**Chapter 6 Binary Trees: Instructor's CD questions**

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

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

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:

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

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）

\*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.

b) Use in dynamic in-memory applications.

c) Encryption algorithms.

d) It is never better than a dynamic representation.

**Chapter 7 Internal Sorting: Instructor's CD questions**

！！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, Mergesort 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) Omega(log n).

b) Omega(n).

\*c) Omega(n log n).

d) Omega(n^2)

e) Omega(n!)

f) None of the above.

(感觉答案错了啊)10. Any sort that can only swap adjacent records as an average case

lower bound of:

a) Omega(log n).

b) Omega(n).

c) Omega(n log n).

\*d) Omega(n^2)

e) Omega(n!)

f) None of the above.

11. The number of permutations of size n is:

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).

c) O(n log n).

d) O(n^2)

e) O(n!)

f) None of the above.

！！13. Shellsort takes advantage of the best-case behavior of which sort?

\*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?

\*a) Selecting the pivot

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:

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: Instructor's CD questions**

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.

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.

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.

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.

8. In external sorting, a run is:

\*a) A sorted sub-section for a list of records.

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?

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

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.

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.

！！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.

(错了吧)10. Hashing is most appropriate for:

\*a) Range queries.

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:

a) Primary key value.

b) Secondary key value.

\*c) Order of arrival.

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.

c) Using a hash table.

3. The primary key is:

\*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:

a) Range queries.

b) Exact match queries.

\*c) Insertion/Deletion.

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.

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.

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.

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.

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) 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.

14. 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