Clase 05 MA3705. 21 de septiembre 2020.

Tabla

- Complejidad de algoritmos conocidos.
- Sistemas de Independencia

Complejidad de algoritmos conocidos

Algoritmos polinomiales y fuertemente polinomiales

B: Número de bits de la entrada.
N: Número de datos de la entrada.
n: Número de vértices de un grafo entrada.
m: Número de aristas de un grafo entrada.

- Algoritmo polinomial (o débilmente polinomial) es uno con complejidad $O(B^k)$ para algún k fijo, es decir es polinomial en el número de bits de la entrada.
- Algoritmo fuertemente polinomial es uno con complejidad $O(N^k)$ para algún k fijo, es decir es polinomial en el número de datos de la entrada.

F. Polinomial $\Rightarrow$ Polinomial

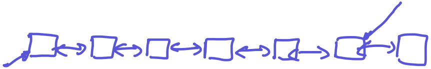
Ejemplos de algoritmos conocidos

Dada una lista de N números naturales en una lista/arreglo.

1. ¿Calcular el máximo?

N números.

Encontrar máximo: $O(N)$.



Encontrar máximo: $O(N)$

2. ¿Ordenar?

arreglo

Mergesort. $O(N \log N)$

Quicksort: $O(N^2)$

No existen algoritmos (en el modelo de comparación) de complejidad $\Theta(N \log N)$

BFS

Caso: $\boxed{n \leq m}$

U: [| | 0 | 1 |] arreglo

Búsqueda en amplitud (BFS):

Entrada: $G = (V, E)$, $r \in V$

$\boxed{U \leftarrow \{r\}}$, $F \leftarrow \emptyset$ } $O(1)$

$\boxed{\text{Cola} \leftarrow \emptyset.}$

Insertar aristas de $\delta(r)$ en Cola. } $O(\deg r)$

mientras Cola $\neq \emptyset$. **hacer**

  Extraer primer e de Cola. e sale

  si $e \in \delta(U)$, $u \in U$, $v \notin U$.

   entonces

    $U \leftarrow U + v$ $O(1)$

    $F \leftarrow F + e$ $O(1)$

    Insertar aristas de $\delta(v)$ en Cola ← $O(\deg v)$

  fin

fin

devolver (U, F)

Mal conteo:

$O(\cancel{\deg r} + 1)$ grado maximo

m veces se entra a la cola

$O(m \cdot \text{grado máximo})$

↑ # veces (iteraciones)

↳ $O(m \cdot \text{grado máximo})$

Mejor conteo:

- Cada arista entra a lo más 2 veces a la cola
- Todo lo que se relaciona con esta arista toma $O(1)$.

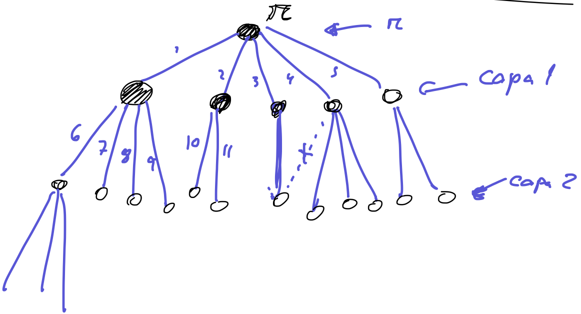
$\boxed{O(m) \text{ total.}}$

BFS

· Total: $O(n+m)$ ← sirve incluso si $n \gg m$.

Búsqueda en amplitud (BFS):

Entrada: $G = (V, E)$, $r \in V$
$U \leftarrow \{r\}$, $F \leftarrow \emptyset$
$\text{Cola} \leftarrow \emptyset$. $\quad O(n+m)$
Insertar aristas de $\delta(r)$ en Cola.
mientras $\text{Cola} \neq \emptyset$. **hacer** $\quad O(m)$
  Extraer primer e de Cola.
  si $e \in \delta(U)$, $u \in U$, $v \notin U$
   entonces
    $U \leftarrow U + v$
    $F \leftarrow F + e$
    Insertar aristas de $\delta(v)$ en Cola
  fin
fin
devolver (U, F)



BFS

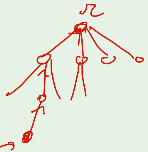
Búsqueda en amplitud (BFS):

Entrada: $G = (V, E)$, $r \in V$
$U \leftarrow \{r\}$, $F \leftarrow \emptyset$
$\text{Cola} \leftarrow \emptyset$.
Insertar aristas de $\delta(r)$ en Cola.
mientras $\text{Cola} \neq \emptyset$. **hacer**
    Extraer primer e de Cola.
    si $e \in \delta(U)$, $u \in U$, $v \notin U$
      entonces
        $U \leftarrow U + v$
        $F \leftarrow F + e$
        Insertar aristas de $\delta(v)$ en Cola
    fin
fin
devolver (U, F)

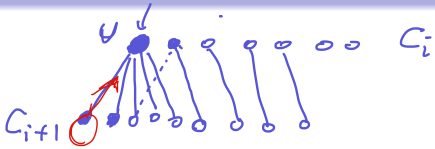
BFS modificado

Búsqueda en amplitud (BFS):

Entrada: $G = (V, E)$, $r \in V$
$U \leftarrow \{r\}$, $F \leftarrow \emptyset$, Nivel$(r) \leftarrow 0$, Padre$(v) \leftarrow \emptyset$ $\forall v$.
$C_0 \leftarrow \{r\}$, $C_1, \ldots, C_{n-1} \leftarrow \emptyset$.
para i *de* 0 *a* $n-1$ **hacer**
  mientras $C_i \neq \emptyset$ **hacer**
    Extraer primer u de C_i.
    para *cada* $v \in N(u)$ **hacer**
      si $v \notin U$ **entonces**
        $U \leftarrow U + v$, $F \leftarrow F + e$
        $\boxed{\text{Padre}(v)} \leftarrow \boxed{u}$, Nivel$(v) \leftarrow i + 1$,
        Insertar v a C_{i+1}
      fin
    fin
  fin
fin
devolver (U, F)



z, padre(z), padre(padre(z))



Propuesto

Demostrar por inducción en i que

$C^*(i) := \{u \in V : \text{Nivel}(u) = i\}$
$= \{u \in V : d(r, u) = i\}$.

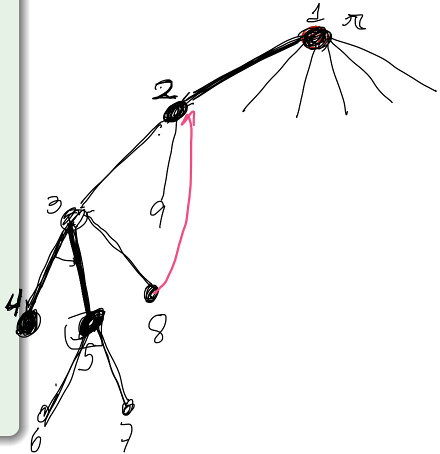
distancia: largo camino mínimo.

$O(n+m)$

Analisis: Igual a BFS.
$O(n+m)$ trabajo

Búsqueda en profundidad (DFS):

Entrada: $G = (V, E)$, $r \in V$
$U \leftarrow \{r\}$, $F \leftarrow \emptyset$
$\textsc{Pila} \leftarrow \emptyset$.
Insertar aristas de $\delta(r)$ en $\textsc{Pila}$.
mientras $\textsc{Pila} \neq \emptyset$. **hacer**
  Extraer último e de $\textsc{Pila}$.
  si $e \in \delta(U)$, $u \in U$, $v \notin U$
   entonces
    $U \leftarrow U + v$
    $F \leftarrow F + e$
    Insertar aristas de $\delta(v)$ en $\textsc{Pila}$
  fin
fin
devolver (U, F)



Búsqueda en profundidad (DFS):

Entrada: $G = (V, E)$, $r \in V$
$U \leftarrow \{r\}$, $F \leftarrow \emptyset$
$\textsc{Pila} \leftarrow \emptyset$.
Insertar aristas de $\delta(r)$ en $\textsc{Pila}$.
mientras $\textsc{Pila} \neq \emptyset$. **hacer**
  Extraer último e de $\textsc{Pila}$.
  si $e \in \delta(U)$, $u \in U$, $v \notin U$
   entonces
    $U \leftarrow U + v$
    $F \leftarrow F + e$
    Insertar aristas de $\delta(v)$ en $\textsc{Pila}$
  fin
fin
devolver (U, F)

Propuesto

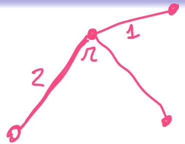
Sea $T = (V, F)$ un árbol DFS.

Demostrar que todas las aristas de $E \setminus F$ conectan a un vértice u con un vértice v en el único u-$\cancel{v}$ r camino en T.

Probar que esto no es necesariamente cierto para BFS.

$O(n+m)$

Algoritmo de Prim



Algoritmo de Prim (Prim 1957 - Jarník 1930):

Entrada: $G = (V, E)$ conexo, $r \in V$, $c : E \to \mathbb{R}$.
Elegir $r \in V$;
$U \leftarrow \{r\}$
$F \leftarrow \varnothing$

$\}$ $O(n+m)$

mientras $\delta(U) \neq \varnothing$ **hacer**
  Sea $e = uv \in \delta(U),\ u \in U,\ v \notin U$ tal que e es la arista de menor peso en $\delta(U)$
  $U \leftarrow U + v$
  $F \leftarrow F + e$
fin
devolver $T = (U, F)$

iteraciones $O(n)$
Cada iteración toma: $O(m)$

$\boxed{O(nm)}$

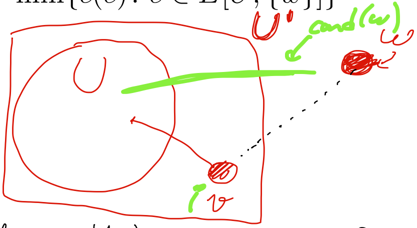
visitan todas las aristas recordando cual es la mejor de $\delta(U)$ que se ha visto.

Algoritmo de Prim.

Algoritmo de Prim (segunda implementación):

Entrada: $G = (V, E)$ conexo, $r \in V$
Elegir $r \in V$; $U \leftarrow \{r\}$; $F \leftarrow \varnothing$ $[O(n+m)]$
mientras $U \neq V$ **hacer**
  (re)calcular para todo $w \notin U$, cand(w). $\Big]\ O(n)$
  Elegir uw con $u \in U$, $w \notin U$ en
   $\arg\min\{v \in V \setminus U : c(\text{cand}(v))\}$ $\Big]\ O(n)$
  $U \leftarrow U + w$
  $F \leftarrow F + uw$ $\Big]\ O(1)$
fin
devolver $T = (U, F)$

Para cada $w \in V \setminus U$:
cand$(w) := uw$
$\Longleftrightarrow$
$c(wu) = \min\{c(e) \colon e \in E[U; \{w\}]\}$

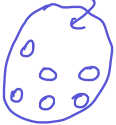


Recalcular: cand(w)
Si $vw \in E \wedge c(vw) \leq \text{cand}(w)$
entonces $\text{cand}(w) \leftarrow vw$. $\Big\}\ O(1)$

total: $O(n+m) + O(n \cdot n) = O(m+n^2) = \boxed{O(n^2)}$.
$\uparrow$ INICIO $\uparrow$ # iteraciones

Mejores implementaciones

Estructura de datos: Heaps (montículos)



números
→ extraer · mínimo
→ actualizar valores (decrecer valores)

Prim se puede implementar mejor con mejores estructuras de datos

En tiempo $O(n^2)$ usando arreglos y listas enlazadas. ✓
En tiempo $O((n+m)\log n)$ usando Heaps Binarios. ✓
En tiempo $O(m + n\log n)$ usando Heaps de Fibonacci.

Chazelle - ?

$O((n+m)\,\alpha(m))$ Ackermann

crece más lento que $\log^*(m)$

Abierto:
¿ $O(n+m)$ para calcular MST?

Opcional: $O((n+m)\log\log n)$
Prim + Boruvka

Desvío: Sistemas de independencia.

Sistemas de independencia

- **Sistema:** $(S, \mathcal{X})$ donde S es finito y $\mathcal{X} \subseteq 2^S$.
- S: conjunto de referencia del sistema.
- $(S, \mathcal{I})$ es un **Sistema de Independencia** si:
  - (vacío es independiente): $\emptyset \in \mathcal{I}$.
  - (cerrado para inclusión): $(\forall X \subseteq Y \subseteq S) \quad Y \in \mathcal{I} \implies X \in \mathcal{I}$.
- $\mathcal{I}$: Conjuntos independientes.
- Para $X \subseteq S$ llamamos $\boxed{\textbf{base de } X}$ a cualquier $B \subseteq X$ independiente y maximal para inclusión.

$\mathcal{Y} = \{X \text{ matching de } E\}$

$(E, \mathcal{Y})$ es sist. de indepencia

· Ejemplo: $S = [n]$

$\mathcal{Y} = \{X \subseteq S : |X| \leq 10\}$

. Ejemplo:

$G = (V, E)$ grafo.

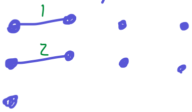
$\mathcal{Y} = \{X \subseteq E : X \text{ acíclico}\}$

Ejemplo

$G = (V, E)$ grafo

$M \subseteq E$ matching

si (V, M) tiene grado máximo 1

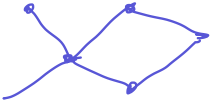


Ejemplos de sistemas de independencia

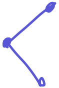
Ejemplo I: $X = \{3, 4, 5\}$

Bases de $X = X$

II Acíclicos



X

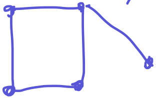


bases de X

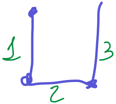
= bosques generadores

III Maching

G



X



Bases de $X = \{\{1,3\},\{2\}\}$