

Name - Ayoni Sardarban
Registration No - 19 BSC 11448

Assignment A

Arrays

Statement: 2.01.01.C.3.1

1. Problem:

Array Manipulation - Starting with a 1-dimensional array of zeroes and a list of operations, for each operation add a value to each element between two given indices, inclusive. Once all operations have been performed, return the maximum value in your array.

For example, if the length of the array of zeroes $n = 10$, The list of queries is as follows:-

a b k
1 5 3
4 8 7
6 9 1

Add the values of k between the indices a and b inclusive.

index $\rightarrow 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9 \ 10$

$[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]$

$[3, 3, 3, 3, 3, 0, 0, 0, 0, 0]$

$[3, 3, 3, 10, 10, 7, 7, 7, 0, 0]$

$[3, 3, 3, 10, 10, 8, 8, 8, 1, 0]$

The largest value is 10. So after all

operations are performed.

2. Efficient Data Structure for the problem.

Arrays is used as the efficient data structure for this problem & statement.

3. Design technique Used for the Algorithm

Normal array is used for solving the problem. A for loop is used to develop an algorithm to find the output of the problem statement.

Explain with example & for each step
dimensions of tree is

4. Pseudocode

```

main()
| int N, K, P, Q, sum, i, j, man → 0, n → 0.
| input N
| input K
| int a[N+1]
for (i → 0 to K)
| input P
| input Q
| a[P] ← a[P] + sum
for (j = Q+1 to N)
| if ((Q+1) ≤ N)
| | a[Q+1] = a[Q+1] - sum
| endif
end for
for (i → 1 to N)
| n → n + a[i]
| if (man < n)
| | man = n
| endif
end for
output man
end main
    
```

5. Time Complexity

Time complexity of the above pseudocode
is ~~$O(n^2)$~~ $O(n)$.

Stacks

1. Problem Statement:

Skyline Real Estate Developers is planning to demolish a number of old, unoccupied buildings and construct a shopping mall in their place. The task is to find the largest solid area in which the mall can be constructed.

There are a number of buildings in a certain two dimensional landscape. Each building has a height, given by $h[i]$ where $i \in [1, n]$. If you join k adjacent buildings, they will form a solid rectangle.

of area $k \times \min(h[i], h[i+1], \dots, h[i+k-1])$
For example heights array and its subarray
 $h = [3, 2, 3]$. $i = 0$. $k = 3$.
A rectangle of height $k = 2$ and length $k = 3$ can be constructed within the boundaries. The area formed is $h \cdot k = 2 \cdot 3 = 6$.

2. Efficient Data Structure Used for the Problem:

Stacks data structure is used for the given problem.

3. Design technique used for the algorithm

Divide and Conquer Approach is used for this problem. Considering the full histogram [start, end]. Find the position of the minimum value in [start, end]. Let it be n .

If the optimal subpart contains the n^{th} bar. Then its area would be:

$$\text{Height}[n] \times (\text{end} - \text{start} + 1)$$

of the optimal subpart doesn't contain the n^{th} bar then, either it will be in $[start, n-1]$ or in $[n+1, end]$, to which is divide and conquer.

$\text{Area}(\text{Start}, \text{end}) = \min(\text{Height}[u] \times (\text{end} - \text{start} + 1), \text{Area}(\text{start}, u-1) + \text{Area}(u+1, \text{end}))$; u is the position of the smallest bar in $[\text{start}, \text{end}]$.

4. Pseudocode:

```
int s[]
```

```
Histogram ( int height[] )
```

```
height.push(0)
```

```
Point sum = 0, i = 0
```

```
while ( i < height.size() )
```

```
    if ( s.empty() || height[i] > height[ s.back() ] )
```

```
        s.push(i)
```

```
        i = i + 1
```

```
    end if
```

```
else
```

```
    int t = s.back()
```

```
    s.pop()
```

```

sum = max($sum, height[i] * (s.empty() ? 1 : i - s.back() - 1));
end if
end else if
end while
return sum

```

d has a sequential code now (ii)

```

end Histogram
main()
{
    int d, base = 0, n
    int d1, d2, d3, d4, d5, d6, d7, d8, d9, d10
    int d11, d12, d13, d14, d15, d16, d17, d18, d19, d20
    int d21, d22, d23, d24, d25, d26, d27, d28, d29, d30
    int d31, d32, d33, d34, d35, d36, d37, d38, d39, d40
    int d41, d42, d43, d44, d45, d46, d47, d48, d49, d50
    int d51, d52, d53, d54, d55, d56, d57, d58, d59, d60
    int d61, d62, d63, d64, d65, d66, d67, d68, d69, d70
    int d71, d72, d73, d74, d75, d76, d77, d78, d79, d80
    int d81, d82, d83, d84, d85, d86, d87, d88, d89, d90
    int d91, d92, d93, d94, d95, d96, d97, d98, d99, d100
    int d101, d102, d103, d104, d105, d106, d107, d108, d109, d110
    int d111, d112, d113, d114, d115, d116, d117, d118, d119, d120
    int d121, d122, d123, d124, d125, d126, d127, d128, d129, d130
    int d131, d132, d133, d134, d135, d136, d137, d138, d139, d140
    int d141, d142, d143, d144, d145, d146, d147, d148, d149, d150
    int d151, d152, d153, d154, d155, d156, d157, d158, d159, d160
    int d161, d162, d163, d164, d165, d166, d167, d168, d169, d170
    int d171, d172, d173, d174, d175, d176, d177, d178, d179, d180
    int d181, d182, d183, d184, d185, d186, d187, d188, d189, d190
    int d191, d192, d193, d194, d195, d196, d197, d198, d199, d200
    int d201, d202, d203, d204, d205, d206, d207, d208, d209, d210
    int d211, d212, d213, d214, d215, d216, d217, d218, d219, d220
    int d221, d222, d223, d224, d225, d226, d227, d228, d229, d230
    int d231, d232, d233, d234, d235, d236, d237, d238, d239, d240
    int d241, d242, d243, d244, d245, d246, d247, d248, d249, d250
    int d251, d252, d253, d254, d255, d256, d257, d258, d259, d260
    int d261, d262, d263, d264, d265, d266, d267, d268, d269, d270
    int d271, d272, d273, d274, d275, d276, d277, d278, d279, d280
    int d281, d282, d283, d284, d285, d286, d287, d288, d289, d290
    int d291, d292, d293, d294, d295, d296, d297, d298, d299, d300
    int d301, d302, d303, d304, d305, d306, d307, d308, d309, d310
    int d311, d312, d313, d314, d315, d316, d317, d318, d319, d320
    int d321, d322, d323, d324, d325, d326, d327, d328, d329, d330
    int d331, d332, d333, d334, d335, d336, d337, d338, d339, d340
    int d341, d342, d343, d344, d345, d346, d347, d348, d349, d350
    int d351, d352, d353, d354, d355, d356, d357, d358, d359, d360
    int d361, d362, d363, d364, d365, d366, d367, d368, d369, d370
    int d371, d372, d373, d374, d375, d376, d377, d378, d379, d380
    int d381, d382, d383, d384, d385, d386, d387, d388, d389, d390
    int d391, d392, d393, d394, d395, d396, d397, d398, d399, d400
    int d401, d402, d403, d404, d405, d406, d407, d408, d409, d410
    int d411, d412, d413, d414, d415, d416, d417, d418, d419, d420
    int d421, d422, d423, d424, d425, d426, d427, d428, d429, d430
    int d431, d432, d433, d434, d435, d436, d437, d438, d439, d440
    int d441, d442, d443, d444, d445, d446, d447, d448, d449, d450
    int d451, d452, d453, d454, d455, d456, d457, d458, d459, d460
    int d461, d462, d463, d464, d465, d466, d467, d468, d469, d470
    int d471, d472, d473, d474, d475, d476, d477, d478, d479, d480
    int d481, d482, d483, d484, d485, d486, d487, d488, d489, d490
    int d491, d492, d493, d494, d495, d496, d497, d498, d499, d500
    int d501, d502, d503, d504, d505, d506, d507, d508, d509, d510
    int d511, d512, d513, d514, d515, d516, d517, d518, d519, d520
    int d521, d522, d523, d524, d525, d526, d527, d528, d529, d530
    int d531, d532, d533, d534, d535, d536, d537, d538, d539, d540
    int d541, d542, d543, d544, d545, d546, d547, d548, d549, d550
    int d551, d552, d553, d554, d555, d556, d557, d558, d559, d560
    int d561, d562, d563, d564, d565, d566, d567, d568, d569, d570
    int d571, d572, d573, d574, d575, d576, d577, d578, d579, d580
    int d581, d582, d583, d584, d585, d586, d587, d588, d589, d590
    int d591, d592, d593, d594, d595, d596, d597, d598, d599, d600
    int d601, d602, d603, d604, d605, d606, d607, d608, d609, d610
    int d611, d612, d613, d614, d615, d616, d617, d618, d619, d620
    int d621, d622, d623, d624, d625, d626, d627, d628, d629, d630
    int d631, d632, d633, d634, d635, d636, d637, d638, d639, d640
    int d641, d642, d643, d644, d645, d646, d647, d648, d649, d650
    int d651, d652, d653, d654, d655, d656, d657, d658, d659, d660
    int d661, d662, d663, d664, d665, d666, d667, d668, d669, d670
    int d671, d672, d673, d674, d675, d676, d677, d678, d679, d680
    int d681, d682, d683, d684, d685, d686, d687, d688, d689, d690
    int d691, d692, d693, d694, d695, d696, d697, d698, d699, d700
    int d701, d702, d703, d704, d705, d706, d707, d708, d709, d710
    int d711, d712, d713, d714, d715, d716, d717, d718, d719, d720
    int d721, d722, d723, d724, d725, d726, d727, d728, d729, d730
    int d731, d732, d733, d734, d735, d736, d737, d738, d739, d740
    int d741, d742, d743, d744, d745, d746, d747, d748, d749, d750
    int d751, d752, d753, d754, d755, d756, d757, d758, d759, d760
    int d761, d762, d763, d764, d765, d766, d767, d768, d769, d770
    int d771, d772, d773, d774, d775, d776, d777, d778, d779, d780
    int d781, d782, d783, d784, d785, d786, d787, d788, d789, d790
    int d791, d792, d793, d794, d795, d796, d797, d798, d799, d800
    int d801, d802, d803, d804, d805, d806, d807, d808, d809, d810
    int d811, d812, d813, d814, d815, d816, d817, d818, d819, d820
    int d821, d822, d823, d824, d825, d826, d827, d828, d829, d830
    int d831, d832, d833, d834, d835, d836, d837, d838, d839, d840
    int d841, d842, d843, d844, d845, d846, d847, d848, d849, d850
    int d851, d852, d853, d854, d855, d856, d857, d858, d859, d860
    int d861, d862, d863, d864, d865, d866, d867, d868, d869, d870
    int d871, d872, d873, d874, d875, d876, d877, d878, d879, d880
    int d881, d882, d883, d884, d885, d886, d887, d888, d889, d890
    int d891, d892, d893, d894, d895, d896, d897, d898, d899, d900
    int d901, d902, d903, d904, d905, d906, d907, d908, d909, d910
    int d911, d912, d913, d914, d915, d916, d917, d918, d919, d920
    int d921, d922, d923, d924, d925, d926, d927, d928, d929, d930
    int d931, d932, d933, d934, d935, d936, d937, d938, d939, d940
    int d941, d942, d943, d944, d945, d946, d947, d948, d949, d950
    int d951, d952, d953, d954, d955, d956, d957, d958, d959, d960
    int d961, d962, d963, d964, d965, d966, d967, d968, d969, d970
    int d971, d972, d973, d974, d975, d976, d977, d978, d979, d980
    int d981, d982, d983, d984, d985, d986, d987, d988, d989, d990
    int d991, d992, d993, d994, d995, d996, d997, d998, d999, d1000
}

```

5. Time Complexity

$O(n \log n)$ (using merge sort)

Queues

1. Problem Statement:-

You are given Q queries. Each query consists of a single number N. You can perform any of the 2 operations on N in each move:

- (i) If we take 2 integers, a and b, where $N = a \times b$ ($a \neq 1, b \neq 1$), then we can change $N = \min(a, b)$.
- (ii) Decrease the value of N by 1.

Determine the minimum number of moves required to reduce the value of N to 0.

2. Efficient Data Structure for this problem :-

Queue data structure is used for this problem statement.

3. Design technique used for the algorithm :-

We use queues to solve this problem by implementing the

unique operation of queue is FIFO.

4. Pseudocode

```
main()
{
    int dis[10000]
    int n, que = 0, qf, tf
    input(que)
    while (que --)
    {
        input(n)
        if (n == 0)
            output 0
        continue
    }
    queue q
    q.push(n)
    dis[n] = 1
    while (!q.empty())
    {
        int now = q.front()
        q.pop()
        if (dis[now - 1] == 0)
        {
            dis[now - 1] = dis[now] + 1
            if (now - 1 == 0)
                break
            q.push(now - 1)
        }
    }
}
```

```

else
for (i = 2 ; i <= row, i++)
{
    if (now % i == 0)
        int fac = min(i, now / i)
        if (dis[fac] == 0)
            dis[fac] = dis[now] + 1
            q.push(fac);
        endif (0 == 0);
    end if
}
end for
end while
end while
output (dis[0] - 1)

```

5. Time Complexity

$$O(n^2) = [1 + \text{row}] \times [1 + \text{row}]$$

$$1 + [\text{row}]^2 + [1 + \text{row}]^2$$

$$O = (1 + \text{row})^2$$

$$(1 + \text{row})^2 = O$$

Linked List

1. Problem Statement:

A linked list is said to contain a cycle if any node is visited more than once while traversing the list.

The function has one parameter, a pointer to a node object named head that points to the head of a linked list. The function returns a boolean denoting whether or not there is a cycle in the list, if there is a cycle return true otherwise false.

2. Data Structure used for this Problem:

Linked List

3. Design technique for the algorithm:

There are 3 scenarios to consider:

- (i) the list is empty (ie head is null)
- (ii) the list does not contain a cycle, so one can traverse the list and terminate. Once there are no more nodes.

(ii) The list contains a cycle so attempting to traverse will keep in a loop.

We must traverse the list using two pointers that we will refer to as slow and fast. Our slow pointer moves forward 1 node at a time, and our fast pointer moves forward 2 nodes at a time. If at any point in time these pointers refer to the same object, then there is a loop, otherwise the loop does not contain a loop.

4. Pseudocode

```
bool hascycle ( Node *head )
```

```
{ Node *fast = head
```

```
; Node *slow = head
```

```
while( fast != NULL and slow != NULL )
```

```
    if( slow == fast ) return true
```

```
    fast = fast -> next
```

```
    slow = slow -> next
```

```
    if( fast == slow ) return true
```

```

    : return 1
    : end if no cycle found in left/binary tree
    : end while
    : return 0
    : hascycle
end

```

S. Time Complexity

$O(n)$

Tree

I. Problem Statement

We define a tree as a binary tree if it has the following ordering requirements:-

- (i) data value of every node in a node's left subtree is less than the data value of that node.
- (ii) The data value of every node in a node's right subtree is greater than the data value of that node.

Given the root node of a binary tree can we determine its binary search tree.

2. Data Structure Used

Linked List is used for representing the trees.

3. Design Technique for the Algorithm

A recursive function is used to implement the algorithm.

4. Pseudocode

```
bool check(Node *root, int minval,  
           int maxval)  
{  
    if (root == NULL)  
        return true;  
    if (root->data < minval ||  
        root->data > maxval)  
        return false;  
    return (check(root->left, minval,  
                 root->data - 1) and  
            check(root->right, root->data + 1));  
}
```

```

    root->data + 1, maxVal)
}
end check. Is that a good id
bool check2(Node *root) {
    return check(root, 0, 10000)
}
;
return . check (root, 0, 10000)
;
end check2. In input set off size
;

```

5. Time Complexity

With partitioning and Beers we had an
 $O(N)$ here to reduce even sort

minimum graph

and H partitioning took up to N^2

1. Problem Statement

We are given a tree (a simple connected graph with no cycles)

Find the maximum number of edges

you can remove edges until the tree

tree to get a forest such that each connected component of the

forest contains an even number of nodes.

For example a tree with 4 nodes can be cut at most 1 time to create an even forest.

2. Data Structure Used

We use a linked list to represent the graph and the tree in the solution.

3. Design Technique for the Algorithm

What we need is a subtree that has even number of vertices and as we get it we can remove the edge that connects it to the remaining of the tree so that we are left with a subtree of even vertices and the remaining tree with vertices $N - n$. We are left with the same problem again with remaining tree having even number of vertices because N is even so even - even = even. So we repeat this algorithm until the remaining tree cannot be decomposed. To maximise the number of subtree we have to remove the subtree which cannot be decomposed in the above manner.

4. Pseudocode

```
dfs (int node)
    visit[node] = true
    int num_vertex = 0
    for (i → 0 to al[node].size())
        if (!visit[al[node][i]])
            num = dfs(al[node][i])
            if (num % 2 == 0)
                ans ++
            else
                num_vertex = num_vertex + num
    if (ans >= num / 2)
        return num_vertex + 1
    end for
    return num_vertex + 1
end dfs
```

5. Time Complexity

$O(N)$

Divide and Conquer

1. Problem Statement

Tower of Hanoi is a mathematical puzzle where we have three rods and n disks. The objective of the puzzle is to move the entire stack to another rod obeying the simple rules:-

- (i) Only one disk can be moved at a time.
- (ii) Each move consists of taking the upper disk from one of the stocks and placing it on top of another stock i.e disk can be moved if it is the uppermost disk on the stack.
- (iii) No disk may be placed on top of a smaller disk.

2. Data Structure Used

Normal number of characters are used.

3. Design Technique

Divide and Conquer algorithm along with a recursive function -

4. Pseudocode

towerofhanoi (int n, char fromrod, char torod,
char auxrod)

```
if n = 1  
    output ('move disk' 1 from rod' fromrod,  
            'to rod', torod)  
end if
```

towerofhanoi (n-1, fromrod, auxrod, torod)

```
output ('move disk' n, from rod',  
        'fromrod', torod, torod)
```

towerofhanoi (n-1, auxrod, torod, fromrod)

```
end towerofhanoi
```

main

```
n = 4
```

```
towerofhanoi (n, A, C, B)
```

```
return 0
```

end main

5. Time Complexity

~~O(n!)~~ O(n log n)

Brute Force

1. Problem Statement

Given an array of n elements.
find the given element in the array.

2. Data Structure Used

Array is used for the problem

3. Design Technique

Linear Search, which is a Brute Force method.

4. Pseudocode

Search (int arr[], int n, int u)

```
; int i  
; for(i = 0 to n)  
;   if(arr[i] = u)  
;     return i  
;   endif
```

```
end for  
return -1
```

end search

main

```
int arr[] = input from user  
int n = 10  
n = size of arr  
result = search(arr, n, n)
```

end main

5. Time Complexity

$O(N)$

Dynamic Programming

1. Problem Statement

n^{th} Catalan Number \rightarrow Catalan numbers are a sequence of natural numbers that occurs in many counting problems like:

- i) The first few catalan numbers for $n = 0, 1, 2, 3, \dots$, are 1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, ...

2. Data Structure Used

Array to store the output for each n^{th} number.

3. Design Technique

An ~~algorithm~~ A recursive function along with storing them in an array of numbers is used.

4. Pseudocode

```
catalan ( int n )  
{  
    int catalan [n + 1];  
    catalan [0] = catalan [1] = 1;
```

```

for (i → 2 to n)
    catalan[i] = 0
    for (j → 0 to i)
        catalan[i] = catalan[i] +
                      catalan[j] * catalan[i-j-1]
    end for
    return catalan[n]
end catalan
main
    for (i ← 0 to 10)
        output(catalan[i])
    end main

```

5. Time Complexity

$$O(N^2)$$