**Cấu trúc dữ liệu và giải thuật**

**THI HỌC KỲ - LỚP E**

//exe1E20:

**E2K01 - DQUEUE - 1**

**Độ khó:**1

**Mô tả đề bài:**

Given an array A[] consisting of n elements and a string str[] consisting of the character strings "F", "B". Your task is to create a dqueue that satisfies the requirement that even numbers are put in the front, odd numbers are put in the back. Next, you perform the operations sequentially in the string str[]: if str[i] is "F", you perform the pop\_front() operation, if the str[i] is "B", you perform the pop\_back() operation. Let's look up Dqueue content from front to back.

**Input:**

* The first line includes the number of test cases T.
* The following lines enter the test suites. Each test set consists of 3 lines: the first line takes n numbers, the next line takes n numbers A[i], the last line puts the string str[]. There is no space in the input string.
* Constraints: 1≤T≤100; 1≤length(str)≤103.

**Output:**

* Output the results of each test line by line.

**Examples:**

|  |  |
| --- | --- |
| **Input:** | **Output:** |
| 1  10  5 10 15 20 25 30 35 40 45 50  FFBBBFF | 10 5 15 |

 //exe2E20:

**E2K02 - DQUEUE - 2**

**Độ khó:**1

**Mô tả đề bài:**

Given array A[] and an array of strings str[] have n elements and n words. The words in the array of string consist of only the words "F" and "B". Your task is to create a Dqueue that satisfies the requirements if the word str[i] is “F” you do push\_Front(A[i]), if the word str[i] is “B” you do push\_back(A[i]) operation. Finally, you give the Dqueue content lookup from the back to the front().

**Input:**

* The first line includes the number of test cases T.
* The following lines enter the test suites. Each test set consists of 3 lines: the first line takes n numbers, the next line takes n numbers A[i], the last line puts n words str[i]. Each character is followed by a space.
* Constraints: 1≤T≤100; 1≤length(str)≤103.

**Output:**

* Output the results of each test line by line.

**Examples:**

|  |  |
| --- | --- |
| **Input:** | **Output:** |
| 1  10  5 10 15 20 25 30 35 40 45 50  F F F B F B B F F B | 50 35 30 20 5 10 15 25 40 45 |

//exe3E20:

**E2K03 - STACK - 1**

**Độ khó:**1

**Mô tả đề bài:**

Given an expression exp[]. Please replace the correct opening and closing parentheses with the value 0, 1. Replace the incorrect opening and closing parentheses with the value -1. For example, the expression exp[] = ( (a+b)- (c+d) is replaced by the string "-10a+/b1-0c+d1.

**Input:**

* The first line includes the number of test cases T.
* The following lines each enter a test. Each test is an exp[] expression.
* Constraints: 1≤T≤100; 1≤length(exp)≤103.

**Output:**

* Output the results of each test line by line.

**Examples:**

|  |  |
| --- | --- |
| **Input:** | **Output:** |
| 3  ((a)  (a))  (((abc))((d))))) | -10a1  0a1-1  000abc1100d111-1-1 |

**E2K04 - STACK - 2**

**Độ khó:**1

**Mô tả đề bài:**

Given a balanced expression exp[], find if it contains duplicate parenthesis or not. A set of parenthesis are duplicate if the same subexpression is surrounded by multiple parenthesis. Examples with the expression exp[]= ((a+b)+((c+d))) gives us the result “Yes” because the subexpression "c+d" is surrounded by two pairs of brackets;  with the expression exp[]= ((a+b)+(c+d))  gives us the result “No” because No subsexpression is surrounded by duplicate brackets.

**Input:**

* The first line includes the number of test cases T.
* The following lines each enter a test. Each test is an exp[] expression.
* Constraints: 1≤T≤100; 1≤length(exp)≤103.

**Output:**

* Output the results of each test line by line.

**Examples:**

|  |  |
| --- | --- |
| **Input:** | **Output:** |
| 3  ((a+b)+((c+d)))  (((a+(b)))+(c+d))  ((a+b)+(c+d)) | Yes  Yes  No |

**E2K05 - BFS - 1**

**Độ khó:**1

**Mô tả đề bài:**

Given a number **N,**and some operations that can be performed, the task is to find the minimum number of moves to convert **N** to 0. In one move operation, one of the following can be performed:

* Increment or decrement the value of N by 1.
* Multiply the value of N by -1.
* Divide the value of N by 2 if N is even.
* Reduce the value of N to √N if N is a perfect square.

Examples with N = 50 gives us the results  is 6 moves (50 (/2)->25()->5(-1) -> 4 (/2) ->2(-1) ->1(-1) ->0.

**Input:**

* The first line includes the number of test cases T.
* The following lines each enter a test. Each test is an integer N.
* Constraints: 1≤T≤100; 2000≤N≤2000.

**Output:**

* Output the results of each test line by line.

**Examples:**

|  |  |
| --- | --- |
| **Input:** | **Output:** |
| 3  50  -50  75 | 6  7  8 |

**E2K06 - BFS - 2**

**Độ khó:**1

**Mô tả đề bài:**

Given two integers N and M, the task is to find the sequence of the minimum number of operations required to convert the number N into M such that in one operation N can be performed:

* Added (N = N + N)
* Subtracted as (N = N – N).
* Multiplied as (N = N\*N).
* Divided as (N = N/N).

Examples with N = 7, M = 392 gives us the results  is 3 operations: 7+7 (14) -> 196 (14\*14) -> 392 (196+196). with N = 7, M = 9 gives us the results  is -1 because of no possible sequence of operations to convert N to M.

**Input:**

* The first line includes the number of test cases T.
* The following lines each enter a test. Each test is a pair of integer N, M.
* Constraints: 1≤T≤100; 1≤N, M≤105.

**Output:**

* Output the results of each test line by line.

**Examples:**

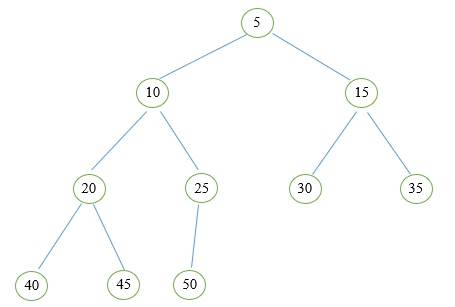
|  |  |
| --- | --- |
| **Input:** | **Output:** |
| 2  7 392  7  9 | 3  -1 |

**E2K07 - TREE - 1**

**Độ khó:**2

**Mô tả đề bài:**

Given array A[] and an array A[] with n elements. Taking the element A[0] as the root node, let's build an additional operation of nodes A[1], A[2],.., A[n-1] such that we get a binary tree that is almost complete binary tree . For example, with array A[] = {5, 10, 15, 20, 25, 30, 35, 40, 45, 50} we get the following binary tree:



**Input:**

* The first line includes the number of test cases T.
* The following lines enter the test cases. Each test consists of 2 lines: the first line input n numbers, the next line input n numbers A[i].
* Constraints: 1≤T≤100; 1≤N, A[i]≤104.

**Output:**

* Output the inoder travelsal of the almost complete binary tree.

**Examples:**

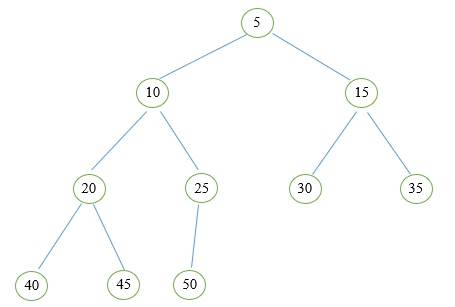
|  |  |
| --- | --- |
| **Input:** | **Output:** |
| 2  10  5 10 15 20 25 30 35 40 45 50  10  1 2 3 4 5 6 7 8 9 10 | 40 20 45 10 50 25 5 30 15 35  8 4 9 2 10 5 1 6 3 7 |

**E2K08 - TREE - 2**

**Độ khó:**2

**Mô tả đề bài:**

Given array A[] and an array A[] with n elements. Taking the element A[0] as the root node, let's build an additional operation of nodes A[1], A[2],.., A[n-1] such that we get a binary tree that is almost complete binary tree . For example, with array A[] = {5, 10, 15, 20, 25, 30, 35, 40, 45, 50} we get the following binary tree:



**Input:**

* The first line includes the number of test cases T.
* The following lines enter the test cases. Each test consists of 2 lines: the first line input n numbers, the next line input n numbers A[i].
* Constraints: 1≤T≤100; 1≤N, A[i]≤104.

**Output:**

* Output the postoder travelsal of the almost complete binary tree.

**Examples:**

|  |  |
| --- | --- |
| Input: | Output: |
| 2  10  5 10 15 20 25 30 35 40 45 50  10  1 2 3 4 5 6 7 8 9 10 | 40 45 20 50 25 10 30 35 15 5  8 9 4 10 5 2 6 7 3 1 |

**E2K09 - THE MAXIMUM SUBSET OF THE TREE**

**Độ khó:**3

**Mô tả đề bài:**

Given a tree that has N nodes, each node has a weight of C[i]. Your task is to find a subset of the tree such that:

(1) No two nodes in the subset are directed connected, which means there is no edge in the selected subset.

(2) The sum of the subset is maximum.

**Input:**

The first line is the number of nodes N (N <= 100 000).

The next line contains N integers C[i] that are the weight of the nodes (-1000 <= C[i] <= 1000).

On the N-1 following line, each line contains two numbers u and v describing an undirected edge from u to v.

**Output:**

Print the maximum sum of the chosen subset.

|  |  |
| --- | --- |
|  |  |

**Example:**

|  |  |
| --- | --- |
| Input | Output |
| 5  1 -2 2 3 -1  1 2  2 3  2 4  1 5 | 6 |
| 5  1 -2 2 3 8  1 2  2 3  2 4  1 5 | 13 |

**Explanation:**

In the first test case, the selected subset is {1, 3, 4}.

In the second dataset, the answer is {3, 4, 5}.

**E2K10 - NON-OVERLAPPING SUBSETS**

**Độ khó:**3

**Mô tả đề bài:**

You are giving an array A[] of N distinct integers. Your task is to find all integers x that, there exists two **non-overlapping** subsets in the arrays that each of them has the sum of all elements equal to x.

A subset of an array is created by removing some (but not all) elements of the array (or not removing any at all).

Two subsets are non-overlapping if there are no elements appearing on both subsets.

**Input:**

* The first line consists of an integer N – the size of array A[] (1 ≤ N ≤ 18).
* The second line consists of N distinct integers A1; A2; …; AN (1 ≤ Ai ≤ 30) – the elements of array A[].

**Output:**

* The first line consists of an integer m – the number of x satisfying the demands.
* The second line consists of m distinct integers, denoting all possible value of x. These integers must be printed in ascending order.

**Example:**

|  |  |
| --- | --- |
| **Input** | **Output** |
| 4  1 3 4 2 | 3  3 4 5 |
| 1  4 | 0 |