

CPSC 311: Analysis of Algorithms (Honors)

Exam 1

October 11, 2002

Name: \_\_\_\_\_

**Instructions:**

1. This is a closed book exam. Do not use any notes or books, other than your 8.5-by-11 inch review sheet. Do not confer with any other student.
2. Show your work. Partial credit will be given. Grading will be based on correctness, clarity and neatness.
3. I suggest that you read the whole exam before beginning to work any problem. Budget your time wisely—according to the point distribution.
4. There are 5 questions worth a total of 100 points, on 7 pages (including this page).

**DO NOT BEGIN THE EXAM UNTIL INSTRUCTED TO DO SO. GOOD LUCK!**

---

*Potentially Useful Formula:*

$$\sum_{k=0}^n x^k = \frac{x^{n+1} - 1}{x - 1}$$

1. (15 pts total)

(a) (8 pts) Use the master theorem to solve the recurrence

$$T(n) = 4T(n/2) + n^2\sqrt{n}$$

assuming  $T(n)$  is constant for  $n \leq 2$ .

(b) (7 pts) Now verify the answer you got in part (a) using induction. Be sure to check both the upper bound ( $O$ ) and the lower bound ( $\Omega$ ).

2. (20 pts total) Consider the following algorithm to merge  $k$  **sorted** lists into one sorted list. Let  $n$  be the total number of elements in all the lists.

```
K-Way-Merge(L1,L2,...,Lk)
  initialize a min-heap H to empty
  for i := 1 to k do
    let x be the smallest element in Li
    remove x from Li
    insert (x,i) into H, using x as the key
  endfor
  for j := 1 to n do
    (x,i) := extract-min(H) /* remove smallest element from H */
    M[j] := x
    if Li is not yet empty then
      let y be the smallest element in Li
      remove y from Li
      insert (y,i) into H, using y as the key
    endif
  endfor
  return M
end /* K-Way-Merge */
```

(a) (10 pts) What is the asymptotic worst-case running time of this algorithm in terms of both  $n$  and  $k$ ? Justify your answer.

(b) (5 pts) Consider a modified mergesort that splits the array into  $k$  segments, sorts each segment recursively and then uses  $k$ -way merge. what is the recurrence describing the worst case running time of this algorithm, as a function of both  $n$  and  $k$ ? You do not have to solve the recurrence.

(c) (5 pts) What is a good choice for  $k$  in part (b) and why?

3. (20 pts total) Consider the following comparison-based selection algorithm, New-Select, to find the  $i$ -th smallest element of an unsorted array  $A$ . Let  $\text{Mystery-Partition}(A, p, r)$  be a procedure that partitions the segment of array  $A$  between indices  $p$  and  $r$  around its median and returns the index of the median. New-Select is recursive; the top level call is  $\text{New-Select}(A, 1, n, i)$ , where  $n$  is the length of  $A$ .

```
New-Select(A,p,r,i):
  if p = r then return A[p] endif
  q := Mystery-Partition(A,p,r)
  k := q - p + 1
  if i = k then
    return A[q]
  elseif i < k then
    return New-Select(A,p,q-1,i)
  else /* i > k */
    return New-Select(A,q+1,r,i-k)
  endif
end /*New-Select*/
```

(a) (15 pts) Assume the worst-case running time of  $\text{Mystery-Partition}$  is  $O(\log n)$ , where  $n$  is the size of segment of the array being partitioned. Derive the recurrence relation for the asymptotic running time of  $\text{New-Select}$ . Solve the recurrence to obtain the best (i.e., smallest) upper bound you can. Clearly state any assumptions you make.

(b) (5 pts) Can there exist such a procedure as  $\text{Mystery-Partition}$  with the claimed running time? Explain.

4. (20 pts total)

(a) (6 pts) What is the desired relationship between disk block size and the parameter  $t$  of a  $B$ -tree, and why?

(b) (7 pts) Is the following tree a B-tree with minimum degree 3? Why or why not?

(c) (7 pts) If necessary, adjust the figure (without changing the keys) so that it is a B-tree with minimum degree 3. Draw the result of inserting the key 9 into it. Use the insert algorithm given in class and the text.

5. (25 pts total) Consider building a computerized card catalog for a library. The catalog is to hold an entry for every book in the library. Each entry must contain, at a minimum, the title, author, and publisher of the book. For simplicity, assume that the library has at most one book by any particular author and that no two different authors represented in the library have the same name. The system must support the following operations by users:

- (i) add an entry when a new book is obtained by the library,
- (ii) answer queries concerning whether the library has a book by a particular author,
- (iii) remove an entry when a book is lost or destroyed.

(a) (8 pts) Discuss how to use each of the following data structures (separately) to build this system: binary search tree or hash table. In particular, describe how each library system operation (i-iii) can be implemented in each data structure. *You may refer to the known operations on the data structures without going into the details of how they are implemented.*

Using a binary search tree

What should be the key:

(i) how to add an entry:

(ii) how to answer a query:

(iii) how to remove an entry:

Using a hash table (your choice of chaining or open addressing)

What should be the key:

(i) how to add an entry:

(ii) how to answer a query:

(iii) how to remove an entry:

(b) (14 pts) Discuss the advantages and disadvantages of using each of the two data structures (from (a)) to build this system. Your discussion should address space usage, and worst-case time and expected time for the three operations (i)-(iii).

comparison of space usage:

comparison of worst case times for

(i):

(ii):

(iii):

comparison of expected times for

(i):

(ii):

(iii):

(c) (3 pts) Now suppose that the computerized card catalog needs to be extended to support an additional operation:

(iv) print out a list of all books in the library in alphabetical order by author's last name.

Which data structure, binary search tree or hash table, is better for this operation? Discuss both worst-case and expected time.