Teoria de Algoritmos
Capitulo 2: Algoritmos Divide y Venceras
Tema 4: Metodo general DV

- El metodo general
- Determinacion del umbral
- Algoritmos de busqueda
- Algoritmos de ordenacion
- Otras aplicaciones

Divide y Vencerás

- **dividir** el problema (P) en varios subproblemas (P_i)
- **vencer** los subproblemas (resolverlos)
- **combinar** las soluciones (S_i) de los subproblemas para obtener la solucion (S) del problema inicial

$$\begin{tikzcd} P \arrow[r] & \{P_i , i \in I\} \arrow[d] \\ S \arrow[u] & \{S_i , i \in I\} \arrow[l] \end{tikzcd}$$

- Este enfoque, sobre todo cuando se utiliza recursivamente, a menudo proporciona soluciones eficientes para problemas en los que los subproblemas son **versiones reducidas** y **resolubles** del problema original.
- Las ecuaciones recurrentes seran naturales en este metodo

Divide y Vencerás

- La eficacia de la técnica DV dependerá de varios factores
- Se trata de un proceso complejo que requiere muchas comprobaciones:
  - No sabemos si el problema P podrá descomponerse.
  - Los P_i tendrán la misma naturaleza entre ellos y con P; han de ser razonablemente pequeños en número; no pueden ser muy numerosos; cada P_i debe ser más sencillo de resolver que P.
  - Hay que esperar que tenga sentido integrar las S_i para obtener la solución final S.
  - Debemos esperar que S se corresponda con la solución del problema P.
- Aplicando este procedimiento debemos obtener un algoritmo que sea mejor que otro algoritmo que resuelva P sin esta técnica en peor tiempo.

Divide y Vencerás, justificación

- Supongamos un problema P, de tamaño n, que sabemos puede resolverse con un algoritmo (básico) A

$$t_1 (n) \leq cn^2.$$

- Dividimos P en 3 subproblemas de tamaños n/2, siendo cada uno de ellos del mismo tipo que A, y consumiendo un tiempo lineal la combinación de sus soluciones: $t(n) \leq dn$
- Tenemos un nuevo algoritmo B, Divide y Vencerás, que consumira un tiempo

$$t_B (n) = 3\ t_A (n/2) + t (n) \leq 3\ t_A (n/2) + dn \leq$$

$$\leq (3c/4)\ n^2 + dn.$$

- B tiene un tiempo de ejecución mejor que el algoritmo A, ya que disminuye la constante oculta

Divide y Vencerás, justificación

- Pero si cada subproblema se resuelve de nuevo con Divide y Venceras, podemos hacer un tercer algoritmo C recursivo que tendria un tiempo:

$$t_C(n) = 3\, t_C(n/2) + t(n)$$

- de forma que:

$$t_C(n) = \begin{cases} t_A(n) & \text{si } n \le n_0 \\ 3\, t_C(n/2) + t(n) & \text{si } n \ge n_0. \end{cases}$$

- $t_C(n)$ es mejor en eficiencia que los algoritmos A y B.
- El dividir el problema P en una serie de subproblemas, no significa que dichos subproblemas sean disjuntos. La división de P es exhaustiva pero no excluyente.
- Al valor n_0 se le denomina umbral y es fundamental para que funcione bien la tecnica

Algoritmo Divide y Vencerás

Funcion DV(P)

Si P es suficientemente pequeño o simple entonces devolver **basico** (P).

Descomponer P en subcasos P(1), P(2), ..., P(k) mas pequeños

para i = 1 hasta k hacer S(i) = DV(P(i))

recombinar las S(i) en S (solucion de P)

Devolver (S)

- Cuando k = 1 DV se llama simplificación.
- No hay ninguna regla para hallar k, sólo basándose en la experiencia sabemos que los DV con pocos subproblemas de tamaños parecidos entre ellos funcionan mejor.
- En función del valor del umbral n_0, obtendremos mejores o peores resultados. Su determinación es clave.

Analsis de algoritmos DV

- Cuando un algoritmo contiene una llamada recursiva a si mismo, generalmente su tiempo de ejecucion puede describirse por una recurrencia que da el tiempo de ejecucion para un caso de tamaño n en funcion de inputs de menor tamaño.
- En el caso de DV nos encontraremos recurrencias como:

$$T(n) = \begin{cases} t(n) & \text{si } n \leq c \\ aT(n/b) + D(n) + C(n) & \text{en otro caso} \end{cases}$$

- donde a es el numero de subproblemas, n/b el tamaño de estos, D(n) el tiempo de dividir el problema en los sub-problemas y C(n) el tiempo de combinacion de las soluciones de los subproblemas

Ejemplo: Las Torres de Hanoi

- Problema: Mover los n discos del tubo A al B usando el tubo C como tubo intermedio temporal
- Enfoque Divide y Venceras:
  - Dos subproblemas de tamaño n-1:
    - (1) Mover los n-1 discos mas pequeños de A a C
    - (*) Mover el nº disco mas pequeño de A a B es facil
    - (2) Mover los n-1 discos mas pequeños de C a B
  - El movimiento de los n-1 discos mas pequeños se hace con la aplicación recursiva del metodo

Determinación del umbral

- Es difícil hablar del umbral n_0 si no tratamos con implementaciones, ya que gracias a ellas conocemos las constantes ocultas que nos permitirán afinar el cálculo de dicho valor.
- El umbral no es único, pero si lo es en cada implementación.
- De partida no hay restricciones sobre el valor que puede tomar n_0, por tanto variará entre cero e infinito
  - Un umbral de valor infinito supone no aplicar nunca DV de forma efectiva, porque siempre estariamos resolviendo con el algoritmo básico siempre.
  - Si $n_0 = 1$, entonces estaríamos en el caso opuesto, ya que el algoritmo básico sólo actúa una vez, y se aplica la recursividad continuamente

Determinación del umbral

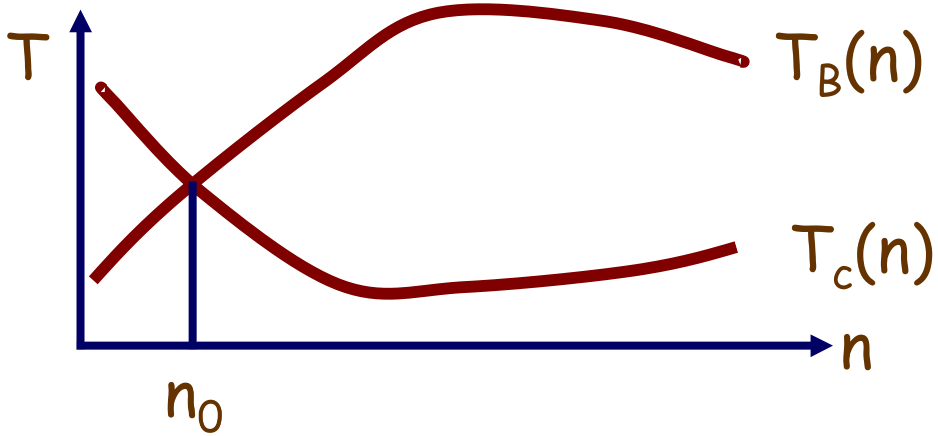
- Supongamos un algoritmo tal que

$$t_C(n) = t_A(n) \qquad \text{si } n \leq n_0$$
$$= 3\, t_C(n/2) + t(n) \qquad \text{si } n \geq n_0$$

- Una implementación concreta $t_A(n) = n^2$ y $t(n) = 16n$ (ms), y un caso de tamaño $n = 1024$.
- Las dos posibilidades extremas nos llevan a
  - Si $n_0 = 1$, $t_C(n) = 32$ m y 34sg
  - Si $n_0 = \infty$, $t_C(n) = 17$ m y 40 sg
- Si puede haber tan grandes diferencias, ¿como podremos determinar el valor optimo del umbral?
- Dos metodos: Experimental y Teorico

Determinación del umbral

- **Metodo experimental**
- Implementamos el algoritmo basico y el algoritmo DV
- Resolvemos para distintos valores de n con ambos algoritmos
- Hay que esperar que conforme n aumente, el tiempo del algoritmo basico vaya aumentando asintoticamente, y el del DV dismuyendo.



Determinación del umbral

- **Metodo teorico**
- La idea del enfoque experimental se traduce teoricamente a lo siguiente

$$t_C(n) = t_A(n) \qquad \text{si } n \le n_0$$
$$= 3\, t_C(n/2) + t(n) \qquad \text{si } n \ge n_0$$

- Cuando coinciden los tiempos de los dos algoritmos

$$t_A(n) = 3\, t_A(n/2) + t(n);\ t_C(n) = t_A(n) \text{ y } n = n_0$$

- Para una implementación concreta (por ejmplo, la anterior, $t_A(n) = n^2$ y $t(n) = 16n$ (ms) y $n = 1024$)

$$n^2 = \tfrac{3}{4} n^2 + 16\, n \rightarrow n = \tfrac{3}{4} n + 16$$

$$\mathbf{n_0 = 64}$$