v a u , entonces se satisface

$$d(v, u) \leq d(v, s) + d(s, u)$$

Tema 3. Algoritmos Voraces

Caminos mínimos

Datos relevantes

- **Candidatos:** conjunto de vértices de los que se conoce la distancia mínima desde el origen
- **Solución:** cuando no quedan candidatos
- **Factibilidad:** siempre es factible
- **Objetivo:** minimizar el camino del origen al resto de nodos
- **Selección:** candidato con menor distancia al origen

Tema 3. Algoritmos Voraces

Caminos mínimos

Algoritmo de Dijkstra

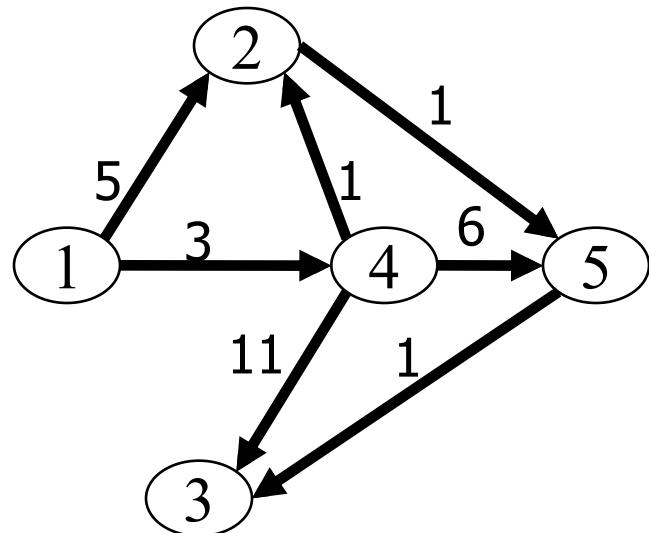
- Se parte de dos conjuntos de nodos ($V = S \cup C$)
 - S : nodos seleccionados para los que se conoce el camino mínimo
 - C : resto de nodos del grafo
- En cada iteración se escoge el nodo de C con menor distancia y se añade a S . Se recalcula los caminos a través del nodo seleccionado
- Si no existe arista, se considera distancia infinita

Encuentra la **solución óptima** al problema

Tema 3. Algoritmos Voraces

Caminos mínimos

Algoritmo de Dijkstra



Paso	Seleccionados	No Seleccionados	Vector de distancias			
			2	3	4	5
0	1	2,3,4,5	5	∞	3	∞
1	1,4	2,3,5	4	14	3	9
2	1,2,4	3,5	4	14	3	5
3	1,2,4,5	3	4	6	3	5

Tema 3. Algoritmos Voraces

Caminos mínimos

```
procedimiento Dijkstra ( $G = \langle V, A \rangle$ :grafo; DEV D:vector de distancias)
{Inicialización}
     $C \leftarrow \{2, 3, \dots, n\}$  { $S = V \setminus C$  solo existe implícitamente}
    para  $i \leftarrow 2$  hasta  $n$  hacer
         $D[i] = \text{Coste}(A, 1, i)$ 
    repetir  $n-2$  veces
         $v \leftarrow$  algún elemento de  $C$  con  $D[v]$  mínimo
         $C \leftarrow C \setminus \{v\}$  {implícitamente  $S \leftarrow S \cup \{v\}$ }
        para cada  $w \in C$  hacer
             $D[w] \leftarrow \min(D[w], D[v] + \text{Coste}(A, v, w))$ 
        fin_para
    fin_repetir
fin_procedimiento
```

Tema 3. Algoritmos Voraces

Caminos mínimos

Algoritmo de Dijkstra. Detalles de implementación

- Complejidad: $O(|V|^2)$
- Utilizar una cola de prioridad (*heap* de Fibonacci) la implementación eficiente garantiza

$$O(|A| \log |V|)$$

Tema 3. Algoritmos Voraces

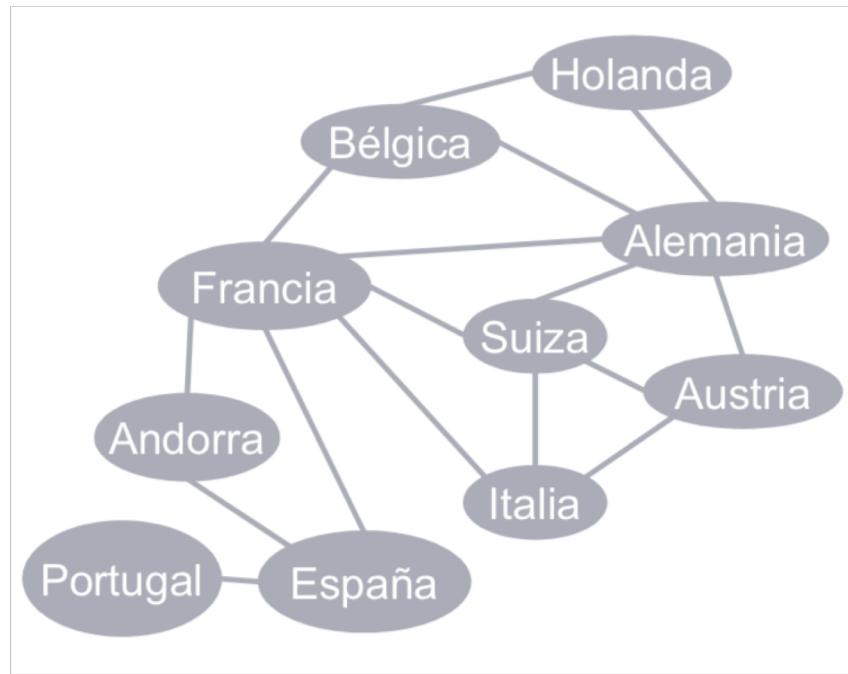
Heurísticas voraces

- **Problema del coloreado de grafos:** Dado un grafo $G = (V,E)$ no dirigido, asignar un color a cada vértice de tal forma que dos adyacentes no tengan el mismo color
- **Objetivo:** Minimizar el número de colores utilizados
- **Problema NP:** no existe ningún algoritmo eficiente que garantice un número mínimo de colores para grafos generales
 - Grafos planos: **4 colores** (Teorema de Appel-Hanke)

Tema 3. Algoritmos Voraces

Heurísticas voraces

• Problema del coloreado de grafos



Tema 3. Algoritmos Voraces

Heurísticas voraces

- **Solución eficiente (no óptima)**
 - El orden en el que se escoja el nodo es decisivo

procedimiento Coloreado ($G = \langle V, A \rangle$:grafo)

i $\leftarrow 1$

repetir Grafo no coloreado

Elegir un color c_i

Colorear todos los vértices que se pueda con c_i

Asegurar la factibilidad

i $\leftarrow 1$

fin_repetir

fin_procedimiento

Tema 3. Algoritmos Voraces

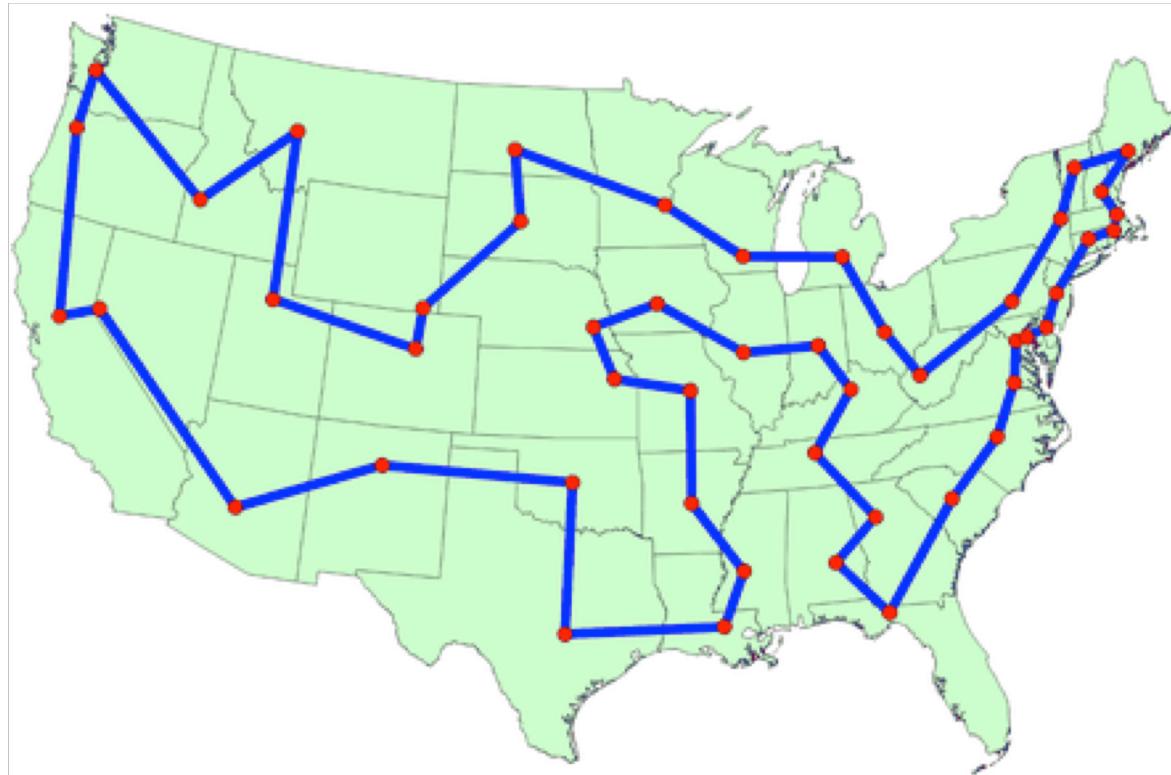
Heurísticas voraces

- **Problema del viajante:** *Dado un grafo $G = (V, E)$, encontrar un camino que empiece y acabe en un vértice v , pasando una única vez por cada vértice de V*
- **Objetivo:** *Obtener el circuito hamiltoniano de coste mínimo*
- **Problema NP:** no existe ningún algoritmo eficiente que garantice la optimalidad

Tema 3. Algoritmos Voraces

Heurísticas voraces

- **Problema del viajante:**



Tema 3. Algoritmos Voraces

Heurísticas voraces

- **Heurística 1:** escoger en cada iteración el vértice más cercano al último nodo añadido al circuito siempre que:
 - No se haya seleccionado previamente
 - No cierre el circuito
- **Heurística 2:** escoger las aristas de coste mínimo (como en Kruskal) pero garantizando que al final se forme un circuito