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

Answers to Examination February 8, 2023

1. Fill in the blanks.

   (a) [10 points] Any comparison-based sorting algorithm on a file of size n must execute $\Omega(n \log n)$ comparisons in the worst case.

   (b) [10 points] 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) [5 points]

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

   (b) [5 points]

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

   (c) [5 points]

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

   (d) [5 points]

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

   (e) [5 points] $\log n = \Theta(\ln n)$

3. [10 points] 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. [10 points each]

   (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);
   ```

   $\Theta(n \log n)$

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

$\Theta(n)$

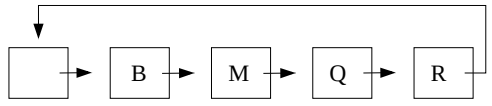
(d) `for(float x = n; x > 2.0; x = sqrt(x))` (`sqrt(x)` returns the square root of `x`.)

$\Theta(\log \log n)$

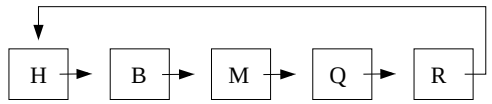
(e)
```
for(int i = 1; i < n; i = 2*i)
    for(int j = 2; j < i; j = j*j);
```
(Hint: use substitution)

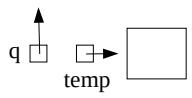
$\Theta(\log n \log \log n)$

5. [20 points] 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.

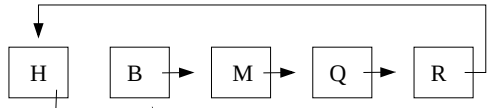


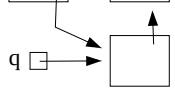
The initial queue. Static pointer q points to the dummy node.
Dummy points to front node. Rear node points to dummy.
All nodes are private; q is the only publically visible part of the queue.





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..

6. [30 points] 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;
  temp->link = s;
  s = temp;
 }

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

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

7. [20 points] 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 assume that $F_n = CK^n$ for all n, for some constant C. This is false, but we can assume it's true. We then have

$$\begin{aligned} F_{n+2} &= F_{n+1} + F_n \\ CK^{n+2} &= CK^{n+1} + CK^n \\ CK^{n+2} - CK^{n+1} - +CK^n &= 0 \\ CK^n(K^2 - K - 1) &= = 0 \end{aligned}$$

There are three solutions to the last equation: $K = 0$ and $K = \dfrac{1 \pm \sqrt{5}}{2}$

Since K must be positive, we have $K = \dfrac{1 + \sqrt{5}}{2}$.

8. (a) [10 points] What is the purpose of the function `power` given below? To compute x^n.
   (b) [20 points] Find a loop invariant of the while loop. The loop invariant is the equation: $x^n = z\,y^m$

```
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;
 }
```

9. [20 points] 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.rear == q.front;
 }

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

10. [20 points] The following code could be used in a C++ program implementing quicksort. What is the loop invariant of the loop?

    1. For any first $<$ i $\leq$ lo, q.A[i] $\leq$ pivot
    2. For any hi $<$ i $\leq$ last, q.A[i] $\geq$ pivot

```
void quicksort(int first,int last) // sorts the subarray A[first .. last]
 {
  if(first < last) // if first >= last, we are done
   {
    int mid = (first+last)/2;
    int pivot = A[mid];
    swap(A[mid],A[first]); // move pivot to first position
    int lo = first;
    int hi = last;
    while(lo < hi)
     {
      if(A[lo+1] > pivot and A[hi] < pivot)
       {
        swap(A[lo+1],A[hi]);
        lo++;
        hi--;
       }
      if(A[lo+1] <= pivot) lo++;
      if(A[hi] >= pivot) hi--;
     }
    //assert(lo == hi);
    swap(A[first],A[lo]); // move pivot between subarrays
    if(lo < mid) // sort the smaller subarray first
     {
      quicksort(first,lo-1);
      quicksort(lo+1,last);
     }
    else
     {
      quicksort(lo+1,last);
      quicksort(first,lo-1);
     }
   }
 }
```