CPSC 320 Sample Final Examination
December 2003

Name: _____
Student ID: _____

– You have to write the 8 questions on this examination. A total of 110 marks are available.

– **Justify all of your answers, except if the question says not to.**

– No notes or electronic equipment are allowed, except for **one** 8.5×11 **sheet of paper, handwritten**.

– Keeps your answers short. If you run out of space for a question, you have written too much.

– The number in square brackets to the left of the question number indicates the number of marks allocated for that question. Use these to help you determine how much time you should spend on each question.

– Use the back of the pages for your rough work.

–

Question	Marks
1	
2	
3	
4	
5	
6	
7	
8	
Total	

UNIVERSITY REGULATIONS:

– No candidate shall be permitted to enter the examination room after the expiration of one half hour, or to leave during the first half hour of the examination.

– CAUTION: candidates guilty of any of the following, or similar, dishonest practices shall be immediately dismissed from the examination and shall be liable to disciplinary action.

1. Making use of any books, papers or memoranda, electronic equipment, or other memory aid devices, other than those authorised by the examiners.
2. Speaking or communicating with other candidates.
3. Purposely exposing written papers to the view of other candidates. The plea of accident or forgetfulness shall not be received.

– Smoking is not permitted during examinations.

[12] 1. Answer each of the following questions with *true* or *false*. Give a *short* justification for each of your answers.

[4] a. $6^n \in O(5^n)$

[4] b. $1.5^n + n^2 \in O(1.5^n + n \log n)$

[4] c. For values of n that are smaller than approximately $n = 1000$, algorithm Select is slower than the naive $O(n \log n)$ algorithm (the one that sorts the array using quicksort or mergesort, and then returns the requested element in constant time).

[12] 2. In class, we discussed the following algorithms, all of which take an array A as parameter.

(a) Insertion sort
(b) Bucket sort
(c) Radix sort
(d) Quicksort
(e) Randomized Quicksort
(f) Select
(g) Randomized Select

For each of the statements below, list *all* of the algorithms listed above for which the statement is true (you do not need to justify your answer(s)).

[3] a. If we change the ordering of the elements in the array A, then the algorithm's running time may change by more than a constant factor.

[3] b. If we change the way in which the elements of A are distributed inside their range, without changing their relative ordering, then the algorithm's average-case running time may change by more than a constant factor.

[3] c. Its average-case running time $T_a(n)$ is much better than its worst-case running time $T_w(n)$, that is, $T_a \in o(T_w)$.

[3] d. It runs *in place*, that is, the amount of storage used when the algorithm is executed (not counting the array A) does not depend on the length of A.

[14] 3. Randomization

[3] a. What is the main advantage of `Randomized Quicksort` over `Quicksort`?

[3] b. Name one reason why the choice of a random level for each node in a Skip List helps us achieve a good average case running time.

[3] c. Write a function `Randomized Binary Search` that performs a version of binary search that uses randomization. You may assume the existence of a function `Random(i,j)` that returns a random integer x uniformly distributed in the range $[i, j]$.

[2] d. What is the worst-case running time of the function `Randomized Binary Search` that you wrote in part (c)?

[3] e. Does `Randomized Binary Search` have any advantage over `Binary Search`? Explain why or why not.

[9] 4. For each of the following recurrence relations, determine whether or not the Master Theorem discussed in class can be used. If it can be used, apply it to derive the solution of the recurrence relation using O notation. If the Master Theorem can not be used, explain why briefly.

[3] a. $T(n) = \begin{cases} 2T(\lfloor\sqrt{n}\rfloor) + n & \text{if } n \geq 2 \\ 1 & \text{if } n \leq 1 \end{cases}$

[3] b. $T(n) = \begin{cases} 9T(n/3) + 2n^2 & \text{if } n \geq 3 \\ 1 & \text{if } n \leq 2 \end{cases}$

[3] c. $T(n) = \begin{cases} 4T(\lfloor n/2 \rfloor) + n^{2+\text{odd}(n)} & \text{if } n \geq 2 \\ 1 & \text{if } n \leq 1 \end{cases}$ where $\text{odd}(n) = \begin{cases} 1 & \text{if } n \text{ is odd} \\ 0 & \text{if } n \text{ is even} \end{cases}$

[11] 5. Consider the following function:

```
Algorithm HappyNewYear(A, p, r)
//
// A is an array, p and r are positions in the array.
//
mid ← ⌊ (p + r)/2 ⌋
x ← Christmas(A[p]) + Christmas(A[mid]) + Christmas(A[r])
if (n ≥ 2) then
    x ← x + HappyNewYear(A, p+1, r-1)
    x ← x + HappyNewYear(A, p+1, r-1)
    x ← x + HappyNewYear(A, p+1, r-1)
    x ← x + HappyNewYear(A, p+1, r-1)
return x
```

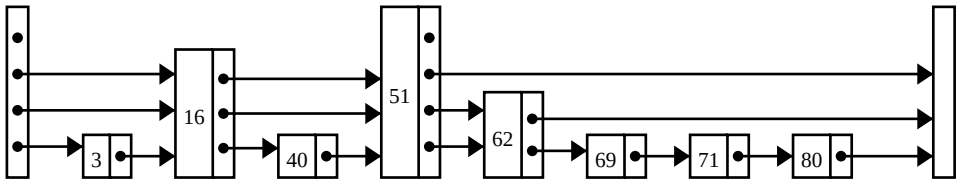
[4] a. Let $C(n)$ be the number of times that function `Christmas` will be called during the execution of the call `HappyNewYear(A,p,r)`, where $n = r - p + 1$. Write a recurrence relation for $C(n)$.

[7] b. Prove using the substitution method that the solution of the recurrence relation you gave in part (a) is in $O(2^n)$.

[20] 6. Skip Lists

[4] a. When a new key is inserted in a skip list, we choose the level of its node randomly, where for each i the probability that the node has level $\geq i$ is p^{i-1}. Explain briefly how this is done. Recall that p is a constant between 0 and 1 that is chosen beforehand (it is easier to think of $p = 1/2$).

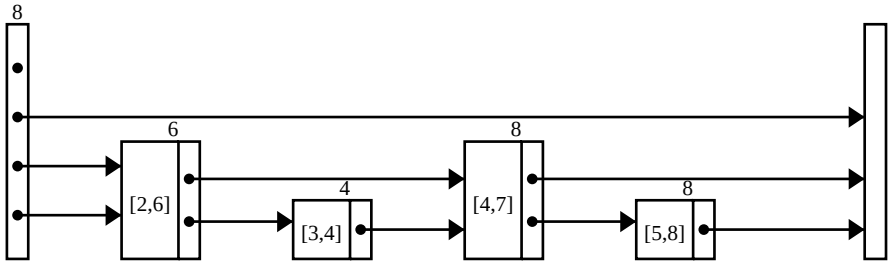
[3] b. Consider now the following skip list:



Show the pointers *followed* by the Search algorithm discussed in class, when the key to search for is 73. Assume that the current value of `MaxLevel` is 3.

[5] c. Show the skip list obtained after the following sequence of operations has been performed on the skip list from part (b): inserting key 44 with level = 3, deleting key 80, deleting key 16, inserting key 72 with level = 2, and deleting key 44. Hint: it is faster if you do them all at once.

[8] d. Suppose now that we have a skip list whose keys are intervals, sorted according to their left endpoint, like the following skip list (its current MaxLevel is 3):



We have augmented that skip list, by storing in each node N the maximum right endpoint $\max[N]$ of the intervals stored between N (included) and the next node whose level is $\geq$ level(N) (not included). For instance, the second node in the figure contains the value 6 because 6 is the maximum right endpoint amongst the two intervals $[2, 6]$ and $[3, 4]$

Explain how you would update the values $\max[N]$ stored in the skip list when a new key is inserted.

[15] 7. The CEO of a software company wants to keep his developers happy by giving them a bonus, but does not want to spend too much money. He thus wants to select as few developers as possible (these will get a bonus, and be happy), chosen so that every other developer likes at least one of those that get a bonus (and hence will be happy for him/her).

The CEO's problem can be modeled using a graph $G = (V, E)$: each node in the graph is a developer, and an edge (u, v) means that developer u likes developer v. The CEO is looking for a minimum subset W of the set V of vertices, where for every vertex $u \notin W$, there is an edge (u, w) where $w \in W$. This is called a *minimum dominating set* for the graph G.

[8] a. Write a greedy algorithm that finds a subset W of V (it does need to be the subset with the fewest elements possible) with that property. Your mark will depend in part on the criterion you use to decide which vertex to add to W. You do not need to give pseudo-code, but you should indicate what data structure you are using to store the vertices that your algorithm has not dealt with yet.

[3] b. What is the running time of the algorithm you described in part (a)?

[4] c. The minimum dominating set problem does not satisfy the greedy choice property. Knowing this, what can you conclude about the algorithm you gave in part (a)?

[17] 8. Bob and Maria have been arguing about where to have lunch after the CPSC 320 final examination, and decide to solve the decision by playing a sequence of Poker games. The first person to win n games gets to choose where they will have lunch. Maria is a slighly better poker player, and has a 51% chance of winning any individual Poker game.

Let $P(i, j)$ be the probability that Bob will be the first person to win n games, given that he only needs to win another i games (that is, Bob has won $n - i$ games so far), and that Maria only needs to win another j games (that is, she has won $n - j$ games so far). It can be proved that

$$P(i, j) = 0.49P(i - 1, j) + 0.51P(i, j - 1) \quad (1)$$

Bob is worried about his chances of winning, and asks you to use dynamic programming to compute the probability $P(n, n)$ that he will win n games before Maria does.

[3] a. State how big a table you need to answer Bob's question using dynamic programming, and the meaning of each entry in the table.

[3] b. Give one possible order that you can use to compute the entries in the table from part (a).

[3] c. List all of the base cases that will be needed when you are computing the entries in the table from part (a). That is, list the cases where you can not use equation (1) to compute $P(i, j)$, and the value of $P(i, j)$ for each of them.

[8] d. Write pseudo-code for an algorithm that uses dynamic programming to determine the probability that Bob will win n Poker games before Maria does.