# Midterm
# CSC 172
# June 22, 1998

Write Your name legibly on the blue book. Choose problems as indicated. The recommended strategy is not to spend more than the time indicated time on any question. This exam is open book and open notes, calculators are allowed. If you do more than the $n$ specified number of problems, the best $n$ will count. Meliora.

1. (10 min) Prove $\sum_{i=1}^{n} \frac{1}{2^i} = 1 - \frac{1}{2^n}$ by induction.

Induction on n.

Basis : n=1, $\sum_{i=1}^{1} \frac{1}{2^i} = \frac{1}{2^1} = 1 - \frac{1}{2^1} = \frac{1}{2}$

Induction : Assuming $\sum_{i=1}^{n} \frac{1}{2^i} = 1 - \frac{1}{2^n}$ we show $\sum_{i=1}^{n+1} \frac{1}{2^i} = 1 - \frac{1}{2^{n+1}}$

$$\sum_{i=1}^{n+1} \frac{1}{2^i} = \sum_{i=1}^{n} \frac{1}{2^i} + \frac{1}{2^{n+1}} = 1 - \frac{1}{2^{n+1}}$$

substituting

$$\sum_{i=1}^{n+1} \frac{1}{2^i} = 1 - \frac{1}{2^n} + \frac{1}{2^{n+1}} = 1 + \frac{1}{2^{n+1}} - \frac{2}{2^{n+1}} = 1 - \frac{1}{2^{n+1}}$$

2. (10 min) An arithmetic progression is

$$s = \sum_{k=0}^{n} a + kd$$

Where a is the first term and d is the common difference between terms
[e.g. s = 4+(0*3)+(4+(1*3))+(4+(2*3))+(4*(3*3))+... = 4+7+10+13+...]
One formula for the sum s is :

$$s = \frac{n+1}{2}(2a + nd)$$

Prove this formula is correct by mathematical induction on $n$.

Basis: n=0, s = 2a/2 formula = a one term sum

Induction :

show $s = \sum_{k=0}^{n+1} (a + kd) = \sum_{k=0}^{n} (a + kd) + a + (n+1)d = \frac{n+2}{2}(2a + (n+1)d)$

substituting the induction hypothesis

$$s = \sum_{k=0}^{n+1}(a+kd) = \frac{n+1}{2}(2a+nd) + a + (n+1)d = \frac{n+2}{2}(2a+(n+1)d)$$

both second and third terms expand to

$$\frac{dn^2 + 2an + 3nd + 4a + 2d}{2}$$

**Do one of the following 2 problems ( 3 or 4):**

3. (12.5 min) We say that $n_0$ and c are witnesses to the fact that f(n) is O(g(n)) if for all $n \geq n_0$ it is true that $f(n) \leq cg(n)$ (i.e. the big-oh proof "works" with $n_0$ and c as the constants).

a) If $n_0 = 1$, what is the smallest value of c such that $n_0$ and c are witnesses to the fact that $(n+2)^2$ is $O(n^2)$?
b) If c=5, what is the smallest nonnegative integer $n_0$ such that $n_0$ and c are witnesses to the fact that $(n+2)^2$ is $O(n^2)$?
c) If $n_0$=0, for what values of c are $n_0$ and c witnesses to the fact that $(n+2)^2$ is $O(n^2)$?

a) c=9,
b) $n_0$=2
c) There are no such values for c.

4. (12.5 min) Prove that $T(n)=1024n$ is $O(n \log n)$.
(Hint : You need to demonstrate a $c$ and an $n_0$ such that ...)

We want to show that there is c and n0 such that for any n>n0, 1000n<=cnlogn for n>n0. So just let c=1 and we cans solve this inequality for the point at which the two functions cross: for n bigger than that value nlogn>1024n. So the point of crossover is nlogn=1000n, or logn=1024, or n0=b1024 for some base b (varying b only varies the value of n0 by a constant, and that can be absorbed in c.). 1024 is a pretty big exponent however you cut it, which shows you that nlogn grows not that much faster than n.

**Do one of the following two problems (5 or 6) :**

5.(12.5 min) Give the time T(N) for the following code samples:

A. (4 min)

```
for (I=0;I<N*N;I++)
     for (J=0;J<N*N*N;J++)
          x++;
```

B. (4 min)

```
inc = 1;
for (I=0;I<N;I++) {
     inc *= 2;
```

```
            if (inc >= N) break ; // exit!
            for (j=0;j<N;j+=inc)
                  x++;
      }
```

C. (4.5 min)

```
      inc = 0 ;
      for (I=0;I<N;I++) {
            inc++ ;
            for (j=0;j<N;j+=inc)
                  x++;
      }
```

A) N^5
B) N/2+N/4+… , so O(N)
C) N+N/2+N/3+…, so O(NlogN)

6. (12.5 min) What is the average running time of the following program sample? In particular, how may multiplies will be performed on the average (in terms of t), if the program is fun lots and lots of times? The Rand() function returns a (uniformly distributed) random integer.

```
while(TRUE) // loop forever unless break out
{
      if (Rand() > Rand())

            for (j=0;j<t;j++)    //  do t multiplies
                  z = 100 * 100; //   and repeat the loop

            else break ;         // break out of loop
{
```

The running time is the amount of work done in each iteration (t multiplies) times the expected number of times through the loop. On the first attepm the test at the if statement fails ½ of the time and succeeds ½ the time, so the probabilith of getting through the loop once is ½. On the second try you have anoother 50-50 chance of getting through the test. You get through twice if you get through the first time and the second time on an average of (1/2)(1/2)=(1/4) the time. It's like tossing a fir coin to see if you get to count antother t multiplications. The sum of these probobilites is (1/2+1/4+…+1/2^n)=(1-1/2^n) [as we saw in problem #1], or in the limit, and the amount of multiplies is t times that.

7. (10 min) Combinatorics.

   A. (3 min) How many ways can 6 men and 6 women be seated in a row if any person can sit next to any other person?
   B. (3 min) How many ways can 6 men and 6 women be seated in a row if the men must occupy alternate seats?
   C. (4 min)How many ways can the 12 individuals (with no seating restrictions) be seated around a round table, (so rotations of an arrangement count as the same)?

   A. 12!
   B. 2*(6!*6!)=12*6!*5!
   C. 11!=12!/12

8. (12.5 min) (Write answers in blue book) Consider the recurrence :

   Basis : T(1) = 0.

   Induction : $T(n) = \frac{1}{2}T(n-1) + 1$ for integers n > 1 .

   Note that since there are no symbolic constants, this is not a "big-oh" problem; we can solve for T(n) exactly.

   A) What are the 5 initial values of T(n)?
      T(1)=0 ; T(2)=1; T(3)=1.5; T(4)=1.75; T(5)=1.875 .

   B) Expand the inductive rule so T(n) is expressed in terms of T(n-2).
      T(n)= (T(n-2)/4) + 3/2
   C) Express T(n) in terms of T(n-3).
      T(n)=(T(n-3)/8) + 7/4
   D) What is the general pattern? That is, Express T(n) in terms of T(n-i).

$$T(n) = \frac{T(n-i)}{2^i} + \frac{2^i - 1}{2^i} = \frac{T(n-i)}{2^i} + 2 - 2^{-(i-1)}$$

   E) For what value of i can we eliminate T(n-i) from the expression? i = n-1
   F) Use your answer to (E) to express T(n) as a function of n alone
      (i.e. without any terms involving the function T).
      $T(n) = 2 - 2^{-(n-2)}$ .

9. (10 min) Write a recursive C++ function **oddSum** that takes a list of integers and returns the sum of the even-valued elements minus the sum of the odd-valued elements. For extra credit, you may show the iterative solution.

For example : **oddSum** called on the list (10,5,12,8,7) should return the value 10+12+8-5-7 = 18.

You may assume the data structures developed in class. (i.e. that list->getNext() returns the next cell in the list and that list->getElement() returns the value stored at a given cell. Templates are not required. You may also use the fact that the Boolean expression (list->getElement() %2) will evaluate to 1 if the stored value is odd and 0 if the stored value is even.

```
int oddSum(Cell * list1) {
      if (list1 == NULL) return 0 ;
      else if (list1->getElement() % 2)
            return oddSum(list->getNext()) - list1->getElement();
             else
            return oddSum(list1->getNext()) + list1->getElement();
}
```

**Iterative**

```
int oddSum(Cell * list1) {
      int sum = 0 ;
      Cell * temp = list1 ;
      while (temp != NULL) {
        if (temp->getElement() % 2)
            sum -= temp->getElement();
             else
            sum += temp->getElement();
         temp = temp->getNext();
      }
      return sum ;
}
```

# Final Exam

CSC 172

19 Dec 98

Write your **NAME** legibly on the bluebook. Work all problems. Best strategy is not to spend more than the indicated time on any question (minutes = points). Part A deals with material through 19 Nov 98, *i.e.* through the second midterm. The minimum of your midterm grades will be replaced by the maximum of it and your score on Part A. Open book, open notes.

## Part A (75 mins)

### A1. Structural Induction (15 mins)

You are given a binary tree (represented as [`int contents, tree *leftchild, tree *rightchild`]). The tree is unstructured: the contents of parents and children have no predictable size relation.
(a) Write a recursive (pseudocode) program to find the minimum contents in the tree. Don't use any global variables!
(b) Prove your program works by structural induction.

### A2. Sorting, Logs, and ! (10 mins)

(a) (2 mins) If you are in the sorting business, why do you care about the quantity $\log(n!)$?

(b) (8 mins) Use Stirling's formula,

$$n! \approx (n/e)^n \sqrt{2\pi n},$$

to give a precise estimate of $\log(n!)$.

### A3. Re-hashing (15 mins)

You have a hash table (open hashing, separate chaining) whose entries are simple unordered linked lists of records. It has N hash buckets, and its load factor is M.
(a) (3 mins) How long (on average, in terms of N and M) does it take to look up a key in this table?

Now you re-hash this whole hash table into an open-hashing table that has N/10 hash buckets and whose table entries are AVL trees of records.
(b) (3 mins) How long (on average, in terms of N and M) does it take to look up a key in this new hash table?
(c) (9 mins) How long (in terms of N and M) did it take you to build the new hash table (including finding all the elements in the first table and inserting them into the second)? [Hint: $\int \log x dx = x \log x - x$].

## A4. Running Time (10 mins)

What is the "running time" of this pseudocode loop? *I.e.* as N gets large, what will be the value of kount as a function of N when the nested loop terminates?

```
int i,j;
for (i = 0; i < N; i++)
      for (j = 0; j < i; j++)
       sub1(N);

void sub1 (int N)
{
int i;
for (i = 0; i < N; i++)
  if (i*i >N) return;
  else sub2(N);
}

void sub2(int N)
{
int i;
for (i = 1; i < N; i*=2)
   kount++;
}
```

## A5. Relational Algebra (15 mins)

For the relations shown, evaluate this relational algebra expression.

$$\sigma_{A>3}(\pi_{A,G}(R \overset{\bowtie}{_{C=F}} S))$$

| A | B | C | D |
|---|---|---|---|
| 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8 |
| 5 | 5 | 5 | 5 |
| 1 | 3 | 6 | 3 |

| E | F | G |
|---|---|---|
| 7 | 7 | 4 |
| 1 | 6 | 0 |
| 5 | 4 | 5 |
| 6 | 3 | 0 |

Relations R (left) and S (right)

# First Midterm

CSC 172

5 Oct 99

Write your **NAME** legibly on the bluebook. Work all problems. Best strategy is not to spend more than the indicated time on any question (minutes = points). Open book, open notes.

1. Induction: (10 mins) An arithmetical progression starts at a first term $a$. Each succeeding term is formed by adding a fixed quantity $b$ to the previous term, so the series goes

$$a, a+b, a+2b, ...., a+(n-1)b$$

The formula for the sum of such a series is

$$\sum_{i=0}^{n-1}(a+bi) = n(2a+(n-1)b)/2.$$

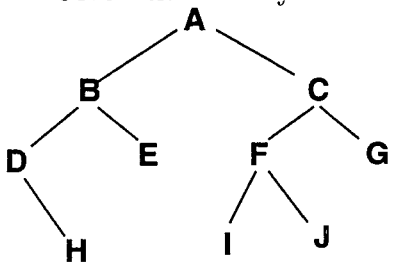
Prove this formula by induction on $n$.

2. Running Time (5 mins) What is the "running time" of this nested loop? Give $T(n)$ as a function of $n$, where $T(n)$ counts the number of times the innermost statement is executed. I want the most accurate bound you can give. Clearly $T(n)$ is $O(n^4)$ but that's not interesting.

```
Tn = 0;
   for(i = 1; i<=n; i++)
      for(j = 1; j<=n; j*=2)
            Tn++;
```

3. Big-Oh: (10 mins)
Prove that $2^{n+5}$ is $O(2^n)$.

4. Tree Traversal (5 mins)
Given this binary tree:




What is the output of an inorder traversal that prints each node's (single-character) contents?

5. Tree Traversal (10 mins)
The algorithm below is run on the tree of the previous problem. aQueue is a queue class with obvious methods, root refers to the root of the tree. What is the output? How is it related to recursive tree traversals?

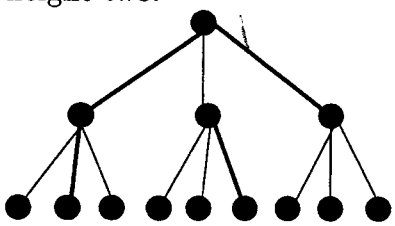
```
public void Mystery (TeeeNode root) {
    TreeNode aNode;
    aQueue.clear();
    aQueue.enQueue(root);
    while (! (aQueue.isEmpty()))
        {
            aNode = (TreeNode) aQueue.deQueue();
            aNode.printContents();
            if (aNode.getLeftChild() != null)
                {
                    aQueue.enQueue(aNode.getLeftChild());
                }
            if (aNode.getRightChild() != null)
                {
                    aQueue.enQueue(aNode.getRightChild());

                }
        }
}
```

6. Logs and Exps (10 mins) Simplify this expression as far as possible using the rules for exponentiation and logarithms

$$log\left[\frac{(x^a)^b}{y^{c+d}}\right]$$

7. Combinatorics (10 min): You're in the Chatahoochnapatawfa County Jail making license plates. The rules are that each license number starts with a group of three letters (from A to Z), in which none of the letters is repeated (so XLX is not allowed, for example). After that there are four numbers; the first number of the four must be 0,1,2,3,4, or 5, and the last two numbers cannot be "99" (so 6075 is no good, nor is 0599, but 1289 is fine). How many possible acceptable plate numbers are there?

8. Trees And Combinatorics (15 min): In a *full tree*, every interior node down to some level has all its children present. Thus all the leaves are in the next level. Here is a complete ternary tree of height two:



a) (5 min) How many nodes are there in the complete ternary tree of height $h$? Write down and justify an expression. Simpler is better, but correctness is most important. In particular, leaving your result as a summation is fine.

b) (10 min) A *path* is a connected set of arcs *from ancestor to some descendent*, but not *vice-versa*. In the tree above the dark arcs show one path of length two and two paths of length one. There is no path of length 3 since paths don't go "up".

How many paths are there in the complete ternary tree of height $h$, including all possible lengths and starting at all possible positions in the tree. Write down and justify an expression. Simpler is better, but correctness is most important. In particular, leaving your result as a summation is fine.

ANSWERS

# Final Examination

CSC 172

18 December 1999

*This is a two-hour (120 minute) exam,* ***NEW! with a strict time limit.*** *Write your* ***NAME*** *legibly on the bluebook. Work all problems. Best strategy is not to spend more than the indicated time on any question (minutes = points). Open book, open notes.* 1. Induction (10 mins)

For the Fibonacci sequence $f(i) = 1, 1, 2, 3, 5, 8, ...$ for $i = 1, 2, 3, 4, 5, 6, ...$:
Prove by induction that for $n \geq 1$,

$$f(n+1)f(n+2) = f(n)f(n+3) + (-1)^n.$$

Answer: for the basis at $n = 1$ we check

$$f(1+1)f(1+2) = 1 \times 2.$$

Also

$$f(1)f(1+3) = 1 \times 3 = 3$$

and $2 = 3 + (-1)^1$, so the basis is OK.
Induction Step:

$$f(n+1)f(n+2) = (f(n-1) + f(n))f(n+2) = f(n-1)f(n+2) + f(n)f(n+2).$$

By induction, this sum is

$$f(n)f(n+1) + f(n)f(n+2) - (-1)^{n-1},$$

or

$$f(n)(f(n+1) + f(n+2)) + (-1)^n = f(n)f(n+3) + (-1)^n.$$

2. Structural Induction (15 mins)
(a) (12 mins.) Prove by induction on the number of nodes in the tree that: The leaves of the tree (not the nodes, the leaves) are visited in the same order in a recursive inorder and a recursive preorder traversal.
(b) (3 mins.) What can you then say about the order in which a recursive postorder traversal visits the leaves?

Answer:
(a) Proof by induction on the number of tree nodes. Basis: if $n = 0$, nothing to prove.

Induction: if tree not empty, then preorder first visits the root, which is not a leaf, so we can ignore this step. Both traversals will next traverse the left subtree; by induction, leaves will be visited in the same order. Inorder will next visit the root, which is not a leaf, so we ignore that. Finally both traversals do the right subtree, which by induction visits the leaves in the same order. (b) By symmetry, the postorder traversal also visits leaves in the same order.

3. Recurrence Relation (15 mins)

(a) Solve this recurrence as completely as you can:

$$T(1) = 1 \quad (1)$$
$$T(n) = 2T(n-1) + n, \quad n \geq 2 \quad (2)$$

Ans: for instance by repeated substitution you get something like $T(n) = 2^n \times 1 + \sum_{i=0}^{n} 2^i(n-i)$. The sum splits into $n\sum 2^i$, (which equals something like $n(2^{n+1}-1)$ ) minus $\sum i2^i$, and the latter sum is about $(n-1)2^n$ if I read my integral tables right. Anyway the whole smash is exponential and in fact is $O(2^n)$, although this is not a proof of that!

(b) Write a small function whose running time exhibits this performance. Say it has arguments (`int startNdx`, `int howMany`) and references a global array `Array`.

```
void recurfunc(int startNdx, int howMany )  \\ initally call with (1,N).
  {
   if (howMany == 1) return;
   recurfunc(1, howMany-1);
   recurfunc(2, howMany-1);
   for (int i = startNdx; i<=howMany; i++) print  Array[i];
}
```

4. Graph Combinatorics (10 mins)

(a) A completely-connected graph has each pair of nodes is connected by an arc. In such a graph with 100 nodes, there is exactly one clique of size 100. Write an expression for the number of cliques of size 37.

Ans: any 37 nodes work, so (100 choose 37).

(b) In a completely-connected graph of with N nodes there is just one path of length 1 from any particular node A to another particular node B. How many simple paths (no cycles allowed!) of length 2 are there from A to B? Of length 3? Of length M? Informally justify (or formally prove) your answer.

Ans: You can put any of the N-2 remaining nodes first in the path to get paths of length 2. For each of those, you can put any of the remaining N-3 nodes first in the path. So I make it for length M, the product (N-2)(N-3)...(N-M). Another way to think of it: after picking A and B, you have (N-2) ways to pick the first node on the path, (N-3) ways to pick the second, etc...

5. Cacheing (15 mins)

(a) (5 mins) A memory cache between the CPU and main memory can save execution time in running a program. Why? That is, what principle(s) of program execution in actual programs does the cache exploit?

Ans: Locality of memory access, both temporal and spatial.

(b) (5 mins) Least-recently-used (LRU) and First-in First-out (FIFO) are names of strategies for finding the cached item to remove (and replace with a new, requested, non-cached item) when the cache is full. What data structures are natural for implementing these two strategies?

Ans. LRU with a (maximum) priority queue whose key is the time of last use, so the least-recently-used item is at the top of the queue.

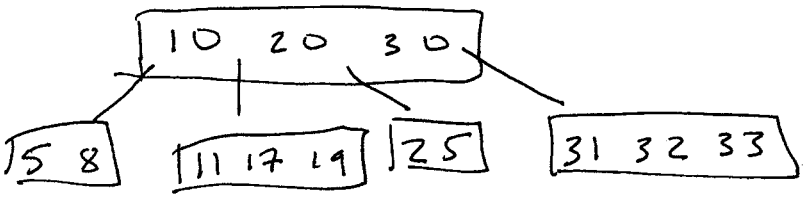
FIFO with a queue; the item brought in first is the first to be evicted.

(c) (5 mins) FIFO and LRU are approximations to an "Optimal Replacement ALgorithm" ORAL. What **is** the optimal replacement algorithm and why is it impractical to implement?

Optimally, you'll delete the item whose first use is farthest in the future. Knowing this involves predicting the course of your execution, however, which involves supernatural powers or actually running the program to see how it goes....

6. (2,3,4) Trees (10 mins)

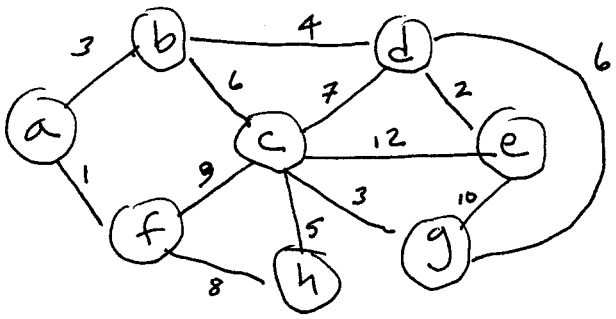
Insert 40 into this (2,3,4) tree.




Ans: see attached.

7. Spanning Tree (10 mins)

Find a minimal spanning tree for this graph by Prim's or Kruskal's algorithm and say which algorithm you used. Show the stages in your work.




Ans: for Prim, add edges in this order: af, ab, bd, de, bc, cg, ch. For Kruskal, af, de, cg or ab, bd, ch bc. or dg

8. Hashing and Searching (15 mins)

You have a closed hash table with a load factor $\lambda = 11/12$. Its size is 1009. The empty locations are uniformly and randomly distributed throughout the table (technically there are no clusters).

You also have an AVL tree: comparing a (query or insertion) key with an AVL tree node takes the same time as a hash-table probe and an AVL tree rotation takes twice as long as a hash-table probe.

(a) (10 mins) To insert a new item into each of these structures takes a certain amount of time. About how many nodes are in an AVL tree whose *worst-case* insertion time is equal to the *average-case* insertion time of the hash table? If the AVL tree has more nodes than this, does worst-case insertion go faster or slower than average-case hash insertion?

Ans. Average no. of probes is 12, so in the worse case an AVL tree of depth 4 (with 16 nodes) can take this long. Insertion in bigger AVL trees will run slower than hash table insertion, not faster.

(b) (2 mins) Does probing strategy matter?

Ans. with no clusters, no difference.

(c) (3 mins) How does your answer to (a) change if the hash-table size goes to 100,013 (holding $\lambda$ constant?)

Ans. No, only the load factor makes a difference.

9. Heaps (10 mins)

An array contains the integers

7 1 9 2 6 3 4 5 8

in sequence. Consider this array as a binary heap.

(a) (1 min.) Give the binary tree represented by the array.

(b) (4 min.) "Heapify" the tree, by performing bubble-ups or bubble-downs, until it obeys the "max-heap" property with the maximum element at the top. You don't need to show your work, just give the final heap if you like.

(c) (6 min.) Still using the binary tree representation, show the stages of heapsort applied to the heap.

Ans: see attached.

10. Optimism amd Quicksort (5 mins)

Can quicksort be modified to give $O(n)$ *best-case* time complexity? If not, why not? If so, how?

Ans. Sure, add an O(n) test to check if the input array is already sorted.

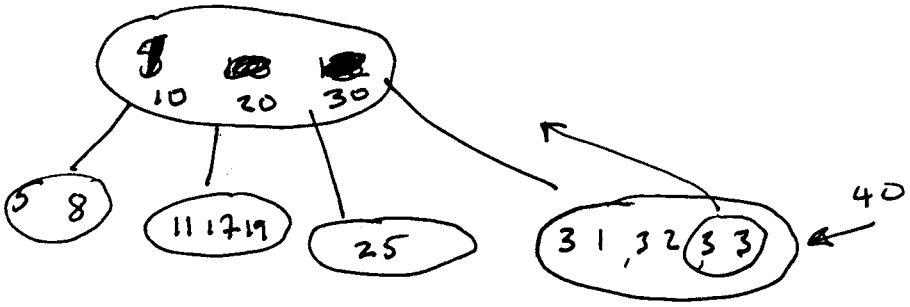
11. Logic and Hashing (5 mins)

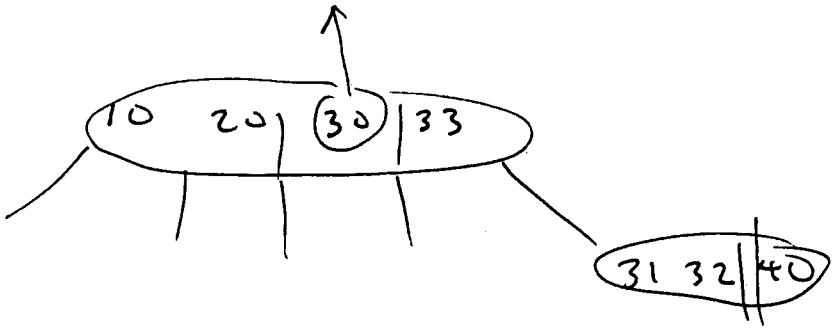
Suppose you want to generate 1-word hash key from a multi-word input key by applying a logical bit operation to all the input key's words, which you can assume look like random 32-bit integers. You can either AND all the input words together, OR them all together, or XOR them all together to generate your hash key. Is any of these methods likely to give a better hash function than the others? If so, which and why?

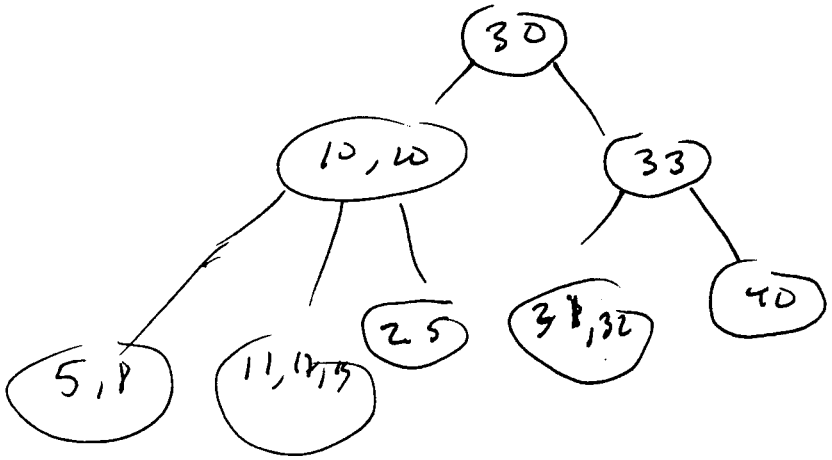
Ans: repeated ANDs will drive the key you generate toward all 0 ORs to 1. Your best bet is XOR, which will tend to preserve the number of 0's and 1's and keep the output hash key random.

6. (2,3,4) Trees (10 mins)
Insert 40 into this (2,3,4) tree.

Ans:



Ans: 9




Figure C.16 Steps in the Heapsort Algorithm