Advanced Counting Techniques

Chapter 8

With Question/Answer Animations

Chapter Summary

- Applications of Recurrence Relations
- Solving Linear Recurrence Relations
  - Homogeneous Recurrence Relations
  - Nonhomogeneous Recurrence Relations
- Divide-and-Conquer Algorithms and Recurrence Relations
- Inclusion-Exclusion

Recurrence 递推关系
Linear 线性
Homogeneous 齐次
Nonhomogeneous 非齐次
Inclusion-Exclusion 包含互斥
Divide-and-Conquer 分治

Divide-and-Conquer Algorithms and Recurrence Relations

Section 8.3

Section Summary

- Divide-and-Conquer Algorithms and Recurrence Relations
- Examples
  - Binary Search
  - Merge Sort
  - Fast Multiplication of Integers
- Master Theorem
- Closest Pair of Points (*not covered yet in these slides*)

Binary二分的
Merge合并
Multiplication操纵

Divide-and-Conquer Algorithmic Paradigm

Definition: A *divide-and-conquer algorithm* works by first *dividing* a problem into one or more instances of the same problem of smaller size and then *conquering* the problem using the solutions of the smaller problems to find a solution of the original problem.

Examples:

- Binary search, covered in Chapters 3 and 5: It works by comparing the element to be located to the middle element. The original list is then split into two lists and the search continues recursively in the appropriate sublist.
- Merge sort, covered in Chapter 5: A list is split into two approximately equal sized sublists, each recursively sorted by merge sort. Sorting is done by successively merging pairs of lists.

Instance 实例
Conquer 求解

Divide-and-Conquer Recurrence Relations

- Suppose that a recursive algorithm divides a problem of size n into $\boldsymbol{a}$ subproblems.
- Assume each subproblem is of size $\boldsymbol{n/b}$.
- Suppose $g(n)$ extra operations are needed in the conquer step.
- Then $f(n)$ represents the number of operations to solve a problem of size n satisisfies the following recurrence relation:

$$f(n) = af(n/b) + g(n)$$

- This is called a *divide-and-conquer recurrence relation.*

Example: Binary Search

- Binary search reduces the search for an element in a sequence of size n to the search in a sequence of size $n/2$. Two comparisons are needed to implement this reduction;
  - one to decide whether to search the upper or lower half of the sequence and
  - the other to determine if the sequence has elements.
- Hence, if $f(n)$ is the number of comparisons required to search for an element in a sequence of size n, then

$$f(n) = f(n/2) + 2$$

when n is even.

https://fhfirehuo.github.io/Attacking-Java-Rookie/Chapter03/BinarySearch.html

Example: Merge Sort

- The merge sort algorithm splits a list of n (assuming n is even) items to be sorted into two lists with $n/2$ items. It uses fewer than n comparisons to merge the two sorted lists.
- Hence, the number of comparisons required to sort a sequence of size n, is no more than than $M(n)$ where

$$M(n) = 2M(n/2) + n.$$

https://www.runoob.com/w3cnote/merge-sort.html

Estimating the Size of Divide-and-Conquer Functions

Theorem 1: Let f be an increasing function that satisfies the recurrence relation

$$f(n) = af(n/b) + cn^d$$

whenever n is divisible by b, where $a \geq 1$, b is an integer greater than 1, and c is a positive real number. Then

$$f(n) \text{ is } \begin{cases} O(n^{\log_b a}) & \text{if } a > 1 \\ O(\log n) & \text{if } a = 1. \end{cases}$$

Furthermore, when $n = b^k$ and $a \neq 1$, where k is a positive integer,

$$f(n) = C_1 n^{\log_b a} + C_2$$

where $C_1 = f(1) + c/(a-1)$ and $C_2 = -c/(a-1)$.

Divisible 可整除

Complexity of Binary Search

Binary Search Example: Give a big-O estimate for the number of comparisons used by a binary search.

Solution: Since the number of comparisons used by binary search is $f(n) = f(n/2) + 2$ where n is even, by Theorem 1, it follows that $f(n)$ is $O(\log n)$.

Estimating the Size of Divide-and-conquer Functions (*continued*)

Theorem 2. Master Theorem: Let f be an increasing function that satisfies the recurrence relation

$$f(n) = af(n/b) + cn^d$$

whenever $n = b^k$, where k is a positive integer greater than 1, and c and d are real numbers with c positive and d nonnegative. Then

$$f(n) \text{ is } \begin{cases} O(n^d) & \text{if } a < b^d, \\ O(n^d \log n) & \text{if } a = b^d, \\ O(n^{\log_b a}) & \text{if } a > b^d. \end{cases}$$

Complexity of Merge Sort

Merge Sort Example: Give a big-O estimate for the number of comparisons used by merge sort.

Solution: Since the number of comparisons used by merge sort to sort a list of n elements is less than $M(n)$ where $M(n) = 2M(n/2) + n$, by the master theorem $M(n)$ is $O(n \log n)$.

Inclusion-Exclusion

Section 8.5

Section Summary

- The Principle of Inclusion-Exclusion
- Examples

Principle 规则

Principle of Inclusion-Exclusion

- In Section 2.2, we developed the following formula for the number of elements in the union of two finite sets:

$$|A \cup B| = |A| + |B| - |A \cap B|$$

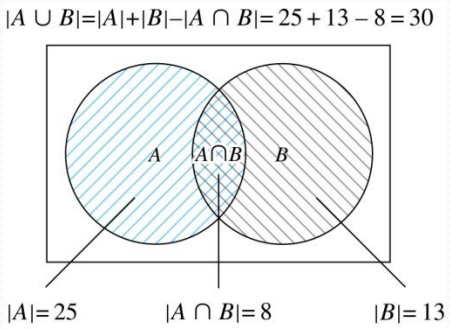
- We will generalize this formula to finite sets of any size.

Two Finite Sets

Example: In a discrete mathematics class every student is a major in computer science or mathematics or both. The number of students having computer science as a major (possibly along with mathematics) is 25; the number of students having mathematics as a major (possibly along with computer science) is 13; and the number of students majoring in both computer science and mathematics is 8. How many students are in the class?

Solution: $|A \cup B| = |A| + |B| - |A \cap B|$

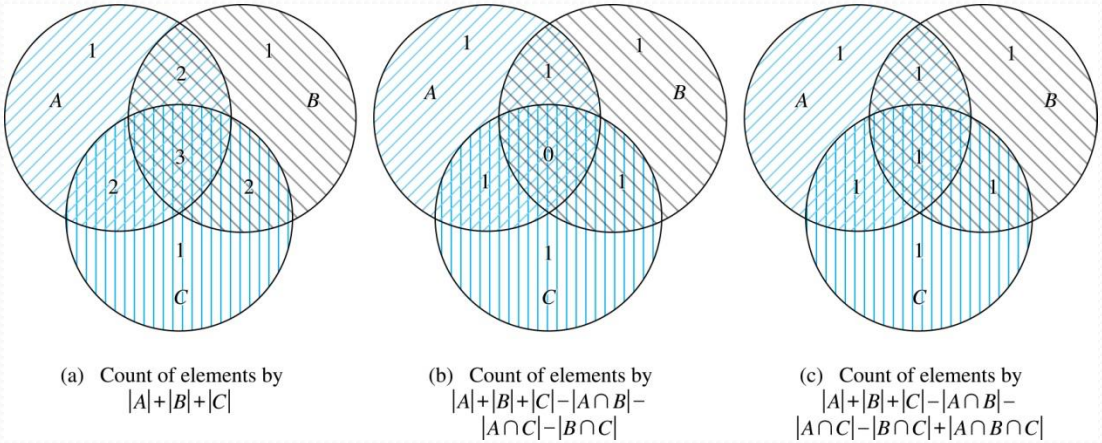
$= 25 + 13 - 8 = 30$



Three Finite Sets

$$|A \cup B \cup C| =$$

$$|A| + |B| + |C| - |A \cap B| - |A \cap C| - |B \cap C| + |A \cap B \cap C|$$



(a) Count of elements by $|A|+|B|+|C|$

(b) Count of elements by $|A|+|B|+|C|-|A \cap B|-|A \cap C|-|B \cap C|$

(c) Count of elements by $|A|+|B|+|C|-|A \cap B|-|A \cap C|-|B \cap C|+|A \cap B \cap C|$

Three Finite Sets Continued

Example: A total of 1232 students have taken a course in Spanish, 879 have taken a course in French, and 114 have taken a course in Russian. Further, 103 have taken courses in both Spanish and French, 23 have taken courses in both Spanish and Russian, and 14 have taken courses in both French and Russian. If 2092 students have taken a course in at least one of Spanish French and Russian, how many students have taken a course in all 3 languages.

Solution: Let S be the set of students who have taken a course in Spanish, F the set of students who have taken a course in French, and R the set of students who have taken a course in Russian. Then, we have

$|S| = 1232$, $|F| = 879$, $|R| = 114$, $|S \cap F| = 103$, $|S \cap R| = 23$, $|F \cap R| = 14$, and $|S \cup F \cup R| = 23$.

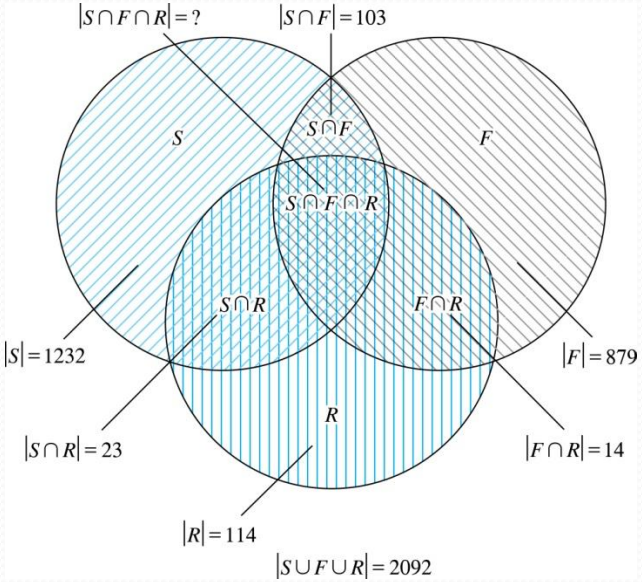
Using the equation

$$|S \cup F \cup R| = |S| + |F| + |R| - |S \cap F| - |S \cap R| - |F \cap R| + |S \cap F \cap R|,$$

we obtain $2092 = 1232 + 879 + 114 - 103 - 23 - 14 + |S \cap F \cap R|$.

Solving for $|S \cap F \cap R|$ yields 7.

Illustration of Three Finite Set Example



The Principle of Inclusion-Exclusion

Theorem 1. The Principle of Inclusion-Exclusion: Let $A_1, A_2, \ldots, A_n$ be finite sets. Then:

$$|A_1 \cup A_2 \cup \cdots \cup A_n| =$$

$$\sum_{i=1}^{n} |A_i| - \sum_{1 \leqslant i < j \leqslant n} |A_i \cap A_j|$$

$$+ \sum_{1 \leqslant i < j < k \leqslant n} |A_i \cap A_j \cap A_k| - \cdots + (-1)^{n+1} |A_1 \cap \cdots \cap A_n|$$

The Principle of Inclusion-Exclusion (*continued*)

Proof: An element in the union is counted exactly once in the right-hand side of the equation. Consider an element a that is a member of r of the sets $A_1, \ldots, A_n$ where $1 \leq r \leq n$.

- It is counted $C(r,1)$ times by $\sum |A_i|$
- It is counted $C(r,2)$ times by $\sum |A_i \cap A_j|$
- In general, it is counted $C(r,m)$ times by the summation of m of the sets A_i.

right-hand side 左手边，数学中常用语

The Principle of Inclusion-Exclusion (cont)

- Thus the element is counted exactly

$$C(r,1) - C(r,2) + C(r,3) - \cdots + (-1)^{r+1}\, C(r,r)$$

times by the right hand side of the equation.

- By Corollary 2 of Section 6.4 and **Binomial Theorem**, we have

$$C(r,0) - C(r,1) + C(r,2) - \cdots + (-1)^{r}\, C(r,r) = 0.$$

- Hence,

$$1 = C(r,0) = C(r,1) - C(r,2) + \cdots + (-1)^{r+1}\, C(r,r).$$