University of Nevada, Las Vegas Computer Science 477/677 Fall 2025

Study for Examination February 11, 2025

1. Fill in the blanks.

   (a) Any comparison-based sorting algorithm on a file of size n executes $\Omega(n \log n)$ comparisons.

   (b) Name two well-known divide-and-conquer sorting algorithms.

   quicksort

   mergesort

   (c) The asymptotic time complexity to find an item in an ordered array of length n is $O(\log n)$ using the divide and conquer algorithm **binary search.**

   (d) in a priority queue, each item in the structure represents an **unfulfilled obligation.**

   (e) Three kinds of priority queues we've mentioned in class are **stack**, **queue**, and *heap*.

   (f) The height of a binary tree with 10 leaves must be at least **4**. Exact answer please: no partial credit.

   (g) "pop" and "push" are operators of **stack**.

   (h) "fetch" and "store" are operators of **array**.

2. Fill in each blank. Write Θ if that is correct; otherwise write O or Ω, whichever is correct. Recall that log means $\log_2$.

   (a) $\log n^2 = \Theta(\log n^3)$

   (b) $\log(n!) = \Theta(n \log n)$

   (c) $\sum_{i=0}^{n-1} i^k = \Theta(n^k)$

   (d) $2^{\log^2 n} = O(n^n)$

   (e) $\log n = \Theta(\ln n)$

3. True or False.

   (i) **F** A good programmer would never use an unordered list as a search structure.

   (ii) **F** If $\log F(n) = \Theta(\log G(n))$, then $F(n)$ must be $\Theta(G(n))$.

4. Find the asymptotic time complexity of each of these code fragments in terms of n, using Θ notation.

   (a)
   ```
   for(float x = n; x > 1; x = x/2)
   ```

   $\Theta(\log n)$

   (b)
   ```
   for(int i = 0; i*i < n; i++)
   ```

   $\Theta(\sqrt{n})$

   (c)
   ```
   for(float x = n; x > 2.0; x = sqrt(x))
   ```

   $\Theta(\log \log n)$

   (d)
   ```
   for(int i = 1; i < n; i++)
     for(int j = 1; j < i; j=2*j)
   ```

   $\Theta(n \log n)$

   (e)
   ```
   for(int i = 1; i < n; i++)
     for(int j = i j < n; j=2*j)
   ```

   $\Theta(n)$

5. A stack of integers could be implemented in C++ as a linked list as follows.

```
struct stack
 {
  int item;
  stack*link;
 };
```

Finish writing the code for the operators push, pop, and empty, below.

```
struct stack
 {
  int item;
  stack*link;
 };

void push(stack*&s,int newitem) // pushes newitem onto s
 {
  stack*temp = new stack;
  temp->item = newitem;
  temp->link = s;
  s = temp;
 }

bool empty(stack*s) // returns true if s is empty, false otherwise
```

```
{
 return s == nullptr;
}

int pop(stack*&s) // returns and deletes most recently inserted item
{
 assert(not empty(s)); // error if you try to pop an empty stack
 int rslt = s->item;
 s = s->link; // memory leak
 return rslt;
}
```

6. Let $W_1, W_2, \ldots$ be the Wonderful numbers. Similarly to the Fibonacci numbers, each Wonderful number is twice the sum of the previous two Wonderful numbers. The first few Wonderful numbers are 1, 2, 6, 16, 44, ... Find a constant K such that $W_n = \Theta(K^n)$. Show your work.

   Assume $W_n = K^n$. Then $K^n = 2(K^{n-1} + K^{n-2})$, yielding the quadratic equation $K^2 - 2K - 2 = 0$. Thus $K = \frac{2 \pm \sqrt{4+8}}{2}$. Since K must be positive, $K = 1 + \sqrt{3}$.

7. The following portion of C++ code contains an array implementation of `queue`. Fill in the missing code for the operators "`enqueue`" and "`empty`."

```
struct queue
{
 int A[N]; // N is a constant large enough to prevent false overflow
 int rear = 0;
 int front = 0;
 // initially the queue is empty
};

void enqueue(queue&q,int newitem) // inserts newitem into q
{
 if(q.rear > N-1) cout << "False overflow." << endl;
 q.A[q.rear] = newitem;
 q.rear++;
}


bool empty(queue&q) // returns true if q is empty, false otherwise
{
 return q.rear == q.front;
}
```

```
int dequeue(queue&q) // returns an item from q and deletes that item
 {
  assert(not empty(q)); // error if you try to dequeue from an empty queue
  int rslt = q.A[q.front];
  q.front++;
  return rslt;
 }
```

8. Exactly one of the following is true for a binary tree with n nodes. Which one?

   The height of the tree is $O(\log n)$.
   The height of the tree is $\Theta(\log n)$.
   The height of the tree is $\Omega(\log n)$.

   The third one, Ω.

9. The items B, Q, T, M, A, X, L, are inserted into a binary search tree in that order.

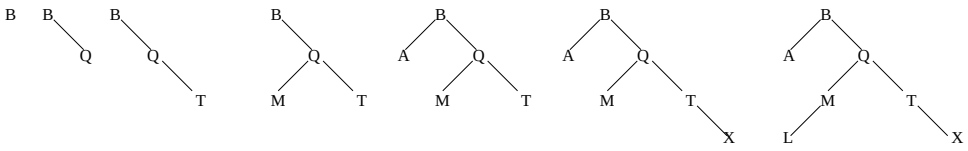
   (a) List the items of the tree in preorder.

   (b) List the items of the tree in inorder.

   (c) List the items of the tree in postorder.

   (d) List the items of the tree in level order.

   First, construct the tree in seven insertion steps.



   preorder: BAQMLTX

   inorder: ABLMQTX

   postorder: ALMXTQB

   level order: BAQMTLX

10. What follows is part of a C++ implementation of binary tree of integer. (It runs, I checked.) Fill in the missing code for the recursive functions `insert`, and `find`.

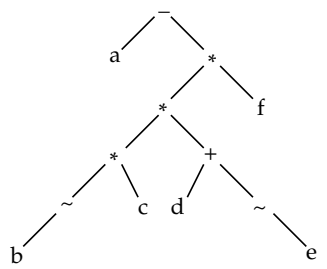
```
struct treenode;
typedef treenode*tree;
struct treenode
 {
  int item;
  tree left;
  tree right;
 };

tree mainroot; // the root of the binary search tree

void insert(tree&root,int n)
  // inserts n into the binary search tree.  Duplicates are not inserted.
 {
  if(root == NULL)
   {
    root = new treenode;
    root->item = n;
   }
  else if(n < root->item)
   insert(root->left,n)
  else if(n > root->item)
   insert(root->right,n)
  else; // duplicate items are not inserted
 }

bool find(tree root, int n)
  // returns true if some node contains n, false otherwise.
 {
  if(root)
   if(root->item == n) return true;
   else if(n < root->item) return find(root->left,n);
   else return find(root->right,n);
  else return false;
 }
```

11. A min-heap implemented as a binary tree which is, in turn, implemented as an array, is shown in the first row of the array below. In the subsequent rows, show the evolution of the heap when "B" is inserted. (The first column stores the size of the heap. How does that effect your computaton?) You may need to add additional rows.

9	A	H	C	M	N	F	G	P	R	K
10	A	H	C	M	N	F	G	P	R	B
10	A	H	C	M	B	F	G	P	R	N
10	A	B	C	M	H	F	G	P	R	N

The initial column shows the current size of the heap. The entry K in column 10 is not data, since the actual heap initially only occupies 9 columns. In row 2, B is inserted into column 10. After two bubble-up steps, min-heap order is restored in row 4.

12. Rewrite the infix expression $a--b*c*(d+-e)*f$ in prefix notation, and then in postfix notation. Don't forget that in prefix and postfix notation, we use "$-$" for subtraction, but we use "$\sim$" for negation. It helps to draw the parse tree first.
Prefix: $-a**\sim bc+d\sim ef$
Postfix: $ab\sim c*de\sim +*f*-$



13. what well-known algorithm does the procedure `mystery1` implement? What well-known algorithm does `mystery2` implement? Assume that A has been declared as int A[n].

```
void swap(int x, int y)
 {
  int z = x;
  x = y;
  y = z;
 }

void mystery1()
 {
  bool finshd = false;
  while(not finshd)
   {
    finshd = true;
    for(int i = 0; i < n-1; i++)
     if(A[i+1] < A[i])
      {
       finshd = false;
       swap(A[i],A[i+1]);
      }
   }
 }
```

Bubblesort.

```
void mystery2()
 {
  for(int i = 0; i < N; i++)
   for(int j = i+1; j < N; j++)
    if(A[j] < A[i]) swap(A[i],A[j]);
 }
```

Insertion sort.

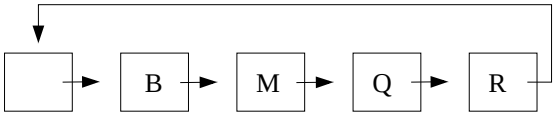
14. Solve these recurrences, giving the asymptotic answer in terms of n, using Θ notation.

    (a) $F(n) = 2F(n/2) + 3F(n/3) + n^2$;
    $\alpha_1\beta_1^\gamma + \alpha_2\beta_2^\gamma = 2(\frac{1}{2})^2 + 3\left(\frac{1}{3}\right)^2 = 5/6 < 1$. Therefore $F(n) = \Theta(n^\gamma) = \Theta(n^2)$

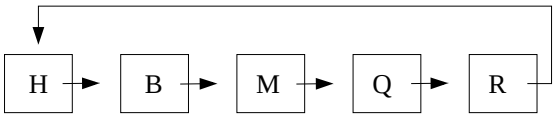
    (b) $F(n) = 8F(n/2) + n^3$;
    $\log_B C = \log_2 8 = 3 = C$. Therefore $F(n) = \Theta(n^C \log n) = \Theta(n^3 \log n)$.

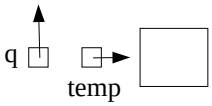
    (c) $F(n) = F(3n/5) + F(4n/5) + n$
    $\left(\frac{3}{5}\right) + \left(\frac{4}{5}\right) = 7/5 > 1$. Therefore we need to find δ such that $\left(\frac{3}{5}\right)^\delta + \left(\frac{4}{5}\right)^\delta = 1$. That choice is $\delta = 2$.
    Thus $F(n) = \Theta(n^\delta) = \Theta(n^2)$.

15. Show a circular queue with dummy node items B, M, Q, R, in that order, from front to rear. then show how the queue changes when you insert H, and then execute dequeue.

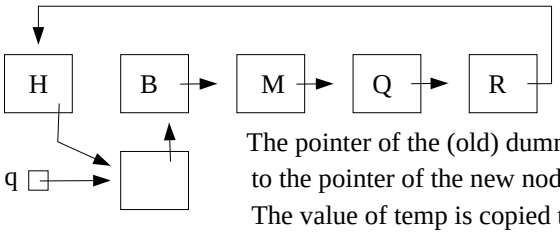


The initial queue. Static pointer q points to the dummy node.
Dummy points to front node. Rear node points to dummy.
All nodes are private; q is the only publically visible part of the queue.



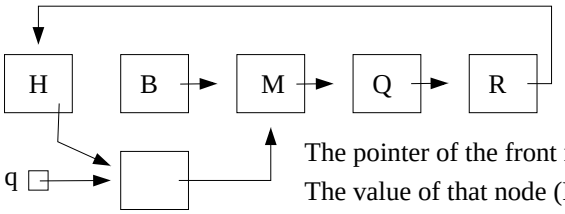


New local variable temp points to a new node.
H, the new datum is written into the dummy node.



The pointer of the (old) dummy node is copied
to the pointer of the new node.
The value of temp is copied to the pointer q
The new node becomes the dummy node, and the old dummy is the rear node.
temp is deallocated. Static q is still the only public part of the structure..

Now execute dequeue.



The pointer of the front node is copied to q.
The value of that node (B in this example)
is returned.

If memory space is a problem, deallocate the old front node.