University of Nevada, Las Vegas Computer Science 477/677 Fall 2023

Study for Examination September 29, 2023

Be prepared to answer questions similar to those here, and also similar to those on the homeworks.

1. Fill in the blanks.

   (a) Any comparison-based sorting algorithm on a file of size n must execute $\Omega(n \log n)$ comparisons in the worst case.

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

   mergesort

   quicksort

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) $\log n^2 = O\left(\log n^3\right)$

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

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

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

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

3. Fill in each blank with one of the words, *stack*, *heap*, *queue*, or *array.*

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

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

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

   (a)
   ```
   for(int i = 0; i*i < n; i++)
   ```
   $\Theta(\sqrt{n})$

   (b)
   ```
   for(int i = 0; i < n; i++)
    for(int j = 1; j < i; j = 2*j);
   ```
   The inner loop takes $\Theta(\log i)$ time for each i.

   We approximate the time complexity of the code using a definite integral:

   $$\int_1^n \ln x \, dx = \Theta(n \log n)$$

(c)

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

The inner loop takes $\Theta(\log n - \log i)$ time for each i, as I showed during the lecture.

The time complexity of the code is approximated by $\displaystyle\int_1^n (\ln n - \ln x)\,dx = \Theta(n)$

(d)

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

(`sqrt(x)` returns the square root of `x`.)

Taking the log of everything, we have

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

let $y = \log x$ and $m = \log n$. Recall that $\log \sqrt{x} = \dfrac{\log x}{2}$, and that $\log 2 = 1$. Substituting, we have:

```
for(float y = m; y > 1.0; y = y/2)
```

Thus the time complexity is $\Theta(\log m) = \Theta(\log \log n)$

(e)

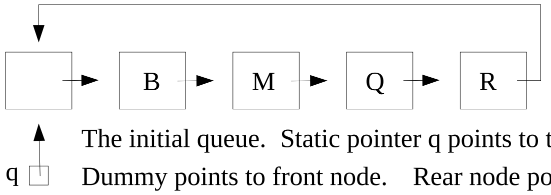
```
for(int i = 1; i < n; i = 2*i)
 for(int j = 2; j < i; j = j*j);
```

Let $k = \log i$, $m = \log n$, and $\ell = \log j$. Substituting, and approximating everything by integers, we have

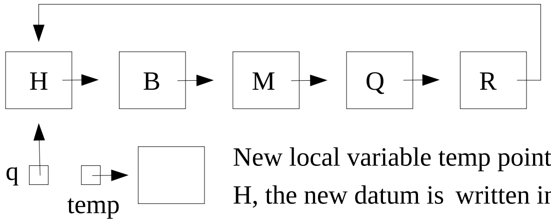
```
for(int k=0; k < m; k = k+1)
 for(int l = 1; l < k; l = 2*l)
```

From example (b), we know the time complexity is $\Theta(m \log m)$, which is $\Theta(\log n \log \log n)$

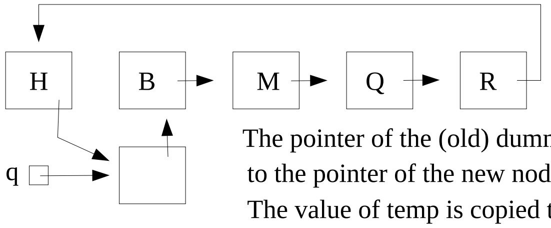
5. Show a circular queue with dummy node items B, M, Q, R, in that order, from front to rear. then show how the queue changes when you insert H, and then execute dequeue.



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.



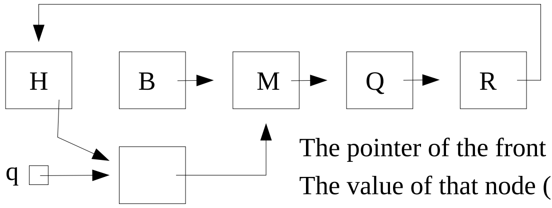
New local variable temp points to a new node.
H, the new datum is written into the dummy node.



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

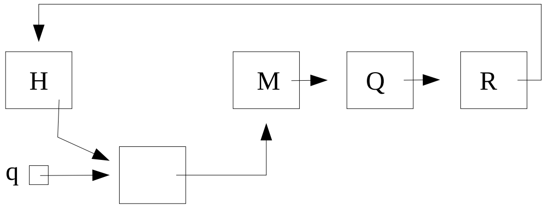
The new node becomes the dummy node, and the old dummy is the rear node.
temp is deallocated. Static q is still the only public part of the structure..

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.



6. 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.

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

int pop(stack*&s)
 {
  int rslt = s->item;
  s = s->link;
  return rslt;
 }

bool empty(stack*s)
 {
  return s == NULL;
 }
```

7. Let $F_1, F_2, \ldots$ be the Fibonacci numbers. Find a constant K such that $F_n = \Theta(K^n)$. Show the steps.

We first state, without proof, that we can assume that $F_n = C\,K^n$. This is true, but the proof is a bit tricky.

Then

$$\begin{aligned}
F_n &= F_{n-1} + F_{n-2} \\
K^n &= K^{n-1} + K^{n-2} \\
\text{Divide by } K^{n-2} : K^2 &= K + 1 \\
K^2 - K - 1 &= 0 \\
\text{By the quadratic formula: } K &= -1 \pm \frac{\sqrt{5}}{2} \\
\text{But } K \geq 0. \text{ Thus } K &= -1 + \frac{\sqrt{5}}{2}
\end{aligned}$$

8. (a) What is the purpose of the function `power` given below?

```
float power(float x, int n) // input condition: n >= 0
 {
  int m = n;
  float y = x;
  float z = 1.0;
  while(m > 0)
   {
    if(m%2) z = z*y;
    m = m/2;
    y = y*y;
   }
  return z;
 }
```

To compute x^n.

(b) Find a loop invariant of the while loop.

$y^m * z = x^n$

9. 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 overflow
  int rear = 0;
  int front = 0; // initially the queue is empty
 };

void enqueue(queue&q,int newitem) // inserts newitem into q
 {
  q.A[q.rear] = newitem;
  q.rear++;
 }

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

int dequeue(queue&q) // returns an item from q and deletes that item
 {
  int rslt = q.A[q.front];
  q.front++;
  return rslt;
 }
```

10. Walk through heapsort for the array RQWPYEFZUB.

1	2	3	4	5	6	7	8	9	10
R	Q	W	P	Y	E	F	Z	U	B
R	Q	W	Z	Y	E	F	P	U	B
R	Z	W	Q	Y	E	F	P	U	B
R	Z	W	U	Y	E	F	P	Q	B
Z	R	W	U	Y	E	F	P	Q	B
Z	Y	W	U	R	E	F	P	Q	B
B	Y	W	U	R	E	F	P	Q	**Z**
Y	B	W	U	R	E	F	P	Q	**Z**
Y	U	W	B	R	E	F	P	Q	**Z**
Y	U	W	P	R	E	F	B	Q	**Z**
Q	U	W	P	R	E	F	B	**Y**	**Z**
W	U	Q	P	R	E	F	B	**Y**	**Z**
B	U	Q	P	R	E	F	**W**	**Y**	**Z**
U	B	Q	P	R	E	F	**W**	**Y**	**Z**
U	R	Q	P	B	E	F	**W**	**Y**	**Z**
F	R	Q	P	B	E	**U**	**W**	**Y**	**Z**
R	F	Q	P	B	E	**U**	**W**	**Y**	**Z**
R	P	Q	F	B	E	**U**	**W**	**Y**	**Z**
E	P	Q	F	B	**R**	**U**	**W**	**Y**	**Z**
Q	P	E	F	B	**R**	**U**	**W**	**Y**	**Z**
B	P	E	F	**Q**	**R**	**U**	**W**	**Y**	**Z**
P	B	E	F	**Q**	**R**	**U**	**W**	**Y**	**Z**
P	F	E	B	**Q**	**R**	**U**	**W**	**Y**	**Z**
B	F	E	**P**	**Q**	**R**	**U**	**W**	**Y**	**Z**
B	F	E	**P**	**Q**	**R**	**U**	**W**	**Y**	**Z**
F	B	E	**P**	**Q**	**R**	**U**	**W**	**Y**	**Z**
E	B	**F**	**P**	**Q**	**R**	**U**	**W**	**Y**	**Z**
B	**E**	**F**	**P**	**Q**	**R**	**U**	**W**	**Y**	**Z**
B	**E**	**F**	**P**	**Q**	**R**	**U**	**W**	**Y**	**Z**