Programación

PLG
Planning and Learning Group

Universidad Carlos III de Madrid

Algoritmos sobre Listas

Complejidad computacional

- Para resolver un problema unos algoritmos son mejores que otros
- Complejidad computacional: mide los recursos (tiempo, memoria) requeridos por un algoritmo
- Complejidad de un algoritmo: la del mejor algoritmo descubierto para resolverlo
- Notación BigO (peor caso)

Complejidad computacional – complejidad temporal (idea)

- Cuantas más instrucciones de CPU necesite un algoritmo más tardará
- Idea: Contar instrucciones de CPU
  - Asignar un valor a una variable
  - Comparar dos valores
  - Realizar una operación aritmética
  - Etc.
- Observación: habitualmente el número de instrucciones de CPU depende del tamaño de la entrada

Complejidad computacional – complejidad temporal (idea)

- Imaginemos un programa que encuentra el máximo elemento de un array de tamaño n

```
int max = A[0];
for (int i = 0; i < n; i++){
  if (A[i] > max)
     max = A[i]
}
```

- Si n es más grande el programa ejecuta más instrucciones
- La idea es determinar como aumenta el número de instrucciones, $f(n)$, a medida que crece n

Complejidad computacional – complejidad temporal (idea)

- No se trata de contar el número de instrucciones, sino de fijarse en lo que hace que $f(n)$ aumente rápidamente: bucles
- Se considera una cota superior (ajustada) de $f(n)$ para el peor caso
- Esto se denomina determinar el comportamiento asintótico del programa. Notación $O()$

Complejidad computacional – complejidad temporal (idea)

- $f(n) = 5n + 12$, el comportamiento asíntotico es $O(n)$
- $f(n) = 3n^2 + 5n + 12$, el comportamiento asíntotico es $O(n^2)$
- $f(n) = 3^n + 5n + 12$, el comportamiento asíntotico es $O(3^n)$

Complejidad computacional – complejidad temporal (idea)

- En un programa sin bucles es $O(1)$ (constante)
- En un programa con un bucle que puede recorrer completamente un array, como el del ejemplo anterior, es $O(n)$
- Si el array se recorre de nuevo por cada elemento (dos bucles anidados) es $O(n^2)$
- Si un algoritmo es $O(n^2)$ y para aplicarlo a 1000 elementos se tarda 3 segundos, para 2000 elementos se tardará $3 \times 2^2 = 12$ segundos y para 3000 serán $3 \times 3^2 = 27$ segundos, etc.

Ejemplos

- Extracción del elemento con un determinado índice de una array:

Ejemplos

- Extracción del elemento con un determinado índice de una array: $O(1)$

Ejemplos

- Extracción del elemento con un determinado índice de una array: $O(1)$
- Buscar una palabra en un diccionario:

Ejemplos

- Extracción del elemento con un determinado índice de una array: $O(1)$
- Buscar una palabra en un diccionario:
  - Elementos restantes
    - Iteración 0: n
    - Iteración 1: $n/2$
    - Iteración 3: $n/4$
    - ...
    - Iteración i: $n/2^i$
  - ¿En qué iteración estamos cuando queda 1 elemento?: $1 = n/2^i$
  - $i = log_2(n)$
  - Luego es $O(log\, n)$

Listas

- Diferentes tipos:
  - Ordenadas o no ordenadas
  - De longitud fija o dinámica
  - Con elementos repetidos o sin elementos repetidos

Algoritmos sobre Listas

- Vamos a trabajar sobre arrays (longitud fija)
- Tres tipos de algoritmos:
  - Búsqueda: buscar un elemento en una lista
  - Ordenación: ordenar una lista
  - Inserción: insertar un elemento en una lista

En este tema

Algoritmos sobre Listas

- Algoritmos de Búsqueda en Listas
- Algoritmos de Ordenación de Listas
- Algoritmos de Inserción en Listas

Algoritmos de búsqueda

- Búsqueda secuencial
  - Explicación y ejemplo
  - O(n)
  - Si la lista está ordenada tarda menos

Algoritmos de búsqueda

- Búsqueda secuencial
  - Explicación y ejemplo
  - O(n)
  - Si la lista está ordenada tarda menos
- Búsqueda binaria
  - Sólo para listas ordenadas
  - Explicación y ejemplo
  - O(log n)

En este tema

Algoritmos sobre Listas

- Algoritmos de Búsqueda en Listas
- Algoritmos de Ordenación de Listas
- Algoritmos de Inserción en Listas

Ordenación de Listas

- El criterio de ordenación determina el orden de los elementos. ej (“>” para números, orden alfabético para palabras)
- Algoritmos de Ordenación:
  - de ordenación interna: la ordenación se hace sobre la misma lista, sin utilizar ninguna otra auxiliar
  - de ordenación externa: se utiliza una lista auxiliar para realizar la ordenación

Métodos de Ordenación Interna

- Utilizamos la misma lista en el proceso de ordenación
- Se intercambian unos elementos por otros
- Algoritmos más conocidos:
  1. Burbuja, $O(n^2)$
     - `https://www.youtube.com/watch?v=JP5KkzdUEYI`

Métodos de Ordenación Interna

- Utilizamos la misma lista en el proceso de ordenación
- Se intercambian unos elementos por otros
- Algoritmos más conocidos:

1. Burbuja, $O(n^2)$
   - `https://www.youtube.com/watch?v=JP5KkzdUEYI`
2. Inserción directa, $O(n^2)$
   - `https://courses.cs.vt.edu/csonline/Algorithms/Lessons/InsertionCardSort/insertioncardsort.swf`

Métodos de Ordenación Interna

- Utilizamos la misma lista en el proceso de ordenación
- Se intercambian unos elementos por otros
- Algoritmos más conocidos:

1. Burbuja, $O(n^2)$
   - `https://www.youtube.com/watch?v=JP5KkzdUEYI`
2. Inserción directa, $O(n^2)$
   - `https://courses.cs.vt.edu/csonline/Algorithms/Lessons/InsertionCardSort/insertioncardsort.swf`
3. Selección directa, $O(n^2)$
   - `https://courses.cs.vt.edu/csonline/Algorithms/Lessons/SelectionCardSort/selectioncardsort.swf`

Métodos de Ordenación Interna

- Utilizamos la misma lista en el proceso de ordenación
- Se intercambian unos elementos por otros
- Algoritmos más conocidos:
  1. Burbuja, $O(n^2)$
     - `https://www.youtube.com/watch?v=JP5KkzdUEYI`
  2. Inserción directa, $O(n^2)$
     - `https://courses.cs.vt.edu/csonline/Algorithms/Lessons/InsertionCardSort/insertioncardsort.swf`
  3. Selección directa, $O(n^2)$
     - `https://courses.cs.vt.edu/csonline/Algorithms/Lessons/SelectionCardSort/selectioncardsort.swf`
  4. Otros más eficientes y también más complicados: QuickSort, $O(n \log n)$

En este tema

Algoritmos sobre Listas

- Algoritmos de Búsqueda en Listas
- Algoritmos de Ordenación de Listas
- Algoritmos de Inserción en Listas

Inserción en listas

- Lista ordenada: inserción ordenada de un elemento
- Lista no ordenada: inserción de un elemento en una posición
- Tipos de listas
  - Longitud fija: array con posiciones suficientes. Sólo las *n* primeras se consideran ocupadas
  - Longitud variable: se podría redimensionar el array, pero en realidad ésto se hace con otras estructuras de datos
- Algoritmo: insertar en el lugar correspondiente y desplazar los siguientes elementos en una posición