Análisis de Algoritmos
Lectura: Técnicas de diseño de algoritmos: Algoritmos ambiciosos

Rodolfo Conde
rodolfo.conde@itam.mx

Instituto Tecnológico Autónomo de México
Maestría en Ciencias de la Computación

13 de noviembre de 2020

Contenido

Ejemplo introductorio: El problema de la selección de actividades

Elementos del método ambicioso

Ejemplo: Arboles generadores mínimos

Notas y ejercicios

Diseño y arquitectura de algoritmos

Usualmente, los algoritmos para resolver problemas de optimización realizan una serie de pasos en los cuales deben escoger entre ciertas opciones. Hemos visto como la PD es una buena herramienta para implementar algoritmos para estos problemas, sin embargo, para muchos de estos problemas la PD es como atacar un pajaro con un misil. Hay algoritmos más simples y eficientes que podrán resolverlos.

En esta lectura introducimos una técnica que muestra estos hechos: La técnica para diseñar Algoritmos ambiciosos. Nuestra referencia principal es [Cor+09].

A continuación

Ejemplo introductorio: El problema de la selección de actividades

Elementos del método ambicioso

Ejemplo: Arboles generadores mínimos

Notas y ejercicios

El problema de la selección de actividades

Comencemos con un ejemplo de un problema de optimización: Intuitivamente, el Problema de la Selección de Actividades consiste en lograr calendarizar varias actividades que utilizan un recurso común, de tal forma que se escoja un conjunto máximo de actividades compatibles mutuamente.

El problema de la selección de actividades

- Cada actividad a_i tiene asociado un tiempo de inicio s_i y un tiempo de finalización f_i tal que $0 \leq s_i < f_i < \infty$.
- Al ser seleccionada a_i, esta se ejecuta en el lapso de tiempo dado por el intervalo $[s_i, f_i)$.
- Dos actividades a_i, a_j son mutuamente compatibles si los intervalos $[s_i, f_i)$ y $[s_j, f_j)$ no se intersectan .

El problema de la selección de actividades

- Cada actividad a_i tiene asociado un tiempo de inicio s_i y un tiempo de finalización f_i tal que $0 \leq s_i < f_i < \infty$.
- Al ser seleccionada a_i, esta se ejecuta en el lapso de tiempo dado por el intervalo $[s_i, f_i)$.
- Dos actividades a_i, a_j son mutuamente compatibles si los intervalos $[s_i, f_i)$ y $[s_j, f_j)$ no se intersectan $(s_i \geq f_j$ ó $s_j \geq f_i)$.

El problema de la selección de actividades

El problema de La Selección de Actividades se define de manera formal como sigue:

Entrada: Dos secuencias numéricas de n elementos s, f, donde el intervalo $[s_i, f_i)$ representa el tiempo de inicio y finalización de la actividad a_i.

Salida: Un conjunto máximo $A \subseteq \{a_1, \ldots, a_n\}$ de actividades mutuamente compatibles.

El problema de la selección de actividades

El problema de La Selección de Actividades se define de manera formal como sigue:

Entrada: Dos secuencias numéricas de n elementos s, f, donde el intervalo $[s_i, f_i)$ representa el tiempo de inicio y finalización de la actividad a_i.

Salida: Un conjunto máximo $A \subseteq \{a_1, \ldots, a_n\}$ de actividades mutuamente compatibles.

En este ejemplo, asumiremos también que la secuencia f esta ordenada, es decir,

$$f_1 \leq f_2 \leq \cdots \leq f_{n-1} \leq f_n.$$

Selección de actividades: Ejemplo

i	1	2	3	4	5	6	7	8	9	10	11
s_i	1	3	0	5	3	5	6	8	8	2	12
f_i	4	5	6	7	9	9	10	11	12	14	16

Selección de actividades: Ejemplo

Intentemos resolver el problema de selección de actividades con Programación Dinámica.

Selección de actividades: Programación Dinámica

El primer paso, como ya sabemos, es encontrar si el problema exibe subestructura óptima en las soluciones óptimas.

Selección de actividades: Programación Dinámica

El primer paso, como ya sabemos, es encontrar si el problema exibe subestructura óptima en las soluciones óptimas.

- S_{ij} el conjunto de actividades que inician despues de a_i termina y finalizan antes que a_j inicie.

Selección de actividades: Programación Dinámica

El primer paso, como ya sabemos, es encontrar si el problema exibe subestructura óptima en las soluciones óptimas.

- S_{ij} el conjunto de actividades que inician despues de a_i termina y finalizan antes que a_j inicie.
- A_{ij} un sunconjunto máximo de actividades mutuamente compatibles de S_{ij}.

Selección de actividades: Programación Dinámica

El primer paso, como ya sabemos, es encontrar si el problema exibe subestructura óptima en las soluciones óptimas.

- S_{ij} el conjunto de actividades que inician despues de a_i termina y finalizan antes que a_j inicie.
- A_{ij} un sunconjunto máximo de actividades mutuamente compatibles de S_{ij}.
- A_{ij} incluye a la actividad a_k.

Selección de actividades: Programación Dinámica

El primer paso, como ya sabemos, es encontrar si el problema exibe subestructura óptima en las soluciones óptimas.

- S_{ij} el conjunto de actividades que inician despues de a_i termina y finalizan antes que a_j inicie.
- A_{ij} un sunconjunto máximo de actividades mutuamente compatibles de S_{ij}.
- A_{ij} incluye a la actividad a_k.
- Dos subproblemas: Resolver para S_{ik} y S_{kj}.

Selección de actividades: Programación Dinámica

- $A_{ik} = A_{ij} \cap S_{ik}$.

Selección de actividades: Programación Dinámica

- $A_{ik} = A_{ij} \cap S_{ik}$.
- $A_{kj} = A_{ij} \cap S_{kj}$.

Selección de actividades: Programación Dinámica

- $A_{ik} = A_{ij} \cap S_{ik}$.
- $A_{kj} = A_{ij} \cap S_{kj}$.
- $A_{ij} = A_{ik} \cup \{a_k\} \cup A_{kj}$.

Selección de actividades: Programación Dinámica

- $A_{ik} = A_{ij} \cap S_{ik}$.
- $A_{kj} = A_{ij} \cap S_{kj}$.
- $A_{ij} = A_{ik} \cup \{a_k\} \cup A_{kj}$.
- $|A_{ij}| = |A_{ik}| + |A_{kj}| + 1$.

Selección de actividades: Programación Dinámica

- $A_{ik} = A_{ij} \cap S_{ik}$.
- $A_{kj} = A_{ij} \cap S_{kj}$.
- $A_{ij} = A_{ik} \cup \{a_k\} \cup A_{kj}$.
- $|A_{ij}| = |A_{ik}| + |A_{kj}| + 1$.

El argumento usual de cortar y pegar una mejor subsolución óptima funciona.

Selección de actividades: Programación Dinámica

Esto nos lleva a considerar la siguiente ecuación recursiva para calcular el tamaño de una solución óptima:

$$c[i,j] = \begin{cases} 0 & S_{ij} = \varnothing, \\ \max_{a_k \in S_{ij}} \{c[i,k] + c[k,j] + 1\} & S_{ij} \neq \varnothing. \end{cases} \tag{1}$$

Selección de actividades: Programación Dinámica

$$c\left[i,j\right] = \begin{cases} 0 & S_{ij} = \varnothing, \\ \text{máx}_{a_k \in S_{ij}} \{c\left[i,k\right] + c\left[k,j\right] + 1\} & S_{ij} \neq \varnothing. \end{cases} \tag{1}$$

Usando un algoritmo recursivo, podemos memoizarlo (de arriba hacia abajo) o ir de abajo hacia arriba.

Selección de actividades: Programación Dinámica

Mas detalles en el pizarrón

Selección de actividades: Programación Dinámica

Mientras es posible resolver el problema de Selección de Actividades con PD, podemos utilizar una propiedad de este problema para desarrollar una solución con otro método para resolver problemas de optimización:

Selección de actividades: Programación Dinámica

Mientras es posible resolver el problema de Selección de Actividades con PD, podemos utilizar una propiedad de este problema para desarrollar una solución con otro método para resolver problemas de optimización:

El Método Ambicioso.

Selección de actividades: La alternativa ambiciosa

Sea S una instancia del problema de Selección de Actividades. En lugar de resolver todos los subproblemas que se deben resolver en cada paso de la PD, intentemos

Selección de actividades: La alternativa ambiciosa

Sea S una instancia del problema de Selección de Actividades. En lugar de resolver todos los subproblemas que se deben resolver en cada paso de la PD, intentemos

- Escoger uno de estos subproblemas,

Selección de actividades: La alternativa ambiciosa

Sea S una instancia del problema de Selección de Actividades. En lugar de resolver todos los subproblemas que se deben resolver en cada paso de la PD, intentemos

- Escoger uno de estos subproblemas,
- resolverlo,

Selección de actividades: La alternativa ambiciosa

Sea S una instancia del problema de Selección de Actividades. En lugar de resolver todos los subproblemas que se deben resolver en cada paso de la PD, intentemos

- Escoger uno de estos subproblemas,
- resolverlo,
- y repetir el proceso hasta obtener una solución.

Selección de actividades: La alternativa ambiciosa

Sea S una instancia del problema de Selección de Actividades.

- Escoger uno de estos subproblemas,

¿Cómo escogemos el subproblema?

Selección de actividades: La alternativa ambiciosa

Sea S una instancia del problema de Selección de Actividades.

- Escoger uno de estos subproblemas,

Debemos escoger un subproblema tal que nos proporcione la máxima ganacia al momento.

Selección de actividades: La alternativa ambiciosa

Sea S una instancia del problema de Selección de Actividades.

- Escoger uno de estos subproblemas,

Debemos escoger un subproblema tal que nos proporcione la máxima ganacia al momento.

Ser ambiciosos

Selección de actividades: La alternativa ambiciosa

Sea S una instancia del problema de Selección de Actividades.

- Escoger uno de estos subproblemas,

En el caso del problema de Selección de Actividades, la intuición nos dice que, dado una instancia del problema, debemos poner en el conjunto máximo la actividad que deje libre el recurso la mayor cantidad de tiempo.

Selección de actividades: La alternativa ambiciosa

Sea S una instancia del problema de Selección de Actividades.

- Escoger uno de estos subproblemas,

Ya que asumimos al inicio del ejemplo que las actividades estan ordenadas por tiempo de finalización, nuestra Selección ambiciosa es

Selección de actividades: La alternativa ambiciosa

Sea S una instancia del problema de Selección de Actividades.

- Escoger uno de estos subproblemas,

Ya que asumimos al inicio del ejemplo que las actividades estan ordenadas por tiempo de finalización, nuestra Selección ambiciosa es a_1

Selección de actividades: La alternativa ambiciosa

Sea S una instancia del problema de Selección de Actividades.

- resolverlo,

De este modo, nuestro candidato a conjunto máximo seria el conjunto $\{a_1\}$ unión con una solución óptima del siguiente conjunto de actividades:

Selección de actividades: La alternativa ambiciosa

Sea S una instancia del problema de Selección de Actividades.

- resolverlo,

De este modo, nuestro candidato a conjunto máximo seria el conjunto $\{a_1\}$ unión con una solución óptima del siguiente conjunto de actividades:
El conjunto de todas las actividades que comienzan despues de que a_1 ha terminado.

Selección de actividades: La alternativa ambiciosa

Sea S una instancia del problema de Selección de Actividades.

- resolverlo,

De este modo, nuestro candidato a conjunto máximo seria el conjunto $\{a_1\}$ unión con una solución óptima del siguiente conjunto de actividades:
Formalmente:

Selección de actividades: La alternativa ambiciosa

Sea S una instancia del problema de Selección de Actividades.

- resolverlo,

De este modo, nuestro candidato a conjunto máximo seria el conjunto $\{a_1\}$ unión con una solución óptima del siguiente conjunto de actividades:

$$S_1 = \{a_i \in S \mid s_i \geq f_1\}.$$

Selección de actividades: La alternativa ambiciosa

Sea S una instancia del problema de Selección de Actividades.

- y repetir el proceso hasta obtener una solución.

Y por supuesto, podemos intentar repetir este método para el conjunto S_1.

Selección de actividades: La alternativa ambiciosa

Sea S una instancia del problema de Selección de Actividades.

- Escoger uno de estos subproblemas,
- resolverlo,
- y repetir el proceso hasta obtener una solución.

A grandes rasgos, esto es el Método ambicioso.

Selección de actividades: La alternativa ambiciosa

En medio de este proceso, surge una pregunta importante:

Selección de actividades: La alternativa ambiciosa

¿Existe una solución óptima que contiene la elección ambiciosa?

Selección de actividades: La alternativa ambiciosa

¿Existe una solución óptima que contiene la elección ambiciosa?

Sí :)

Selección de actividades: La alternativa ambiciosa

¿Existe una solución óptima que contiene la elección ambiciosa?

No :(

Selección de actividades: La alternativa ambiciosa

¿Existe una solución óptima que contiene la elección ambiciosa?

Vamos a averiguarlo.

Selección de actividades: La alternativa ambiciosa

Lema 1

Sea S_k un conjunto (subproblema) del problema de Selección de Actividades y sea a_m una actividad en S_k que tiene el tiempo de finalización más pequeño. Entonces a_m esta incluida en algún subconjunto máximo de actividades mutuamente compatibles de S_k.

Selección de actividades: La alternativa ambiciosa

Lema 1

Sea S_k un conjunto (subproblema) del problema de Selección de Actividades y sea a_m una actividad en S_k que tiene el tiempo de finalización más pequeño. Entonces a_m esta incluida en algún subconjunto máximo de actividades mutuamente compatibles de S_k.

Demostración.

Selección de actividades: La alternativa ambiciosa

Lema 1

Sea S_k un conjunto (subproblema) del problema de Selección de Actividades y sea a_m una actividad en S_k que tiene el tiempo de finalización más pequeño. Entonces a_m esta incluida en algún subconjunto máximo de actividades mutuamente compatibles de S_k.

Demostración.

- Sea A_k un subconjunto máximo de actividades mutuamente compatibles de S_k.

Selección de actividades: La alternativa ambiciosa

Lema 1

Sea S_k un conjunto (subproblema) del problema de Selección de Actividades y sea a_m una actividad en S_k que tiene el tiempo de finalización más pequeño. Entonces a_m esta incluida en algún subconjunto máximo de actividades mutuamente compatibles de S_k.

Demostración.

- Sea A_k un subconjunto máximo de actividades mutuamente compatibles de S_k.
- Sea $a_j \in A_k$ una actividad con el tiempo de finalización más pequeño.

Selección de actividades: La alternativa ambiciosa

- Si $a_m = a_j$ entonces ya terminamos, pues A_k es un subconjunto máximo de actividades mutuamente compatibles de S_k y $a_m \in S_k$.

Selección de actividades: La alternativa ambiciosa

- Si $a_m = a_j$ entonces ya terminamos, pues A_k es un subconjunto máximo de actividades mutuamente compatibles de S_k y $a_m \in S_k$.
- Si $a_m \neq a_j$ entonces sea

Selección de actividades: La alternativa ambiciosa

- Si $a_m = a_j$ entonces ya terminamos, pues A_k es un subconjunto máximo de actividades mutuamente compatibles de S_k y $a_m \in S_k$.
- Si $a_m \neq a_j$ entonces sea

$$A'_k = A_k - \{a_j\} \cup \{a_m\}.$$

Selección de actividades: La alternativa ambiciosa

$$A'_k = A_k - \{a_j\} \cup \{a_m\}.$$

Mostraremos que A'_k es un subconjunto máximo de actividades mutuamente compatibles de S_k.

Selección de actividades: La alternativa ambiciosa

$$A'_k = A_k - \{a_j\} \cup \{a_m\}.$$

Todas las actividades del conjunto $A_k - \{a_j\} \subseteq A_k$ son mutuamente compatibles.

Selección de actividades: La alternativa ambiciosa

$$A'_k = A_k - \{a_j\} \cup \{a_m\}.$$

Solo falta probar que a_m es compatible con las actividades de $A_k - \{a_j\}$.

Selección de actividades: La alternativa ambiciosa

$$A'_k = A_k - \{a_j\} \cup \{a_m\}.$$

a_j es compatible con las actividades de $A_k - \{a_j\}$ y ademas tenemos que $f_j \leq f_l$ para toda $a_l \in A_k - \{a_j\}$.

Selección de actividades: La alternativa ambiciosa

$$A'_k = A_k - \{a_j\} \cup \{a_m\}.$$

a_j es compatible con las actividades de $A_k - \{a_j\}$ y ademas tenemos que $f_j \leq f_l$ para toda $a_l \in A_k - \{a_j\}$ (a_j tiene el tiempo de finalización más pequeño en A_k).

Selección de actividades: La alternativa ambiciosa

$$A'_k = A_k - \{a_j\} \cup \{a_m\}.$$

Pero a_m tiene el tiempo de finalización más pequeño en S_k, entonces tenemos que $f_m \leq f_j$.

Selección de actividades: La alternativa ambiciosa

$$A'_k = A_k - \{a_j\} \cup \{a_m\}.$$

Por lo tanto, para cualquier actividad $a_l \in A_k - \{a_j\}$,

Selección de actividades: La alternativa ambiciosa

$$A'_k = A_k - \{a_j\} \cup \{a_m\}.$$

Por lo tanto, para cualquier actividad $a_l \in A_k - \{a_j\}$,

$$s_l \geq f_j$$

Selección de actividades: La alternativa ambiciosa

$$A'_k = A_k - \{a_j\} \cup \{a_m\}.$$

Por lo tanto, para cualquier actividad $a_l \in A_k - \{a_j\}$,

Selección de actividades: La alternativa ambiciosa

$$A'_k = A_k - \{a_j\} \cup \{a_m\}.$$

Por lo tanto, para cualquier actividad $a_l \in A_k - \{a_j\}$,

$$s_l \geq f_m$$

Selección de actividades: La alternativa ambiciosa

$$A'_k = A_k - \{a_j\} \cup \{a_m\}.$$

NOTA: El caso $s_j \geq f_l$ no se puede dar.

Selección de actividades: La alternativa ambiciosa

$$A'_k = A_k - \{a_j\} \cup \{a_m\}.$$

Por lo tanto, a_m es compatible con todas las actividades de $A_k - \{a_j\}$. Ya que $|A_k| = |A'_k|$, concluimos que A'_k es un subconjunto máximo de actividades mutuamente compatibles de S_k y este conjunto contiene a a_m. $\square$

Selección de actividades: La alternativa ambiciosa

Entonces la selección ambiciosa es segura para resolver el problema de Selección de Actividades.

Selección de actividades: Algoritmo ambicioso recursivo

Require: Actividades ordenadas por tiempo de finalización en orden ascendente.

function REC-ACTIVITY-SELECTOR(s,f,k,n)   // $k = 0$ ($f_0 = 0$)
  $m = k + 1$
  while $m \leq n$ **and** $s[m] < f[k]$ **do**
    $m = m + 1$
  end while

  if $m \leq n$ **then**
    return $\{a_m\} \cup$REC-ACTIVITY-SELECTOR(s,f,m,n)
  else
    return $\varnothing$
  end if
end function

Selección de actividades: Algoritmo ambicioso recursivo

```
function R-A-S(s,f,k,n)
    m = k + 1
    while m ≤ n and s[m] < f[k] do
        m = m + 1
    end while

        if m ≤ n then
            return {a_m} ∪ R-A-S(s,f,m,n)
        else
            return ∅
        end if
    end function
```

El problema inicial es el subproblema S_0.

Selección de actividades: Algoritmo ambicioso recursivo

```
function R-A-S(s,f,k,n)
    m = k + 1
    while m ≤ n and  s[m] < f[k] do
        m = m + 1
    end while

        if m ≤ n then
            return {a_m} ∪ R-A-S(s,f,m,n)
        else
            return ∅
        end if
    end function
```

El parámetro k indica el subproblema actual S_k. a_k es la actividad que termina antes de que empiecen todas las activades que estan en S_k.

Selección de actividades: Algoritmo ambicioso recursivo

```
function R-A-S(s,f,k,n)
    m = k + 1
    while m ≤ n and  s[m] < f[k] do
        m = m + 1
    end while

        if m ≤ n then
            return {a_m} ∪ R-A-S(s,f,m,n)
        else
            return ∅
        end if
    end function
```

El ciclo while busca la primer actividad a_m que empieza despues de que termina a_k ($a_m \in S_k$).

Selección de actividades: Algoritmo ambicioso recursivo

```
function R-A-S(s,f,k,n)
    m = k + 1
    while m ≤ n and s[m] < f[k] do
        m = m + 1
    end while

        if m ≤ n then
            return {a_m} ∪ R-A-S(s,f,m,n)
        else
            return ∅
        end if
    end function
```

Si $m \leq n$, entonces a_m es la actividad buscada (la selección ambiciosa) y $S_m \neq \varnothing$ el nuevo subproblema a resolver

Selección de actividades: Algoritmo ambicioso recursivo

function $\mathrm{R\text{-}A\text{-}S}$(s,f,k,n)
 $m = k + 1$
 while $m \leq n$ **and** $s[m] < f[k]$ **do**
  $m = m + 1$
 end while

  if $m \leq n$ **then**
   return $\{a_m\} \cup \mathrm{R\text{-}A\text{-}S}$(s,f,m,n)
  else
   return $\varnothing$
  end if
 end function

En otro caso, No hay más activades que comiencen despues de que a_k termina, por lo tanto, $S_k = \varnothing$ y regresamos $\varnothing$.

Selección de actividades: Algoritmo ambicioso recursivo

function R-A-S(s,f,k,n)
    $m = k + 1$
    while $m \leq n$ **and** $s[m] < f[k]$ **do**
        $m = m + 1$
    end while

        if $m \leq n$ **then**
            return $\{a_m\} \cup$ R-A-S(s,f,m,n)
        else
            return $\varnothing$
        end if
    end function

Correctez

Selección de actividades: Algoritmo ambicioso recursivo

```
function R-A-S(s,f,k,n)
    m = k + 1
    while m ≤ n and  s[m] < f[k] do
        m = m + 1
    end while

        if m ≤ n then
            return {a_m} ∪ R-A-S(s,f,m,n)
        else
            return ∅
        end if
    end function
```

Complejidad de tiempo/espacio

A continuación

Ejemplo introductorio: El problema de la selección de actividades

Elementos del método ambicioso

Ejemplo: Arboles generadores mínimos

Notas y ejercicios

Método ambicioso

Un algoritmo construido aplicando el Método ambicioso obtiene la solución óptima a un problema al realizar una serie de decisiones sobre un conjunto de opciones. En cada decisión, el algoritmo elige la opción que parece mejor en ese momento. Esta heurística no siempre funciona para calcular una solución óptima, pero muchas veces funciona, como en el ejemplo del problema de Selección de Actividades.

Método ambicioso

Analicemos los elementos principales del método ambicioso y la forma en la que se construyen algoritmos (ambiciosos) usando este método

Método ambicioso: Elementos principales

La propiedad fundamental de un algoritmo ambicioso es que en cada paso, se hace una elección ambiciosa, es decir, una elección que nos proporciona el mejor valor en ese momento.

Método ambicioso: Elementos principales

Para lograr diseñar un algoritmo ambicioso, en general seguimos estos pasos básicos:

Método ambicioso: Elementos principales

1. Convertir el problema de optimización en uno equivalente en el cual hacemos una eleción y esto nos deja con un subproblema para resolver.

Método ambicioso: Elementos principales

1. Convertir el problema de optimización en uno equivalente en el cual hacemos una eleción (ambiciosa) y esto nos deja con un subproblema para resolver.

Método ambicioso: Elementos principales

1. Convertir el problema de optimización en uno equivalente en el cual hacemos una eleción (ambiciosa) y esto nos deja con un subproblema para resolver.
2. Probar que siempre hay una solución optima para el problema original que contiene la elección ambiciosa.

Método ambicioso: Elementos principales

1. Convertir el problema de optimización en uno equivalente en el cual hacemos una eleción (ambiciosa) y esto nos deja con un subproblema para resolver.
2. Probar que siempre hay una solución optima para el problema original que contiene la elección ambiciosa. Esto garantiza que la elección ambiciosa siempre es segura.

Método ambicioso: Elementos principales

1. Convertir el problema de optimización en uno equivalente en el cual hacemos una eleción (ambiciosa) y esto nos deja con un subproblema para resolver.
2. Probar que siempre hay una solución optima para el problema original que contiene la elección ambiciosa.
3. Demostrar subestructura óptima.

Método ambicioso: Elementos principales

1. Convertir el problema de optimización en uno equivalente en el cual hacemos una eleción (ambiciosa) y esto nos deja con un subproblema para resolver.
2. Probar que siempre hay una solución optima para el problema original que contiene la elección ambiciosa.
3. Demostrar subestructura óptima. Mostrar que si hacemos una elección ambiciosa, lo que queda es un subproblema con la propiedad de que si combinamos la solución óptima del subproblema con la elección ambiciosa, obtenemos una solución óptima del problema original.

Método ambicioso: ¿Cuándo aplica?

No existe un forma (que funcione en todos los casos) para decidir si un algoritmo ambicioso funcionará para resolver un problema de optimización en particular.

Método ambicioso: ¿Cuándo aplica?

Las propiedades de elección ambiciosa y subestructura óptima son los dos ingredientes principales para tomar la decisión.

La elección ambiciosa

La primer propiedad fundamental del método ambicioso es la elección ambiciosa: Construir una solución óptima global primero escogiendo una opción localmente óptima (ambiciosa) y entonces resolver un subproblema.

La elección ambiciosa

La primer propiedad fundamental del método ambicioso es la elección ambiciosa: Construir una solución óptima global primero escogiendo una opción localmente óptima (ambiciosa) y entonces resolver un subproblema.

- Diferencia importante respecto a PD.

La elección ambiciosa

La primer propiedad fundamental del método ambicioso es la elección ambiciosa: Construir una solución óptima global primero escogiendo una opción localmente óptima (ambiciosa) y entonces resolver un subproblema.

- Diferencia importante respecto a PD.
- PD: Resuelve los subproblemas relacionados con todas las opciones locales (arriba hacia abajo o viceversa).

La elección ambiciosa

La primer propiedad fundamental del método ambicioso es la elección ambiciosa: Construir una solución óptima global primero escogiendo una opción localmente óptima (ambiciosa) y entonces resolver un subproblema.

- Diferencia importante respecto a PD.
- PD: Resuelve los subproblemas relacionados con todas las opciones locales (arriba hacia abajo o viceversa).
- PD: Respuesta actual depende de soluciones a subproblemas

La elección ambiciosa

La primer propiedad fundamental del método ambicioso es la elección ambiciosa: Construir una solución óptima global primero escogiendo una opción localmente óptima (ambiciosa) y entonces resolver un subproblema.

- Diferencia importante respecto a PD.
- PD: Resuelve los subproblemas relacionados con todas las opciones locales (arriba hacia abajo o viceversa).
- PD: Respuesta actual depende de soluciones a subproblemas
- Método ambicioso: Elección ambiciosa puede depender de soluciones previas, pero no de soluciones futuras.

La elección ambiciosa

La primer propiedad fundamental del método ambicioso es la elección ambiciosa: Construir una solución óptima global primero escogiendo una opción localmente óptima (ambiciosa) y entonces resolver un subproblema.

- Diferencia importante respecto a PD.
- PD: Resuelve los subproblemas relacionados con todas las opciones locales (arriba hacia abajo o viceversa).
- PD: Respuesta actual depende de soluciones a subproblemas
- Método ambicioso: Elección ambiciosa puede depender de soluciones previas, pero no de soluciones futuras.
- ¡¡Se debe probar que la elección ambiciosa funciona!!

Subestructura Óptima

Un problema de optimización exibe subestructura óptima cuando una solución al problema contiene soluciones óptimas para los subproblemas.

Subestructura Óptima

Un problema de optimización exibe subestructura óptima cuando una solución al problema contiene soluciones óptimas para los subproblemas.

- Propiedad común con PD

Subestructura Óptima

Un problema de optimización exibe subestructura óptima cuando una solución al problema contiene soluciones óptimas para los subproblemas.

- Propiedad común con PD
- MA: Asumimos que al hacer una elección ambiciosa y combinar con una solución a un subproblema, da como resultado una solución óptima global.

Subestructura Óptima

Un problema de optimización exibe subestructura óptima cuando una solución al problema contiene soluciones óptimas para los subproblemas.

- Propiedad común con PD
- MA: Asumimos que al hacer una elección ambiciosa y combinar con una solución a un subproblema, da como resultado una solución óptima global.
- Este esquema, de forma implicita implica

Subestructura Óptima

Un problema de optimización exibe subestructura óptima cuando una solución al problema contiene soluciones óptimas para los subproblemas.

- Propiedad común con PD
- MA: Asumimos que al hacer una elección ambiciosa y combinar con una solución a un subproblema, da como resultado una solución óptima global.
- Este esquema, de forma implicita implica: Inducción en los subproblemas

Programación Ambiciosa vs Programación Dinámica

Hay que tener cuidado: En general, cuando descubrimos subestructura óptima, seguramente podremos aplicar la técnica de PD. Pero, no siempre es posible aplicar el MA. Y eso esta probado, existen problemas para los cuales es posible resolverlos con PD, pero los algoritmos ambiciosos no funcionan.

A continuación

Ejemplo introductorio: El problema de la selección de actividades

Elementos del método ambicioso

Ejemplo: Arboles generadores mínimos

Notas y ejercicios

Un problema interesante y útil de Grafos

El problema que veremos a continuación tiene varias aplicaciones prácticas[1] y tiene que ver con Teoría de Grafos. Es un problema de optimización (especificamente, minimización).

[1] Por ejemplo, en Diseño de circuitos electrónicos.

Teoría previa

Antes de comenzar, debemos conocer (a nivel básico) los siguientes objetos y definiciones:

Teoría previa

Antes de comenzar, debemos conocer (a nivel básico) los siguientes objetos y definiciones:

- Grafos.

Teoría previa

Antes de comenzar, debemos conocer (a nivel básico) los siguientes objetos y definiciones:

- Grafos.
- Conexidad.

Teoría previa

Antes de comenzar, debemos conocer (a nivel básico) los siguientes objetos y definiciones:

- Grafos.
- Conexidad (Componentes conexas).

Teoría previa

Antes de comenzar, debemos conocer (a nivel básico) los siguientes objetos y definiciones:

- Grafos.
- Conexidad (Componentes conexas).
- Árboles (generadores) y bosques.

Definición del problema

Sea $G = (V, E)$ un grafo no dirigido simple con una función de costos para las aristas $w\colon E \to \mathbb{R}$. En el problema que vamos a definir, nos interesa

Definición del problema

Sea $G = (V, E)$ un grafo no dirigido simple con una función de costos para las aristas $w\colon E \to \mathbb{R}$. En el problema que vamos a definir, nos interesa

- Encontrar un árbol generador T de G.

Definición del problema

Sea $G = (V, E)$ un grafo no dirigido simple con una función de costos para las aristas $w \colon E \to \mathbb{R}$. En el problema que vamos a definir, nos interesa

- Encontrar un árbol generador T de G.
- El costo de T, $w(T)$, definido como

$$w(T) = \sum_{\overline{uv} \in E(T)} w(\overline{uv})$$

debe ser mínimo sobre el conjunto de todos los árboles generadores de G.

El problema del árbol generador mínimo

El problema del árbol generador mínimo esta dado por los siguientes elementos:

Entrada: Un grafo $G = (V, E)$ conexo y simple con una función de costos $w\colon E \to \mathbb{R}$ para las aristas.

Salida: Un árbol generador T de G tal que la cantidad $w(T)$ es mínima entre todos los árboles generadores de G.

Diseño de una solución con el Método ambicioso

Construiremos una solución para este problema de forma incremental. El algoritmo al que llegaremos puede ser clasificado como un algoritmo ambicioso, pues en cada paso hace una elección que en ese momento luce como la mejor (ambiciosa).

Arboles generadores mínimos: Algoritmo genérico

Primero, presentamos un algoritmo genérico el cual puede ser especializado a uno con pasos muy especificos, siguiendo diversas metodologías para construir algoritmos.

Arboles generadores mínimos: Algoritmo genérico

Require: G grafo simple conexo y $w\colon E \to \mathbb{R}$

function GENERIC-MST(G, w)

$A = \varnothing$

while A no sea un árbol generador **do**

Encontrar una arista $\overline{uv}$ segura para A

$A = A \cup \{\overline{uv}\}$

end while

return A

end function

Arboles generadores mínimos: Algoritmo genérico

function GENERIC-MST(G, w)
    $A = \emptyset$

    while A no es árbol generador **do**
        Encontrar $\overline{uv} \in G.E$ segura para A
        $A = A \cup \{\overline{uv}\}$
    end while

    return A
end function

El algoritmo construye un AGM manteniendo el siguiente invariante:

Arboles generadores mínimos: Algoritmo genérico

function $\textsc{Generic-MST}(\mathsf{G}, \mathsf{w})$
    $A = \varnothing$

    while A no es árbol generador **do**
        Encontrar $\overline{uv} \in G.E$ segura para A
        $A = A \cup \{\overline{uv}\}$
    end while

    return A
end function

Al iniciar cualquier iteración, A es un subconjunto de un arbol generador mínimo.

Arboles generadores mínimos: Algoritmo genérico

```
function GENERIC-MST(G, w)
    A = ∅

    while A no es árbol generador do
        Encontrar uv ∈ G.E segura
        para A
        A = A ∪ {uv}
    end while

    return A
end function
```

No es difícil probar que el invariante es correcto.

Arboles generadores mínimos: Algoritmo genérico

function GENERIC-MST(G, w)
  $A = \varnothing$

  while A no es árbol generador **do**
    Encontrar $\overline{uv} \in G.E$ segura para A
    $A = A \cup \{\overline{uv}\}$
  end while

  return A
end function

El truco es como encontrar una arista segura.

Arboles generadores mínimos: Algoritmo genérico

function GENERIC-MST(G, w)
  $A = \varnothing$

  while A no es árbol generador **do**
    Encontrar $\overline{uv} \in G.E$ segura para A
    $A = A \cup \{\overline{uv}\}$
  end while

  return A
end function

Mientras el algoritmo entra en el ciclo while, A es un subconjunto de un AGM, así que debe existir una arista $\overline{uv}$ segura para A tal que $\overline{uv} \notin A$.

Arboles generadores mínimos: Algoritmo genérico

function GENERIC-MST(G, w)
    $A = \emptyset$

    while A no es árbol generador **do**
        Encontrar $\overline{uv} \in G.E$ segura para A
        $A = A \cup \{\overline{uv}\}$
    end while

    return A
end function

Sabiendo que un árbol generador $T \subseteq G$ es

Arboles generadores mínimos: Algoritmo genérico

function GENERIC-MST(G, w)
    $A = \emptyset$

    while A no es árbol generador **do**
        Encontrar $\overline{uv} \in G.E$ segura para A
        $A = A \cup \{\overline{uv}\}$
    end while

    return A
end function

Sabiendo que un árbol generador $T \subseteq G$ es

- conexo

Arboles generadores mínimos: Algoritmo genérico

function GENERIC-MST(G, w)
  $A = \varnothing$

  while A no es árbol generador **do**
    Encontrar $\overline{uv} \in G.E$ segura para A
    $A = A \cup \{\overline{uv}\}$
  end while

  return A
end function

Sabiendo que un árbol generador $T \subseteq G$ es

- conexo
- acíclico

Arboles generadores mínimos: Algoritmo genérico

function GENERIC-MST(G, w)
$A = \emptyset$

while A no es árbol generador **do**
Encontrar $\overline{uv} \in G.E$ segura para A
$A = A \cup \{\overline{uv}\}$
end while

return A
end function

Sabiendo que un árbol generador $T \subseteq G$ es

- conexo
- acíclico
- $|V(T)| = |V(G)|$ y $|E(T)| = |V(G)| - 1$.

Arboles generadores mínimos: Algoritmo genérico

function GENERIC-MST(G, w)
  $A = \varnothing$

  while A no es árbol generador **do**
    Encontrar $\overline{uv} \in G.E$ segura para A
    $A = A \cup \{\overline{uv}\}$
  end while

  return A
end function

Sabiendo que un árbol generador $T \subseteq G$ es

- acíclico

Una arista segura no debe introducir ciclos en A.

Arboles generadores mínimos: Algoritmo genérico

function GENERIC-MST(G, w)
    $A = \varnothing$

    while A no es árbol generador **do**
        Encontrar $\overline{uv} \in G.E$ segura para A
        $A = A \cup \{\overline{uv}\}$
    end while

    return A
end function

En general, dependiendo de la implementación particular del algoritmo, mientras A no sea un AGM, A será un bosque.

Arboles generadores mínimos: El Método ambicioso

Ahora intentaremos desarrollar una implementación del algoritmo genérico con el Método Ambicioso.

Arboles generadores mínimos: El Método ambicioso

El primer paso es probar que el problema exibe subestructura óptima.

Arboles generadores mínimos: El Método ambicioso

Ahora debemos proporner una selección ambiciosa y probar que esta elección siempre funciona.

Arboles generadores mínimos: El Método ambicioso

function GENERIC-MST(G, w)
    $A = \varnothing$

    while A no es árbol generador **do**
        Encontrar $\overline{uv} \in G.E$ segura para A
        $A = A \cup \{\overline{uv}\}$
    end while

    return A
end function

En cada iteración del algoritmo Generic-MST, debemos escoger una arista segura de un conjunto de aristas seguras para A

Arboles generadores mínimos: El Método ambicioso

function GENERIC-MST(G, w)
    $A = \emptyset$

    while A no es árbol generador **do**
        Encontrar $\overline{uv} \in G.E$ segura para A
        $A = A \cup \{\overline{uv}\}$
    end while

    return A
end function

Como Elección ambiciosa, proponemos escoger la arista segura con el peso más pequeño.

Arboles generadores mínimos: El Método ambicioso

¿Funciona nuestra elección ambiciosa?

Arboles generadores mínimos: El Método ambicioso

¿Funciona nuestra elección ambiciosa?

Vamos a averiguarlo

Arboles generadores mínimos: El Método ambicioso

Definición 2

Sea $G = (V, E)$ conexa con una función de pesos $w\colon E \to \mathbb{R}$, T un árbol generador de G y $A \subseteq T$. Denotamos al conjunto de aristas seguras de A por $S(A)$. Una arista $e \in S(A)$ es ligera si

$$w(e) = \min\{w(e') \mid e' \in S(A)\}.$$

Arboles generadores mínimos: El Método ambicioso

Lema 3

Sea $G = (V, E)$ *conexa con una función de pesos* $w\colon E \to \mathbb{R}$, $T \subseteq G$ *un AGM y* $A \subsetneq T$. *Si* $e \in S(A)$ *es una arista ligera, entonces existe* $T' \subseteq G$ *AGM tal que* $e \in T'$.

Arboles generadores mínimos: El Método ambicioso

¡¡Bien!! Nuestra elección ambiciosa funciona. Con nuestro método ambicioso, podemos construir una nueva versión del algoritmo Generic-MST, una versión más ambiciosa.

Arboles generadores mínimos: Algoritmo ambicioso

Require: G grafo simple conexo y $w: E \to \mathbb{R}$
function MST-GREEDY-K(G, w)
$A = \varnothing$
for $v \in G.V$ **do**
$C_v = \{v\}$
end for

Ordenar las aristas por peso w incrementalmente
for $\overline{uv} \in G.E$ en orden no decreciente de peso **do**
if $C_u \neq C_v$ **then**
$A = A \cup \{\overline{uv}\}$
Unir los conjuntos C_v y C_u
end if
end for
return A
end function

A continuación

Ejemplo introductorio: El problema de la selección de actividades

Elementos del método ambicioso

Ejemplo: Arboles generadores mínimos

Notas y ejercicios

Ejercicios del capítulo I

EJERCICIO 1

Prueba de manera formal la correctez del algoritmo ambicioso para el problema de Selección de Actividades.

EJERCICIO 2

Supongamos que en lugar de escoger la actividad que termina primero, escogemos la actividad que empieza al último que es compatible con todas las actividades previamente seleccionadas. Prueba que esta propuesta puede ser usada para construir un algoritmo ambicioso que proporcione una solución óptima.

Bibliografía I

Thomas H. Cormen y col. *Introduction to Algorithms, Third Edition*. 3rd. The MIT Press, 2009. ISBN: 0262033844, 9780262033848.