Data structures and algorithms

Chapter 1 Data Structures and Algorithms

1. The primary purpose of most computer programs is
   a) to perform a mathematical calculation.
   *b) to store and retrieve information.
   c) to sort a collection of records.
   d) all of the above.
   e) none of the above.
2. An integer is a
   *a) simple type
   b) aggregate type
   c) composite type
   d) a and b
   e) none of the above
3. A payroll records is a
   a) simple type
   b) aggregate type
   c) composite type
   *d) a and b
   e) none of the above
4. Which of the following should NOT be viewed as an ADT?
   a) list
   b) integer
   c) array
   *d) none of the above
5. A mathematical function is most like a
   *a) Problem
   b) Algorithm
   c) Program
6. An algorithm must be or do all of the following EXCEPT:　算法？
   a) correct
   b) composed of concrete steps
   *c) ambiguous（模糊地，引起歧义的）
   d) composed of a finite(限定的) number of steps
   e) terminate
7. A solution is efficient if
   a. it solves a problem within the require resource constraints.
   b. it solves a problem within human reaction time.
   c. it solves a problem faster than other known solutions.
   d. a and b.
   *e. a and c.
   f. b and c.

8. An array is

a) A contiguous (连续的)block of memory locations where each memory location stores a fixed-length data item.

b) An ADT composed of a homogeneous（同性质的）collection of data items, each data item identified by a particular number.

c) a set of integer values.

*d) a and b.

e) a and c.

f) b and c.

9. Order the following steps to selecting a data structure to solve a problem.

(1) Determine the basic operations to be supported.

(2) Quantify the resource constraints for each operation.

(3) Select the data structure that best meets these requirements.

(4) Analyze the problem to determine the resource constraints that any solution must meet.

a) (1, 2, 3, 4)

b) (2, 3, 1, 4)

c) (2, 1, 3, 4)

*d) (4 ,1, 2, 3)

e) (1, 4, 3, 2)

10. Searching for all those records in a database with key value between 10 and 100 is known as:

a) An exact match query. 准确查询

*b) A range query. 范围查询

c) A sequential search.

d) A binary search.

Chapter 2 Mathematical Preliminaries

1. A recurrence relation（递归关系） is often used to model programs with

a) for loops.

b) branch control like "if" statements.

*c) recursive calls.

d) function calls.

2. Which of the following is not a good proof technique.

a) proof by contradiction.(反证法)

*b) proof by example.

c) proof by mathematical induction.

3. We can use mathematical induction to: 数学归纳法

a) Find a closed-form solution for a summation.

*b) Verify（证实）a proposed closed-form solution for a summation.

c) Both find and verify a closed-form solution for a summation.

Chapter 3 Algorithm Analysis

1. A growth rate applies to:
   a) the time taken by an algorithm in the average case.
   b) the time taken by an algorithm as the input size grows.
   c) the space taken by an algorithm in the average case.
   d) the space taken by an algorithm as the input size grows.
   e) any resource you wish to measure for an algorithm in the average case.
   *f) any resource you wish to measure for an algorithm as the input size grows.
2. Pick the growth rate that corresponds to the most efficient algorithm as n gets large:
   a) 5n
   *b) 20 log n
   c) 2n^2
   d) 2^n
   这里要注意，好的算法的是随着 n 的增大增长率小的
3. Pick the growth rate that corresponds to the most efficient algorithm when n = 4.
   a) 5n（20）
   b) 20 log n（40）
   c) 2n^2（32）
   *d) 2^n（16）
5. Asymptotic analysis（渐进算法分析） refers to:
   a) The cost of an algorithm in its best, worst, or average case.
   *b) The growth in cost of an algorithm as the input size grows towards infinity（无限的时间或空间）.
   c) The size of a data structure.
   d) The cost of an algorithm for small input sizes
6. For an air traffic control system, the most important metric is:
   a) The best-case upper bound.
   b) The average-case upper bound.
   *c) The worst-case upper bound.
   d) The best-case lower bound.
   e) The average-case lower bound.
   f) The worst-case lower bound.
7. When we wish to describe the upper bound for a problem we use:
   *a) The upper bound of the best algorithm we know.
   b) The lower bound of the best algorithm we know.
   c) We can't talk about the upper bound of a problem because there can always be an arbitrarily slow algorithm.
8. When we describe the lower bound for a problem we use:
   a) The upper bound for the best algorithm we know.
   b) the lower bound for the best algorithm we know.
   c) The smallest upper bound that we can prove for the best algorithm that could possibly exist.
   *d) The greatest lower bound that we can prove for the best algorithm that could possibly exist.

9. When the upper and lower bounds for an algorithm are the same, we use:
   a) big-Oh notation.
   b) big-Omega notation.
   *c) Theta notation.
   d) asymptotic analysis.
   e) Average case analysis.
   f) Worst case analysis.
10. When performing asymptotic analysis, we can ignore constants and low order terms because:
   *a) We are measuring the growth rate as the input size gets large.
   b) We are only interested in small input sizes.
   c) We are studying the worst case behavior.
   d) We only need an approximation.
11. The best case for an algorithm refers to:
   a) The smallest possible input size.
   *b) The specific input instance of a given size that gives the lowest cost.
   c) The largest possible input size that meets the required growth rate.
   d) The specific input instance of a given size that gives the greatest cost.
12. For any algorithm:？ ？ ？ ？ ？
   *a) The upper and lower bounds always meet, but we might not know what they are.
   b) The upper and lower bounds might or might not meet.
   c) We can always determine the upper bound, but might not be able to determine the lower bound.
   d) We can always determine the lower bound, but might not be able to determine the upper bound.
13. If an algorithm is Θ(f(n)) in the average case, then it is:
   a) Ω(f(n)) in the best case.
   *b) Ω(f(n)) in the worst case.
   c) O(f(n)) in the worst case.
14. For the purpose of performing algorithm analysis, an important property of a basic operation is that:
   a) It be fast.
   b) It be slow enough to measure.
   c) Its cost does depend on the value of its operands(操作数).
   *d) Its cost does not depend on the value of its operands.
15. For sequential search,
   a) The best, average, and worst cases are asymptotically the same.
   *b) The best case is asymptotically（渐进）better than the average and worst cases.
   c) The best and average cases are asymptotically better than the worst case.
   d) The best case is asymptotically better than the average case, and the average case is asymptotically better than the worst case.

Chapter 4 Lists, Stacks and Queues

1. An ordered list is one in which:
   *a) The element values are in sorted order.
   *b) Each element a position within the list.
2. An ordered list is most like a:
   a) set.
   b) bag.
   *c) sequence.
3. As compared to the linked list implementation for lists, the array-based list implementation requires:
   a) More space
   b) Less space
   *c) More or less space depending on how many elements are in the list.
4. Here is a series of C++ statements using the list ADT in the book.

```
L1.append(10);
L1.append(20);
L1.append(15);
```

If these statements are applied to an empty list, the result will look like:
   a) < 10 20 15 >
   *b) < | 10 20 15 >
   c) < 10 20 15 | >
   d) < 15 20 10 >
   e) < | 15 20 10 >
   f) < 15 20 10 | >(fence 位置始终在最前面)
5. When comparing the array-based and linked implementations, the array-based implementation has:
   *a) faster direct access to elements by position, but slower insert/delete from the current position.
   b) slower direct access to elements by position, but faster insert/delete from the current position.
   c) both faster direct access to elements by position, and faster insert/delete from the current position.
   d) both slower direct access to elements by position, and slower insert/delete from the current position.
6. For a list of length n, the linked-list implementation's prev function requires worst-case time:
   a) O(1).
   b) O(log n).
   *c) O(n).
   d) O(n^2).
7. Finding the element in an array-based list with a given key value requires worst case time:
   a) O(1).
   b) O(log n).
   *c) O(n).
   d) O(n^2).

8. In the linked-list implementation presented in the book, a header node is used:

*a) To simplify special cases.

b) Because the insert and delete routines won't work correctly without it.

c) Because there would be no other way to make the current pointer indicate the first element on the list.

9. When a pointer requires 4 bytes and a data element requires 4 bytes, the linked list implementation requires less space than the array-based list implementation when the array would be:

a) less than 1/4 full.

b) less than 1/3 full.

*c) less than half full.

d) less than 2/3 full.

e) less than 3/4 full

f) never.

10. When a pointer requires 4 bytes and a data element requires 12 bytes, the linked list implementation requires less space than the array-based list implementation when the array would be: ?

*a) less than 1/4 full.

b) less than 1/3 full.

c) less than half full.

d) less than 2/3 full.

e) less than 3/4 full

f) never.

11. When we say that a list implementation enforces homogeneity,(同种) we mean that:

a) All list elements have the same size.

*b) All list elements have the same type.

c) All list elements appear in sort order.

12. When comparing the doubly and singly linked list implementations, we find that the doubly linked list implementation(双链表节约了时间 浪费了额外的空间)

*a) Saves time on some operations at the expense of additional space.

b) Saves neither time nor space, but is easier to implement.

c) Saves neither time nor space, and is also harder to implement.

13. We use a comparator() function in the Dictionary class ADT:

a) to simplify implementation.

*b) to increase the opportunity for code reuse.

c) to improve asymptotic efficiency of some functions.

14. All operations on a stack can be implemented in constant time except:

a) Push

b) Pop

c) The implementor's choice of push or pop (they cannot both be implemented in constant time).

*d) None of the above.

15. Recursion is generally implemented using 递归用栈

a) A sorted list.

*b) A stack.
c) A queue.

Chapter 5 Binary Trees

1. The height of a binary tree is:
   a) The height of the deepest node.
   b) The depth of the deepest node.
   *c) One more than the depth of the deepest node.
2. A full binary tree is one in which:
   *a) Every internal node has two non-empty children.
   b) all of the levels, except possibly the bottom level, are filled.(complete binary tree)
3. The relationship between a full and a complete binary tree is:
   a) Every complete binary tree is full.
   b) Every full binary tree is complete.
   *c) None of the above.
4. The Full Binary Tree Theorem（原理）states that:
   *a) The number of leaves in a non-empty full binary tree is one more than the number of internal nodes.
   b) The number of leaves in a non-empty full binary tree is one less than the number of internal nodes.
   c) The number of leaves in a non-empty full binary tree is one half of the number of internal nodes.
   d) The number of internal nodes in a non-empty full binary tree is one half of the number of leaves.
5. The correct traversal to use on a BST to visit the nodes in sorted order is:
   a) Preorder traversal.
   *b) Inorder traversal.
   c) Postorder traversal.
6. When every node of a full binary tree stores a 4-byte data field, two 4-byte child pointers, and a 4-byte parent pointer, the overhead fraction is approximately: ((4+4*2)/(4+4*2+4))
   a) one quarter.
   b) one third.
   c) one half.
   d) two thirds.
   *e) three quarters.
   f) none of the above.
   overhead fraction 结构性开销指的是非数据占用的部分占总的比例
   树的结构性开销要知道怎么算
7. When every node of a full binary tree stores an 8-byte data field and two 4-byte child pointers, the overhead fraction(结构性开销) is approximately:
   a) one quarter.
   b) one third.
   *c) one half.

d) two thirds.
e) three quarters.
f) none of the above.

8. When every node of a full binary tree stores a 4-byte data field and the internal nodes store two 4-byte child pointers, the overhead fraction is approximately:
a) one quarter.
b) one third.
*c) one half.
d) two thirds.
e) three quarters.
f) none of the above.

9. If a node is at position r in the array implementation for a complete binary tree, then its parent is at:
*a) (r - 1)/2 if r > 0
b) 2r + 1 if (2r + 1) < n
c) 2r + 2 if (2r + 2) < n
d) r - 1 if r is even
e) r + 1 if r is odd.

10. If a node is at position r in the array implementation for a complete binary tree, then its right child is at:
a) (r - 1)/2 if r > 0
b) 2r + 1 if (2r + 1) < n
*c) 2r + 2 if (2r + 2) < n
d) r - 1 if r is even
e) r + 1 if r is odd.

11. Assume a BST is implemented so that all nodes in the left subtree of a given node have values less than that node, and all nodes in the right subtree have values greater than or equal to that node. When implementing the delete routine, we must select as its replacement:
BST 的左子树的值小于根，右子树的值大于等于根
a) The greatest value from the left subtree.
*b) The least value from the right subtree. 删除的时候找右子树的最小值
c) Either of the above.

12. Which of the following is a true statement:
a) In a BST, the left child of any node is less than the right child, and in a heap, the left child of any node is less than the right child.
*b) In a BST, the left child of any node is less than the right child, but in a heap, the left child of any node could be less than or greater than the right child.
c) In a BST, the left child of any node could be less or greater than the right child, but in a heap, the left child of any node must be less than the right child.
d) In both a BST and a heap, the left child of any node could be either less than or greater than the right child

13. When implementing heaps and BSTs, which is the best answer?
a) The time to build a BST of n nodes is O(n log n), and the time to build a heap of n nodes is O(n log n).

b) The time to build a BST of n nodes is O(n), and the time to build a heap of n nodes is O(n log n).

*c) The time to build a BST of n nodes is O(n log n), and the time to build a heap of n nodes is O(n). 建树的时间复杂度是 nlogn，建堆的时间复杂度是 n

d) The time to build a BST of n nodes is O(n), and the time to build a heap of n nodes is O(n).

14. The Huffman coding tree works best when the frequencies for letters are

a) Roughly the same for all letters.

b) Skewed so that there is a great difference in relative frequencies for various letters.

不同字母的频率有显著差异

15. Huffman coding provides the optimal(最理想的；最佳的) coding when:

a) The messages are in English.

b) The messages are binary numbers.

c) The frequency of occurrence for a letter is independent of its context within the message. 字母的出现频率与其在消息中的上下文无关

d) Never.

Chapter 6 non-Binary Trees

1. The primary ADT access functions used to traverse a general tree are:

左孩子右兄弟

a) left child and right sibling

b) left child and right child

*c) leftmost child and right sibling

d) leftmost child and next child

2. The tree traversal that makes the least sense for a general tree is:

a) preorder traversal

*b) inorder traversal 对于一棵通用树，最没有意义的遍历

c) postorder traversal

3. The primary access function used to navigate the general tree when performing UNION/FIND is:

a) left child

b) leftmost child

c) right child

d) right sibling

*e) parent

4. When using the weighted union rule for merging disjoint sets, the maximum depth for any node in a tree of size n will be: **log n** **记下来好了，懒得推了**

a) nearly constant

*b) log n

c) n

d) n log n

e) n^2

5. We use the parent pointer representation for general trees to solve which problem?

a) Shortest paths

b) General tree traversal

*c) Equivalence classes 等价类问题

d) Exact-match query

6. When using path compression along with the weighted union rule for merging disjoint sets, the average cost for any UNION or FIND operation in a tree of size n will be:

*a) nearly constant

b) log n

c) n

d) n log n

e) n^2

7. The most space efficient representation for general trees will typically be:

a) List of children

*b) Left-child/right sibling

c) A K-ary tree.

8. The easiest way to represent a general tree is to:

a) convert to a list.

*b) convert to a binary tree.

c) convert to a graph.

9. As K gets bigger, the ratio of internal nodes to leaf nodes:（当 K 增加时，空指针数目也在不断增加） 内部节点与叶子节点的比例？

***a) Gets smaller.**

b) Stays the same.

c) Gets bigger.

d) Cannot be determined, since it depends on the particular configuration of the tree.

10. A sequential tree representation is best used for:（顺序树表示法：目的在于存储一系列结点的值，其中包含了尽可能少的但对于重建树结构必不可少的的信息。节省空间）

*a) Archiving the tree to disk. 节省空间

b) Use in dynamic in-memory applications.

c) Encryption algorithms.

d) It is never better than a dynamic representation.

Chapter 7 Internal Sorting

1. A sorting algorithm is stable if it:

a) Works for all inputs.

*b) Does not change the relative ordering of records with identical key values.(不改变关键码值相同的的纪录的相对顺序)

c) Always sorts in the same amount of time (within a constant factor) for a given input size.

2. Which sorting algorithm does not have any practical use?

a) Insertion sort.

*b) Bubble sort.

c) Quicksort.

d) Radix Sort.

e) a and b.

3. When sorting n records, Insertion sort has best-case cost:
   a) O(log n).
   *b) O(n).
   c) O(n log n).
   d) O(n^2)
   e) O(n!)
   f) None of the above.
4. When sorting n records, Insertion sort has worst-case cost:
   a) O(log n).
   b) O(n).
   c) O(n log n).
   *d) O(n^2)
   e) O(n!)
   f) None of the above.
5. When sorting n records, Quicksort has worst-case cost:
   a) O(log n).
   b) O(n).
   c) O(n log n).
   *d) O(n^2)
   e) O(n!)
   f) None of the above.
6. When sorting n records, Quicksort has average-case cost:
   a) O(log n).
   b) O(n).
   *c) O(n log n).
   d) O(n^2)
   e) O(n!)
   f) None of the above.
7. When sorting n records, Merge-sort has worst-case cost:
   a) O(log n).
   b) O(n).
   *c) O(n log n).
   d) O(n^2)
   e) O(n!)
   f) None of the above.
8. When sorting n records, Radix sort has worst-case cost:
   a) O(log n).
   b) O(n).
   c) O(n log n).
   d) O(n^2)
   e) O(n!)
   *f) None of the above.O(n*k+r*k)
9. When sorting n records with distinct keys, Radix sort has a lower bound of:
   a) Ω(log n).

b) Ω(n).
*c) Ω(n log n).
d) Ω(n^2)
e) Ω(n!)
f) None of the above.

10. Any sort that can only swap adjacent records as an average case lower bound of:
a) Ω(log n).
b) Ω(n).
c) Ω(n log n).
*d) Ω(n^2) 相邻记录交换的方法的时间复杂度是 n 方
e) Ω(n!)
f) None of the above.

11. The number of permutations of size n is: 大小为 n 的排列数为
a) O(log n).
b) O(n).
c) O(n log n).
d) O(n^2)
*e) O(n!)
f) None of the above.

12. When sorting n records, Selection sort will perform how many swaps in the worst case?
a) O(log n).
*b) O(n). 最坏要交换 n 次，注意交换次数和时间复杂度之间问的是哪个
c) O(n log n).
d) O(n^2)
e) O(n!)
f) None of the above.

13. Shell sort takes advantage of the best-case behavior of which sort?
希尔排序用了插入排序在最佳条件下的表现
*a) Insertion sort
b) Bubble sort
c) Selection sort
d) Shellsort
e) Quicksort
f) Radix sort

14. A poor result from which step causes the worst-case behavior for Quicksort?
哪个步骤的糟糕结果会导致 Quicksort 的最坏情况行为？
*a) Selecting the pivot 选枢轴是比较重要的
b) Partitioning the list
c) The recursive call

15. In the worst case, the very best that a sorting algorithm can do when sorting n records is:
在最坏的情况下，排序算法在排序 n 个记录时所能做的最好的事情是？
a) O(log n).
b) O(n).
*c) O(n log n).

d) O(n^2)
e) O(n!)
f) None of the above.

Chapter 8 File Processing and External Sorting

1. As compared to the time required to access one unit of data from main memory, accessing one unit of data from disk is:
   a) 10 times faster.
   b) 1000 times faster.
   c) 1,000,000 time faster.
   d) 10 times slower.
   e) 1000 times slower.
   *f) 1,000,000 times slower.
2. The most effective way to reduce the time required by a disk-based program is to:
   a) Improve the basic operations.
   *b) Minimize the number of disk accesses.减少磁盘访问次数
   c) Eliminate the recursive calls.
   d) Reduce main memory use.
3. The basic unit of I/O when accessing a disk drive is:
   a) A byte.
   *b) A sector. Sector/Block 是 I/O 最小单元，Cluster 是文件分配基本单元
   c) A cluster.
   d) A track.
   e) An extent.
4. The basic unit for disk allocation under DOS or Windows is:
   a) A byte.
   b) A sector.
   *c) A cluster.
   d) A track.
   e) An extent.
5. The most time-consuming part of a random access to disk is usually:
   *a) The seek. 寻道时间一般最长
   b) The rotational delay.
   c) The time for the data to move under the I/O head.
6. The simplest and most commonly used buffer pool replacement strategy is:
   a) First in/First out.
   b) Least Frequently Used.
   *c) Least Recently Used.
7. The C++ programmer's view of a disk file is most like:
   *a) An array.
   b) A list.

c) A tree.
d) A heap.
***考的可能性大
8. In external sorting, a run is:
*a) A sorted sub-section for a list of records. 记录的子部分排序
b) One pass through a file being sorted.
c) The external sorting process itself.
9. The sorting algorithm used as a model for most external sorting algorithms is:
a) Insertion sort.
b) Quicksort.
*c) Mergesort.
d) Radix Sort.
10. Assume that we wish to sort ten million records each 10 bytes long (for a total file size of 100MB of space). We have working memory of size 1MB, broken into 1024 1K blocks. Using replacement selection and multiway merging, we can expect to sort this file using how many passes through the file?
a) About 26 or 27 (that is, log n).
b) About 10.
c) 4.
*d) 2.

Chapter 9 Searching: Instructor's CD questions

1. Which is generally more expensive?
a) A successful search.
*b) An unsuccessful search.
2. When properly implemented, which search method is generally the most efficient for exact-match queries? 如果实施得当，对于精确匹配查询，哪种搜索方法通常是最有效的?
a) Sequential search.
b) Binary search.
c) Dictionary search.
d) Search in self-organizing lists
*e) Hashing
3. Self-organizing lists attempt to keep the list sorted by:
a) value
*b) frequency of record access
c) size of record
***考的可能性大
4. The 80/20 rule indicates that:
a) 80% of searches in typical databases are successful and 20% are not.
*b) 80% of the searches in typical databases are to 20% of the records. 80 的查询在 20 的记录上

c) 80% of records in typical databases are of value, 20% are not.

5. Which of the following is often implemented using a self-organizing list?
   *a) Buffer pool.
   b) Linked list.
   c) Priority queue.
6. A hash function must:
   *a) Return a valid position within the hash table.
   b) Give equal probability for selecting an slot in the hash table.
   c) Return an empty slot in the hash table.

***考的可能性大

7. A good hash function will:
   a) Use the high-order bits of the key value.
   b) Use the middle bits of the key value.
   c) Use the low-order bits of the key value.
   *d) Make use of all bits in the key value.
8. A collision resolution technique that places all records directly into the hash table is called:
   a) Open hashing.
   b) Separate chaining.
   *c) Closed hashing.
   d) Probe function.
9. Hashing is most appropriate for:
   a) In-memory applications.
   b) Disk-based applications.
   *c) Either in-memory or disk-based applications. 内存或基于磁盘的应用程序
10. Hashing is most appropriate for:
   *a) Range queries. 区域查找
   b) Exact-match queries.
   c) Minimum/maximium value queries.
11. In hashing, the operation that will likely require more record accesses is:
   *a) insert
   b) delete

Chapter 10 Indexing: Instructor's CD questions

1. An entry-sequenced file stores records sorted by: 条目顺序文件按照到达顺序存储
   a) Primary key value.
   b) Secondary key value.
   *c) Order of arrival. 到达顺序
   d) Frequency of access.
2. Indexing is:
   a) Random access to an array.
   *b) The process of associating a key with the location of a corresponding data record. 索引将键值和某个位置关联的过程
   c) Using a hash table.

3. The primary key is: 主键
   *a) A unique identifier for a record.
   b) The main search key used by users of the database.
   c) The first key in the index.
4. Linear indexing is good for all EXCEPT: 线性索引
   a) Range queries.
   b) Exact match queries.
   *c) Insertion/Deletion.线性索引不利于插入和删除
   d) In-memory applications.
   e) Disk-based applications.
5. An inverted list provides access to a data record from its:
   a) Primary key.
   *b) Secondary key. 倒置列表
   c) Search key.
6. ISAM degrades over time because:
   a) Delete operations empty out some cylinders.
   *b) Insert operations cause some cylinders to overflow.
   c) Searches disrupt the data structure.
7. Tree indexing methods are meant to overcome what deficiency in hashing?
   *a) Inability to handle range queries. 树索引克服哈希的无法范围查询
   b) Inability to handle updates.
   c) Inability to handle large data sets.
8. Tree indexing methods are meant to overcome what deficiency in linear indexing?
   a) Inability to handle range queries.
   *b) Inability to handle updates. 克服线性索引的无法处理更新
   c) Inability to handle large data sets.
9. Tree indexing methods are meant to overcome what deficiency in in-memory data structures such as the BST?
   a) Inability to handle range queries.
   b) Inability to handle updates.
   *c) Inability to handle large data sets. 克服内存中的无法处理大型数据集的不足
10. A 2-3 tree is a specific variant of a:
   a) Splay tree.
   *b) B-tree.
   c) BST.
   d) Trie.
11. The most important advantage of a 2-3 tree over a BST is that:
   a) The 2-3 tree has fewer nodes.
   b) The 2-3 tree has a higher branching factor.
   *c) The 2-3 tree is height balanced.
12. The B-tree:
   a) Extends the leaf nodes downward. B 树向上长
   *b) Extends the root node upwards.
13. The primary difference between a B-tree and a B+-tree is:

*a) The B+-tree store records only at the leaf nodes.

b) The B+-tree has a higher branching factor.

c) The B+-tree is hight balanced.

d) The B+-tree is smaller.

15. In real-life applications, the B+-tree will typically have about how many levels?

a) log n (base 2) for n > 1000.

b) 16

c) 8

*d) 4

e) 2

f) 1

Chapter 11 Graphs

1. Which is not the name for a standard graph traversal?

*a) Preorder.

b) Depth first.

c) Breadth first.

2. Depth-first search is best implemented using:

*a) A stack or recursion.深度优先遍历用栈和递归

b) A queue.

c) A tree.

3. Breadth-first search is best implemented using:

a) A stack or recursion.

*b) A queue. 广度优先遍历用队列

c) A tree.

***考的概率大

4. The goal of a topological sort is to:

a) Sort all of the graph vertices by value.

*b) Sort all of the graph vertices 点 so that each vertex 点 is listed prior to any others that depend on it.

拓扑排序是用来排序点的，每个点的前置点都在它的前面

c) Sort all of the graph vertices by distance from the source vertex.

5. A topological sort requires all of the following except:

a) The graph be directed.

b) The graph contain no cycles.

*c) The graph contain weights on the edges.

***考的概率大

6. The single-source shortest path problem can be used to:

a) Sort all of the graph vertices by value.

b) Sort all of the graph vertices so that each vertex is listed prior to any others that depend on it.

*c) Sort all of the graph vertices by distance from the source vertex.

单源最短路径用来排序每个从原点出发的点的距离

***考的概率大

7. Dijkstra's algorithm requires that vertices be visited in:

a) Depth-first order.

b) Breadth-first order.

*c) Order of distance from the source vertex. 迪杰斯特拉单元最短路径算法用来排序源点到每个点的距离

d) No particular order.

8. In the all-pairs shortest paths problem, a k-path is:

a) The shortest path to vertex k.

b) The sorest path that goes through vertex k.

*c) A path such that all intermediate vertices have index less than k.

9. For a graph of n nodes, no algorithm to solve the all-pairs shortest paths problem could possibly have a cost less than:

a) Ω(log n)

b) Ω(n)

c) Ω(n log n)

*d) Ω(n^2)

e) Ω(2^n)

10. Which is a good example of a greedy algorithm? 贪心算法

a) Floyd's all-pairs shortest path algorithm.

*b) Prim's minimal-cost spanning tree algorithm. 最小生成树

c) Depth-first search.

d) Topological sorting.

By Guozheng

ChengDu - In SiChuan University Jiangan Campus
At the Second Basic Experiment Building