Clase 05 MA3705. 21 de septiembre 2020.

Tabla

- Complejidad de algoritmos conocidos.

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.

Ejemplos de algoritmos conocidos

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

1. ¿Calcular el máximo?

2. ¿Ordenar?

BFS

Búsqueda en amplitud (BFS):

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

BFS

BÚSQUEDA EN AMPLITUD (BFS):

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

BFS

Búsqueda en amplitud (BFS):

Entrada: $G = (V, E)$, $r \in V$
$U \leftarrow \{r\}$, $F \leftarrow \emptyset$
$\textsc{Cola} \leftarrow \emptyset$.
Insertar aristas de $\delta(r)$ en $\textsc{Cola}$.
mientras $\textsc{Cola} \neq \emptyset$. **hacer**
    Extraer primer e de $\textsc{Cola}$.
    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 $\textsc{Cola}$
    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$(r) \leftarrow 0$,
$C_0 \leftarrow \{r\}$, $C_1, \ldots, C_{n-1} \leftarrow \emptyset$.
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**
        $U \leftarrow U + v$, $F \leftarrow F + e$
        Padre$(v) \leftarrow u$, Nivel$(v) \leftarrow i + 1$,
        Insertar v a C_{i+1}
      fin
    fin
  fin
fin
devolver (U, F)

Propuesto

Demostrar por inducción en i que

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

DFS

Búsqueda en profundidad (DFS):

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

DFS

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-v camino en T.

Probar que esto no es necesariamente cierto para BFS.

Algoritmo de Prim

Algoritmo de Prim (Prim 1957 - Jarník 1930):

Entrada: $G = (V, E)$ conexo, $r \in V$
Elegir $r \in V$;
$U \leftarrow \{r\}$
$F \leftarrow \varnothing$
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 $T = (U, F)$

Algoritmo de Prim.

Algoritmo de Prim (segunda implementación):

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

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

Mejores implementaciones

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.

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 **base de** X a cualquier $B \subseteq X$ independiente y maximal para inclusión.

Ejemplos de sistemas de independencia

Ejemplos de sistemas de independencia

¿Cómo encontrar una base de un conjunto?

Obs: Todo conjunto independiente $I \subseteq X$ se puede extender a una base de X.

Cuidado: Oráculo de independencia

Propiedad útil:

Sea $s_1, \ldots, s_m$ es un ordenamiento de S y $S_i = \{s_1, \ldots, s_i\}$. Si aplicamos el algoritmo glotón en el orden b

¿Bases de tamaño/peso máximo?

Obs importante: Si todas las bases tienen el mismo tamaño. El siguiente algoritmo devuelve una base del conjunto completo. Para peso necesitaremos algo más fuerte.

Equicardinalidad de bases

Equicardinalidad de bases

Decimos que un sistema de independencia $(S, \mathcal{I})$ satisface **equicardinalidad de bases** si para todo $X \subseteq S$ las bases de X tienen el mismo cardinal.