

University of Nevada, Las Vegas Computer Science 477/677 Fall 2024

Answers to Assignment 2: Due September 12, 2026

Follow our TA, Shareem Arjyal's (arjyal@unlv.nevada.edu) instructions on how to turn in the assignment.

Name: _____

You are permitted to work in groups, get help from others, read books, and use the internet.

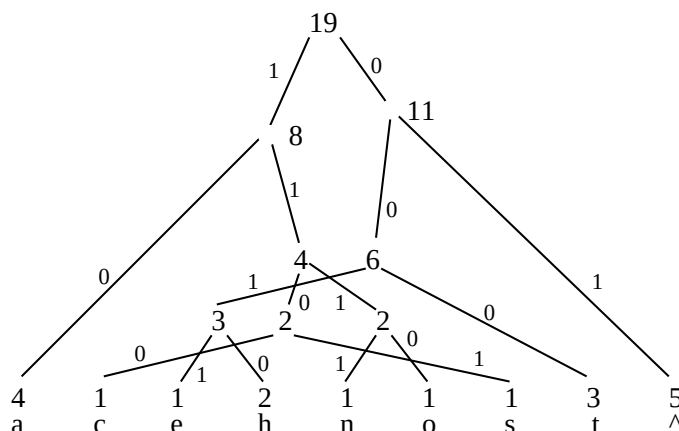
1. Fill in the blanks.

- (i) No good programmer would ever use an unsorted list as a search structure. **T** (True or False.)
- (ii) Heapsort is a fast version of **selection sort**
- (iii) Treesort is a fast version of **insertion sort**
- (iv) Shell sort is inspired by **bubblesort** but is faster.
- (v) The items of a priority queue represent **unfulfilled obligations**

2. Prefix Codes and Huffman's Algorithm.

- (i) Here is a prefix-free code for the alphabet $\{a, c, e, h, i, n, s, t, \wedge\}$, where $\wedge$ denotes a blank. The first column contains is the source alphabet, the second column the weight of each symbol, and the third column its codeword.

a	4	10
c	1	1100
e	1	0011
h	2	0010
n	1	1111
i	1	1110
s	1	1101
t	3	000
$\wedge$	5	01



Draw a Huffman tree for that code, and then Find the source message encoded by the following binary string, where $\wedge$ means space.

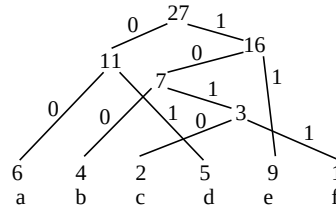
0000010001101110010000011101100000111101111011001001010000

the cat sat on a hat

(ii)

Here is another weighted alphabet. Use Huffman's algorithm to find an optimal prefix code for that alphabet. Enter the codewords in the third column of the table.

a	6	00
b	4	101
c	2	1010
d	5	01
e	9	11
f	1	1011



3. Loop Invariants. In a program, each loop has a loop invariant. The invariant of a loop is a statement (frequently, but not necessarily, an equation) such that

- (a) The invariant holds just before the first iteration of the loop.
- (b) For any iteration of the loop, if the invariant holds at the beginning of that iteration it holds at the end of that iteration.

It follows logically that the invariant holds just after the last iteration of the loop. Using that fact, the invariant can (hopefully) be used to prove correctness of the loop.

- (i) The purpose of the following code is return an integer p which is twice the value of its integral parameter $n \geq 0$. The code has just one loop and its loop invariant I is the equation $2m + p = 2n$.

```
int double(int n)
{
(0)  assert(n >= 0);
(1)  int m = n;
(2)  int p = 0;
(3)  // initially, I holds here
(4)  while(m > 0)
(5)  {
(6)    // if, during any iteration, I holds here,
(7)    m = m-1;
(8)    p = p+2;
(9)    // then I holds here.
(10) }
(11) // finally, I holds here. Correctness of the code follows from the
(12) // the fact that I holds and that m = 0.
(12) return p;
}
```

At the end of the code, $m = 0$, since the loop has ended. The loop invariant is that $p + m = 2n$. Since $m = 0$, we have $p = 2n$, that is, the code is correct.

- (ii) The purpose of the following subprogram is to return the product of its two integral parameters, n and k , where $k \geq 0$. For simplicity, we assume $k \geq 0$. Find the loop invariant of the loop.

```
int product(int n; int k)
{
    assert(k >= 0);
    int m = n;
    int l = k;
    int p = 0;
    while(l > 0)
    {
        if(l%2) p = p+m;
        m = 2*m;
        l = l/2;
    }
    return p;
}
```

The goal is to cause write $n * k$. The loop invariant is $n * k = m * l + p$.

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

- (i)

```
for(int i = 0; i < n; i++)
    for(int j = 0; j < n; j++)
        cout << "Hello world." << endl;;
```


 $\Theta(n^2)$
- (ii)

```
for(int i = n; i > 1; i = i/2)
    cout << "Hello world." << endl;;
```


 $\Theta(\log n)$
- (iii)

```
for(int i = 1; i < n; i++)
    for(int j = 1; j < n; j = 2*j)
        cout << "Hello world." << endl;;
```


 $\Theta(n \log n)$
- (iv)

```
for(int i = 1; i < n; i = 2*i)
    for(int j = 0; j < i; j++)
        cout << "Hello world." << endl;;
```


 $\Theta(n)$
- (v)

```
for(int i = 1; i < n; i = 2*i)
    for(int j = i; j < n; j++)
        cout << "Hello world." << endl;;
```


 $\Theta(n \log n)$
- (vi)

```
for(int i = 2; i < n; i = i*i)
    cout << "Hello world." << endl;;
```


 $\Theta(\log \log n)$

(vii) `for(int i = 1; i < n; i = i+1)`
`for(int j = 1; j < sqrt{i}; j++)`
 $\Theta(n \log \log n)$

5. Solve the following recurrences, expressing each answer in terms of n using Θ notation.

(i) $F(n) = 2F(n/2) + n$

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

(ii) $F(n) = 2F(n/2) + 1$

$$\Theta(n)$$

(iii) $F(n) = 2F(n/2) + n^2$

$$\Theta(n^2)$$

(iv) $F(n) = 2F(n/4) + 1$

$$\Theta(n^{\frac{1}{2}}) = \Theta(\sqrt{n})$$

(v) $F(n) = 2F(n/4) + \sqrt{n}$

$$\Theta(\sqrt{n} \log n)$$

(vi) $F(n) = F(n - \sqrt{n}) + 1$

$$\Theta(n^{\frac{1}{2}}) = \Theta(\sqrt{n})$$

(vii) $F(n) = 2F(n - 1) + 1$. $\Theta(2^n)$

6. Each of the following three C++ functions sorts an array `int x[n]`, using some sorting algorithm which has been mentioned in class. Assume that the function *swap* exchanges the values of its two parameters.

(i) Which sorting algorithm is encoded below? Ans: Selection sort.

```
void sort(int x[n])
{
    for(int i = 0; i+1 < n; i++)
        for(int j = i+1; j < n; j++)
            if(x[j] < x[i]) swap(x[i],x[j]);
}
```

(ii) Which sorting algorithm is encoded below? Ans: Insertion sort.

```
void sort(int x[n])
{
    for(int i = 1; i < n; i++)
        for(int j = i; j > 0 and x[j] < x[j-1]; j--)
            swap(x[j],x[j-1]);
}
```

(iii) Which sorting algorithm is encoded below? Ans: Bubble sort.

```
void sort(int x[n])
{
    for(int j = n-1; j > 0; j--)
        for(int i = 0; i < j; i++)
            if(x[i+1] < x[i]) swap(x[i],x[i+1]);
}
```

7. What is the name, and what is the purpose, of the following C++ function? Again, assume x is an array of size n .

```
bool found(int sought)
// input condition: x is in increasing order
{
    int lo = 0;
    int hi = n;
    while(lo+1 < hi)
    {
        int mid = (lo+hi)/2;
        if(sought < x[mid]) hi = mid;
        else lo = mid;
    }
    return x[lo] == sought;
}
```

Returns **true** if one some entry of x is equal to *sought*, otherwise **false**.

8. Name two different divide-and-conquer sorting algorithms.

mergesort and *quicksort*.

9. Name three different kinds of priority queue.

stack, *queue* and *heap*. (“maxheap” and “minheap” do not count as two distinct answers, since they are structurally identical.)

10. Finally, a much more difficult problem. You are given two sorted lists, say A and B . Your task is to find the median of the union of those two lists. For example, if $A = (1, 2, 5)$ and $B = (3, 4, 9, 10)$, the union is the list $(1, 2, 5, 3, 4, 9, 10)$. By sorting the union, you can observe that the median number is 4.

Describe the algorithm you would use to find the median for an arbitrary case, Let n be the size of the combined list. You want the least possible asymptotic time complexity.

You could take middle item of the sorted concatenation of the two given arrays. This would take $\Theta(n \log n)$ time. But you can do it much faster!