Data Structures and Algorithms (1)

Instructor: Ming Zhang
Textbook Authors: Ming Zhang, Tengjiao Wang and Haiyan Zhao
Higher Education Press, 2008.6 (the "Eleventh Five-Year" national planning textbook)
https://courses.edx.org/courses/PekingX/04830050x/2T2014/
Chapter 1 Overview

• Problem solving
• Data structures and abstract data types
• The properties and categories of algorithms
• Evaluating the efficiency of the algorithms
1.4 Complexity analysis of algorithm

Asymptotic analysis of algorithm

\[ f(n) = n^2 + 100n + \log_{10}n + 1000 \]

- \( f(n) \) is the growth rate as the data scale of \( n \) gradually increases.
- When \( n \) increases to a certain value, the item with the highest power of \( n \) in the equation has the biggest impact.
  - other items can be neglected.
1.4 Complexity analysis of algorithm

Asymptotic analysis of algorithm: Big O notation

- The definition domain of function $f$ and $g$ is nature numbers, the range is non negative real numbers.
- **Definition**: If positive number $c$ and $n_0$ exists, which makes for any $n \geq n_0$, $f(n) \leq cg(n)$,
- Then $f(n)$ is said to be in the set of $O(g(n))$, abbreviated as $f(n) = O(g(n))$,
- Big O notation: it represents the upper bound of the growth rare of a function
  - There could be more than one upper bounds of the growth rare of a function
- When the upper bound and the lower bound are the same, you can use Big $\Theta$ notation.
1.4 Complexity analysis of algorithm

**Big O notation**

- \( f(n) = O(g(n)) \), only when
  - There exists two parameters \( c > 0 \), \( n_0 > 0 \), for any \( n \geq n_0 \), \( f(n) \leq cg(n) \)
- iff \( \exists c, n_0 > 0 \) s.t. \( \forall n \geq n_0 : 0 \leq f(n) \leq cg(n) \)

\[ \begin{align*}
  f(n) & \quad g(n) \\
  \text{is large enough} & \quad \text{is the upper bound of } f(n)
\end{align*} \]
1.4 Complexity analysis of algorithm

**Time unit of Big O notation**

- Simple boolean or arithmetic operations
- Simple I/O
  - Input or output of a function
    - For example, operations such as read data from an array
  - Files I/O operations or keyboard input are not excluded
- Return of function
1.4 Complexity analysis of algorithm

Rules of operation of Big O notation

- **Rule of addition:** $f_1(n) + f_2(n) = \mathcal{O}(\max(f_1(n), f_2(n)))$
  - Sequential structure, if structure, switch structure

- **Rule of Multiplication:** $f_1(n) \cdot f_2(n) = \mathcal{O}(f_1(n) \cdot f_2(n))$
  - for, while, do-while structure

```plaintext
for (i=0; j<n; i++)
  for (j=i; j<n; j++)
    k++;

\[ \sum_{i=0}^{n-1} (n-i) = \frac{n(n-1)}{2} = \frac{n^2 - n}{2} = \mathcal{O}(n^2) \]
```
Asymptotic analysis of algorithm: Big $\Omega$ notation

- If positive number $c$ and $n_0$ exists, which makes for any $n \geq n_0$, $f(n) \geq cg(n)$,
- Then $f(n)$ is said to be in the set of $O(g(n))$, abbreviated as $f(n)$ is $O(g(n))$, or $f(n) = O(g(n))$
- The only difference of Big O notation and Big $\Omega$ notation is the direction of inequation.
- When you adopt the $\Omega$ notation, you’d better find the tightest (largest) lower bound of all the lower bound of the growth rate of the function.
Big \( \Omega \) notation

- \( f(n) = \Omega(g(n)) \)
  - iff \( \exists c, n_0 > 0 \text{ s.t. } \forall n \geq n_0, 0 \leq cg(n) \leq f(n) \)

- The only difference with Big O notation is the direction of inequation

\[ n \text{ is large enough} \]
\[ g(n) \text{ is the lower bound of } f(n) \]
Asymptotic analysis of algorithm: Big $\Theta$ notation

- When the upper bound and the lower bound are the same, you can use $\Theta$ notation.

- Definition:
  
  If a function is in the set of $O(g(n))$ and $\Omega(g(n))$, it is called $\Theta(g(n))$.

- In other words, when the upper bound and the lower bound are the same, you can use Big $\Theta$ notation.

- There exist $c_1$, $c_2$, and positive integer $n_0$, which makes for any positive integer $n > n_0$, The following two inequality are correct at the same time:

  $$c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$$
1.4 Complexity analysis of algorithm

**Big $\Theta$ notation**

- $f(n) = \Theta(g(n))$
  - iff $\exists c_1, c_2, n_0 > 0$ s.t. $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$, $\forall n \geq n_0$

- When the upper bound and the lower bound are the same, you can use $\Theta$ notation.

$n$ is large enough
g($n$) has the same growth rate with $f(n)$

$n_0$
1.4 Complexity analysis of algorithm

The growth rate curve of function $f(n)$

- $2^n$
- $n^2$
- $n \log_2 n$
- $n \log_2 n$
- $n$
1.4 Complexity analysis of algorithm

Problem space vs time overhead

The input data space of problem

![Graph showing time cost vs scale n]

- **worst**
- **average**
- **best**

Data space in average situation
1.4 Complexity analysis of algorithm

**Sequential Search**

- You are required to find a given K in an array with a scale of n sequentially
- **Best situation**
  - The first element of the array is K
  - You only need to check one element
- **Worst situation**
  - K is the last element of the array
  - You need to check all the n elements of the array.
Find value k sequentially—–the average case

• If value is distributed with equal probability
  - The probability that K occurs in every position is 1/n

• The average cost is O(n)

\[
\frac{1 + 2 + \ldots + n}{n} = \frac{n + 1}{2}
\]
Find value k sequentially—-the average case

- Distributed with different probability
  - Probability that K occurs in position 1 is 1/2
  - Probability that K occurs in position 2 is 1/4
  - Probability that K occurs in other positions are all

\[
\frac{1 - 1/2 - 1/4}{n - 2} = \frac{1}{4(n - 2)}
\]

- The average cost is O(n)

\[
\frac{1}{2} + \frac{2}{4} + \frac{3 + \ldots + n}{4(n - 2)} = 1 + \frac{n(n + 1) - 6}{8(n - 2)} = 1 + \frac{n + 3}{8}
\]
1.3 Algorithm

Binary search

For sequential linear list that is in order

- $K_{\text{mid}}$: The value of the element that is in the middle of the array
  - If $k_{\text{mid}} = k$, the search is successful
  - If $k_{\text{mid}} > k$, the search continues in the left half
  - Otherwise, if $k_{\text{mid}} < k$, You can ignore the part that before mid and search will go on in the right part

- **Fast**
  - $k_{\text{mid}} = k$, search will be ended up
  - $K_{\text{mid}} \neq k$, reduce half of the searching range at least
1.4 Complexity analysis of algorithm

Performance analysis of binary search

- The largest search length is \( \left\lceil \log_2 (n + 1) \right\rceil \)

- The search length of the situation that failed is \( \left\lfloor \log_2 (n + 1) \right\rfloor \) or \( \left\lceil \log_2 (n + 1) \right\rceil \)

- The average cost is \( O(\log n) \)

- In complexity analysis of algorithm
  - The base of \( \log n \) is 2
  - When the base changed, the magnitude of algorithm will not change
1.4 Complexity analysis of algorithm

Time/Space tradeoff

• **Data structure**
  - A certain space to store every data item
  - A certain amount of time to perform a single basic operation

• **The cost and benefit**
  - limit of time and space
  - Software engineering
The space-time tradeoffs

- Increasing the space overhead may improve the algorithm's time overhead
- To save space, often need to increase the operation time
Selecting data structure and algorithm

- You need to analyze the problem carefully
  - Especially the logic relations and data types involved in the process of solving problems—problem abstraction, data abstraction
  - Preliminary design of data structure often precede the algorithm design

- Note the data structure of scalability
  - Consider when the size of input data changes, whether data structure is able to adapt to the evolution and expansion of problem solving
1.4 Complexity analysis of algorithm

Question: Selecting data structure and algorithm

• Goal of problem solving?

• Process of choosing data structure and algorithm?
Question: three elements of data structure

Which of the structures below are logical structure and has nothing to do with the storage and operation().

A. Sequential table  B. Hash table  
C. Linear list  D. Single linked list

The following terms ( _____ ) has nothing to do with the storage of data.

A. Sequential table  B. Linked list  
C. Queue  D. Circular linked list
Data Structures and Algorithms

Thanks

the National Elaborate Course (Only available for IPs in China)
http://www.jpk.pku.edu.cn/pkujpk/course/sjjg/

Ming Zhang, Tengjiao Wang and Haiyan Zhao
Higher Education Press, 2008.6 (awarded as the "Eleventh Five-Year" national planning textbook)