Algoritmos Codiciosos

Ernesto Rodriguez - Juan Roberto Alvaro Saravia

Universidad Francisco Marroquin

ernestorodriguez@ufm.edu - juanalvarado@ufm.edu

Cordinación de Actividades

- Se tiene un conjunto de actividades $A = \{a_1, a_2, \ldots, a_n\}$
- Las actividades hacen uso de un recurso comun.
- Toda actividad tiene una hora de inicio (t_0) y hora de finalización (t_f).
- Dos actividades $a_1, a_2 \in A$ son **compatibles** si los intervalos $[t_0^{a_1}, t_f^{a_1})$ y $[t_0^{a_2}, t_f^{a_2})$ no traslapan.
- Las actividades $a_1, a_2, \ldots, a_n$ se encuentran ordenadas de forma ascendiente respecto a t_f, es decir: $t_f^{a_1} \leq t_f^{a_2} \leq \ldots \leq t_f^{a_n}$

Problema: Encontrar el subconjunto $A_c \subset A$ con la mayor cantidad de **actividades compatibles**.

Cordinación de Actividades

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

Actividades compatibles:

- $\{a_3, a_9, a_{11}\}$
- $\{a_1, a_4, a_8, a_{11}\}$
- $\{a_2, a_4, a_9, a11\}$

Programación Dinamica (idea)

Algorithm 1 Cordinar

```
1:  procedure CORDINAR(t_0, t_f, A)
2:      let A_c = {}
3:      for a_i in A do
4:          let opcion = {}
5:          if [t_0^{a_i}, t_f^{a_i}) ⊆ [t_0, t_f) then
6:              let resto = A\{a_i}
7:              let opcion_0 = Cordinar(t_0, t_0^{a_i}, resto)
8:              let opcion_f = Cordinar(t_f^{a_i}, t_f, resto)
9:              opcion ← {a_i} ∪ opcion_0 ∪ opcion_f
10:             A_c ← mayor(A_c, opcion)
11:     return A_c
```

Programación Dinamica

- El algoritmo hace uso de la memorización para evitar repetir trabajo.
- Sin embargo, ¿es necesario considerar todas las opciones?
- ¿Se puede decidir si agregar o no una actividad sin tener que resolver todos los sub-problemas?

Seleción Codiciosa

- Seleccionar la actividad que deje la mayor cantidad de tiempo libre.
- En otras palabras, seleccionar la actividad que termine lo más temprano posible.
- En caso de conflicto, hacer una seleción arbitraria.

Algoritmos Codiciosos

- La programación dinamica es una herramienta poderosa ya que considera un amplio espacio de combinaciones.
- Sin embargo, existen ocasiones en que la programación dinamica es un metodo "exagerado".
- Ciertos problemas pueden ser resueltos seleccionando la opción optima en cada paso, sin importar como se desenvuelba el resto de la busqueda.
- A estos algoritmos se les conoce como **algoritmos codiciosos**.

Creación de Algoritmos Codiciosos

1. Encontrar la sub-estructura optima
2. Crear una solución recursiva que busque esa sub-estructura
3. Demostrar que al hacer la elección codiciosa, solamente queda un sub-problema
4. Demostrar que es “seguro” hacer la solución codiciosa
5. Implementar el algoritmo recursivamente
6. Implementar el algoritmo iterativamente

Elecciones codiciosas vs Programación dinamica

- Las decisiones codiciosas solo consideran el “pasado”, mientras la programación dinamica considera tambien el “futuro”
- La programación dinamica es más general que los algoritmos codiciosos.
- Los algoritmos codiciosos son más eficientes.
- En algunas ocasiones, agregarle restricciones a un problema permite utilizar algoritmos codiciosos:
  - TSP vs Shortest Path
  - Candelarización vs Candelarización Generalizada
  - Ordenamiento vs Ordenamiento Generalizado
  - Knapsack fracionado vs Knapsack con Limite vs Knapsack Generalizado