Algoritmos Voraces ($Greedy$)

Métodos Voraces

Objetivos

- Comprender la filosofía de diseño de métodos voraces
- Conocer las características de un problema resoluble mediante un algoritmo voraz
- Heurísticas voraces
- Resolución de diversos subproblemas

Características generales de los Algoritmos Voraces

- Simplicidad
- Miopía
- Facilidad de diseño, implementación y eficiencia

Ejemplo: Dar la vuelta

Algoritmo para dar cambio en una máquina expendedora:

Da_la_vuelta(n) : cj monedas a devolver

C = conjunto de monedas
$S \leftarrow \emptyset$
$s \leftarrow 0$
mientras $s \neq n$ **hacer**
  $x \leftarrow$ el mayor elemento de C tal que $s + x \leq n$
  si no existe ese elemento **entonces**
    Devolver "no encuentro la solución"
  fin si
  $S \leftarrow S \cup \{$una moneda de valor $x\}$
  $s \leftarrow s + x$
fin mientras
Devolver S

Elementos de un algoritmo voraz

- Conjunto de candidatos
- Función Solución
- Función Factible
- Función Selección
- Función Objetivo

Esquema general de un Algoritmo Voraz

Voraz(C: conjunto de candidatos) : conjunto solución

$S \leftarrow \emptyset$
mientras $C \neq \emptyset$ y no Solución(S) **hacer**
  $x \leftarrow$ Selección(C)
  $C \leftarrow C \setminus \{x\}$
  si factible($S \cup \{x\}$ **entonces**
    $S \leftarrow S \cup \{x\}$
  fin si
fin mientras
si Solución(S) **entonces**
  Devolver S
en otro caso
  Devolver "No hay solución"
fin si

Arbol Generador Minimal

Sea $G = (V, A)$ un grafo conexo no dirigido, ponderado, con pesos no negativos. Obtener un grafo parcial conexo tal que la suma de las aristas seleccionadas sea mínima.

Este subgrafo es *necesariamente* un árbol: **árbol generador minimal** (AGM) o *árbol de recubrimiento mínimo* (en inglés, *minimum spanning tree*).

Aplicaciones:

- red de comunicaciones de mínimo coste
- reforzar líneas críticas con mínimo coste

Dos enfoques de solución:

- aristas: Algoritmo de Kruskal
- nodos: Algoritmo de Prim

Algoritmo de Kruskal (I)

- Conjunto de candidatos: aristas
- Función Solución: el conjunto de aristas forma un AGM
- Función Factible: el conjunto de aristas no forma ciclos
- Función Selección: la arista de menor coste
- Función Objetivo: minimizar la suma de los costes de las aristas

Conjunto prometedor: se puede extender para producir una solución *óptima*

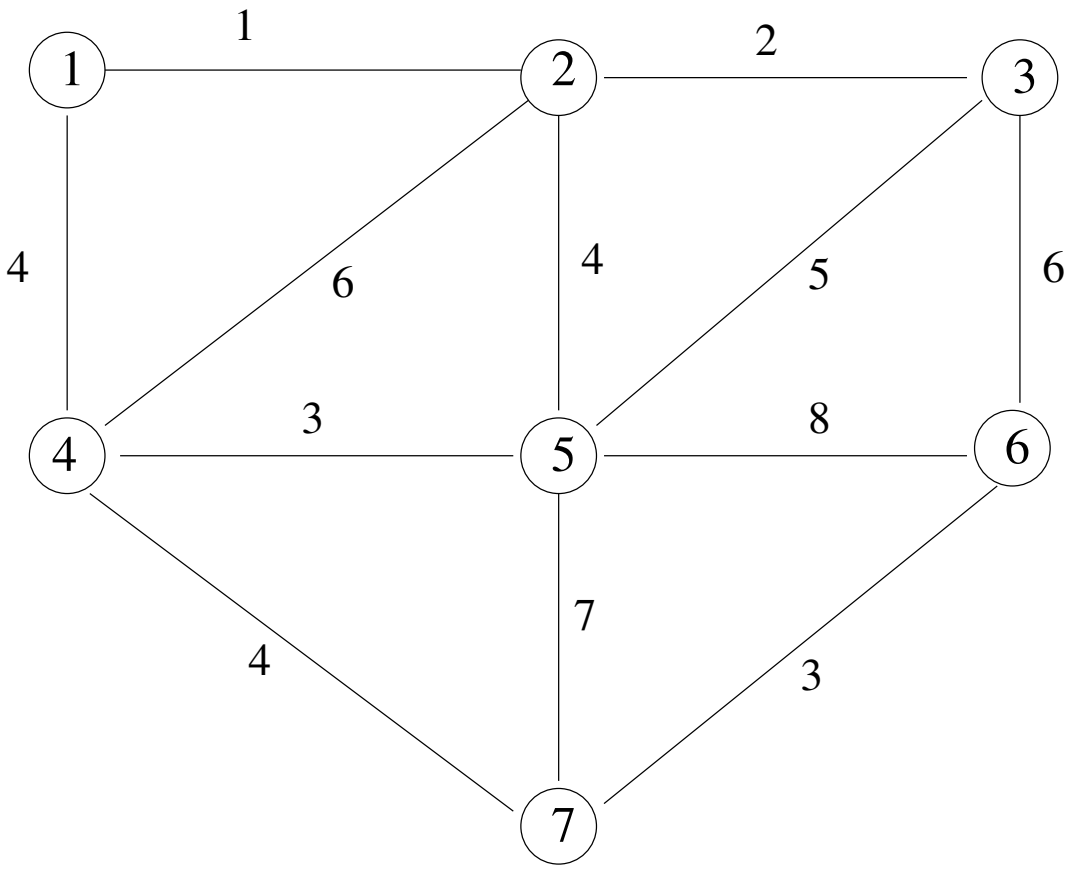
Algoritmo de Kruskal (II)

Kruskal($G = (V, A)$; $L : A \to \mathbb{R}$) : cj aristas

Ordenar A por longitudes crecientes
$n \leftarrow |V|$
$T \leftarrow \emptyset$
Iniciar n conjuntos, cada uno con un elemento distinto de V
repite
  $e \leftarrow \{u, v\}$, de mínimo coste
  compu $\leftarrow$ buscar(u)
  compv $\leftarrow$ buscar(v)
  si compu $\neq$ compv **entonces**
    fusionar(compu, compv)
    $T \leftarrow T \cup \{e\}$
  fin si
hasta $|T| = n - 1$
Devolver T

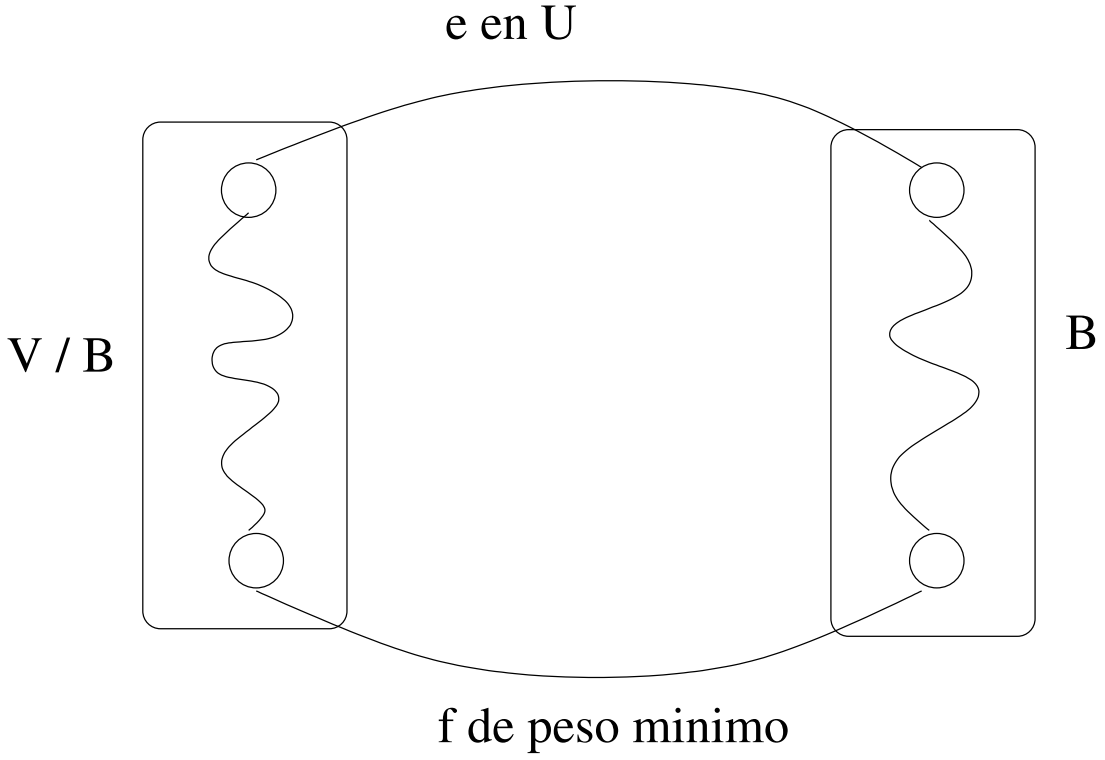
Eficiencia: $\mathrm{O}(a \log(n))$

Ejemplo: Algoritmo de Kruskal



Corrección del algoritmo

Lema. Sea $G = (V, A)$ un grafo conexo, no dirigido, ponderado con pesos no negativos. Sea $B \subset V$. Sea $T \subseteq A$ tal que no hay ninguna arista de T saliendo de B. Sea e la arista de menor peso que sale de B. Entonces $T \cup \{e\}$ es prometedor.



Algoritmo de Prim

- Conjunto de candidatos
- Función Solución
- Función Factible
- Función Objetivo
- Función Selección

Algoritmo de Prim

Construcción de un árbol al que se añade una rama en cada paso.

Prim($G = (V, A)$; $L : A \to \mathbb{R}$) : cj aristas

$T \leftarrow \emptyset$
$B \leftarrow \{$un elemento cualquiera de $V\}$
mientras $B \neq V$ **hacer**
  Encontrar una $e = \{u, v\}$ de peso mínimo tal que $u \in V \setminus B$ y $v \in B$
  $T \leftarrow T \cup \{e\}$
  $B \leftarrow B \cup \{u\}$
fin mientras
Devolver T

Algoritmo de Prim

Prim($L[1..n, 1..n]$) : cj aristas

$T \leftarrow \emptyset$
$B \leftarrow \{1\}$
para $i \leftarrow 2$ hasta n **hacer**
  vecino$[i] \leftarrow 1$
  pesmin$[i] \leftarrow L[i, 1]$
fin para
para $l \leftarrow 1$ hasta $n - 1$ **hacer**
  min $\leftarrow \infty$
  para $j \leftarrow 2$ hasta n **hacer**
    si $0 \leq$ pesomin$[j] < min$ **entonces**
      $min \leftarrow$ pesmin$[j]$
      $k \leftarrow j$
    fin si
  fin para
  $T \leftarrow T \cup \{\{k,$ vecino$[k]\}\}$
  pesmin$[k] \leftarrow -1$
  para $j \leftarrow 2$ hasta n **hacer**
    si $L[k, j] <$ pesmin$[j]$ **entonces**
      pesmin$[j] \leftarrow L[k, j]$
      vecino$[j] \leftarrow k$
    fin si
  fin para

fin para
Devolver T

Caminos mínimos en un grafo

Sea $G = (V, A)$ un grafo dirigido ponderado con pesos no negativos. Sea s un nodo fijo de V. Se desea encontrar los caminos de coste mínimo desde s hasta cualquier otro vértice, así como su coste.

Algoritmo de Dijkstra

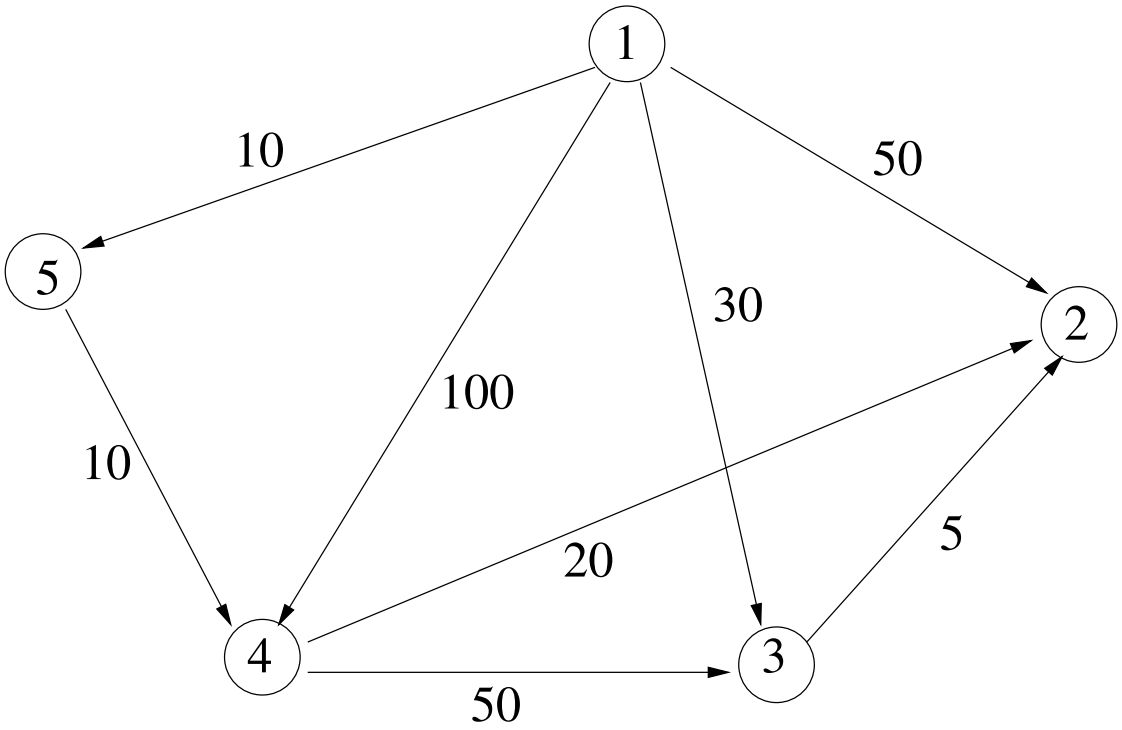
Dos conjuntos de nodos: $V = S \cup C$.
S: nodos ya seleccionados.

camino especial: camino más corto con nodos de S

Dijkstra($L = [1..n, 1..n]$): matriz$[2..n]$

$C \leftarrow 2, 3, \ldots, n$ {$S = V \setminus C$ sólo existe implícitamente}
para $i \leftarrow 2$ hasta n **hacer**
  $D[i] \leftarrow L[1, i]$
fin para
repite
  $v \leftarrow$ algún elemento de C que minimiza $D[v]$
  $C \leftarrow C \setminus \{v\}$ {implícitamente $S \leftarrow S \cup \{v\}$}
  para cada $w \in C$ **hacer**
    $D[w] \leftarrow min(D[w], D[v] + L[v, w])$
  fin para
hasta
Devolver D

Ejemplo



Paso	v	C	D
Inic.	—	$\{2,3,4,5\}$	$[50,30,100,10]$
1	5	$\{2,3,4\}$	$[50,30,20,10]$
2	4	$\{2,3\}$	$[40,30,20,10]$
3	3	$\{2\}$	$[35,30,20,10]$