# Experiment - 2.1

## Stacks and Queues

|  |  |
| --- | --- |
| **Student Name**: SANSKAR AGRAWAL | **UID:** 20BCS5914 |
| **Branch**: CSE | **Section/Group:** 806/B |
| **Semester**: 5th |  |
| **Subject Name**: CC Lab |  |

### Aim/Overview of the practical:

You have three stacks of cylinders where each cylinder has the same diameter, but they may vary in height. You can change the height of a stack by removing and discarding its topmost cylinder any number of times.

Find the maximum possible height of the stacks such that all of the stacks are exactly the same height. This means you must remove zero or more cylinders from the top of zero or more of the three stacks until they are all the same height, then return the height.

**Example**

H1 = [1,2,1,1]

H2 = [1,1,2]

H3 = [1,1]

There are 4, 3 and 2 cylinders in the three stacks, with their heights in the three arrays. Remove the top 2 cylinders from H1(heights = [1, 2]) and from H2 (heights = [1, 1]) so that the three stacks all are 2 units tall. Return 2 as the answer.

**Note:** An empty stack is still a stack.

### Task to be done/ Which logistics used:

Return the height of stack when they are equalized. Approach : using stack,loop and conditional statements.

### Steps for experiment/practical/Code:

#include <bits/stdc++.h> using namespace std; string ltrim(const string &); string rtrim(const string &); vector<string> split(const string &); int equalStacks(vector<int> h1, vector<int> h2, vector<int> h3) { int s1 = h1.size(), s2 = h2.size(), s3 = h3.size(), sum1 = 0, sum2 = 0, sum3 = 0;

int i, j; for(i = 0; i < s1; i++) sum1

+= h1[i]; for(i = 0; i

< s2; i++) sum2

+= h2[i]; for(i = 0; i

< s3; i++) sum3

+= h3[i];

int height = 0;

while(!h1.empty() && !h2.empty() && !h3.empty())

{

if(sum1 == sum2 && sum2 == sum3)

{ height

= sum1; return height;

}

if (sum1 >= sum2 && sum1 >= sum3)

{

sum1 -= h1.back(); h1.pop\_back();

}

else if (sum2 >= sum1 && sum2 >= sum3)

{

sum2 -= h2.back(); h2.pop\_back();

}

else

{

sum3 -= h3.back(); h3.pop\_back();

} }

return height;

}

int main() {

int n1,n2,n3; cin>>n1>>n2>>n3; vector<int>h1; vector<int>h2; vector<int>h3;

for(int i=0;i<n1;i++)

{ int x; cin>>x; h1.push\_back(x);

}

for(int i=0;i<n2;i++)

{ int

y; cin>>y;

h2.push\_back(y);

}

for(int i=0;i<n3;i++)

{ int z; cin>>z; h3.push\_back(z);

}

reverse(h1.begin(), h1.end());

reverse(h2.begin(), h2.end());

reverse(h3.begin(), h3.end()); cout<<equalStacks(h1,h2,h3);

}

### Observations/Discussions/ Complexity Analysis:

Time Complexity : O(n) .

Space Complexity : O(1) because no extra space has been used.

### Result/Output/Writing Summary:



**Learning outcomes (What I have learnt):**

* 1. Learnt about stacks.
  2. Got an overview of the type of questions on hacker-rank.
  3. Get to know about crucial test cases.
  4. Got an understanding about referencing of stacks.

**Evaluation Grid (To be created as per the SOP and Assessment guidelines by the faculty):**

|  |  |  |  |
| --- | --- | --- | --- |
| Sr. No. | Parameters | Marks Obtained | Maximum Marks |
| 1. |  |  |  |
| 2. |  |  |  |
| 3. |  |  |  |
|  |  |  |  |

# Experiment - 2.2

## Stacks and Queues

|  |  |
| --- | --- |
| **Student Name**: SANSKAR AGRAWAL | **UID:** 20BCS5914 |
| **Branch**: CSE | **Section/Group:** 806/B |
| **Semester**: 5th |  |
| **Subject Name**: CC Lab |  |

### Aim/Overview of the practical:

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:

1: If we take 2 integers a and b where ,N=a X b (a!=1,b!=1) , then we can change N =max(a,b) 2: Decrease the value of N by 1 .

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

### Task to be done/ Which logistics used:

Return minimum number of moves required to reduce the value of N to 0. Approach : using queue and loop.

### Steps for experiment/practical/Code:

#include <map> #include <cmath> #include <queue> #include <cstdio> #include <vector>

#include <iostream> #include

<algorithm>

using namespace std; #define NSIZE 1000000

vector< vector<int> > ar(NSIZE+1);

bool primes[NSIZE+1];

int cache[NSIZE], dcache[NSIZE];

void gen\_primes() {

for (int i = 2; i <= NSIZE; i++) {

}

for (int i = 1; i <= NSIZE; i++) { bool prime = true; for (int j = 2; j\*j<= i; j++) { if (i % j

== 0) { prime = false; break;

}

}

primes[i] = prime;

}

}

void get\_a(int n)

{ if

(primes[n])

return; if (dcache[n] != -1) return;

else

dcache[n] = 1;

for (int i = 2; i\*i <= n; i++) { if (n % i == 0) { int v = n / i; if (v == 1 || i == 1) continue; v = v > i ? v : i;

ar[n].push\_back(v);

}

}

}

int w[NSIZE];

void fill\_cache(int steps, int number, int start)

{ int indx = start;

int st = 1;

if (cache[start] != -1) st = cache[start] + 1;

while(1) { int pos = w[indx];

cache[pos] = st++;

indx = pos; if (st == steps + 1)

break;

}

}

int \*q; int qpos = 0; int qend = 0; int steps = 0; int cal\_steps(int v) { for (int i = 0; i <= v; i++) w[i]

= -1; qpos = 0;

qend = 0;

steps = 0; q[qend++] = v;

q[qend++] = -1;

while(1) { int val

= q[qpos++]; if (val

== -1) { steps

++; q[qend++] =

-1;

val = q[qpos++];

}

if (val == 0) { return steps;

}

get\_a(val); for (int i = 0; i < ar[val].size(); i++) { if

(w[ar[val][i]] == -1)

w[ar[val][i]] = val; int tmp\_val = ar[val][i];

q[qend++] = tmp\_val;

}

val -= 1; q[qend++] = val;

}

return -1;

}

int main() { std::ios\_base::sync\_with\_stdio (false);

q = new int[NSIZE \* 19];

int n, v; cin >> n; int max

= n; for (int i = 0; i < NSIZE; i++) { cache[i] = -1; dcache[i] = -1;

}

gen\_primes();

while(n--) { cin >>

v; if (v == 0) { cout << "0" << endl;

continue;

}

cout << cal\_steps(v) << endl;

}

return 0;

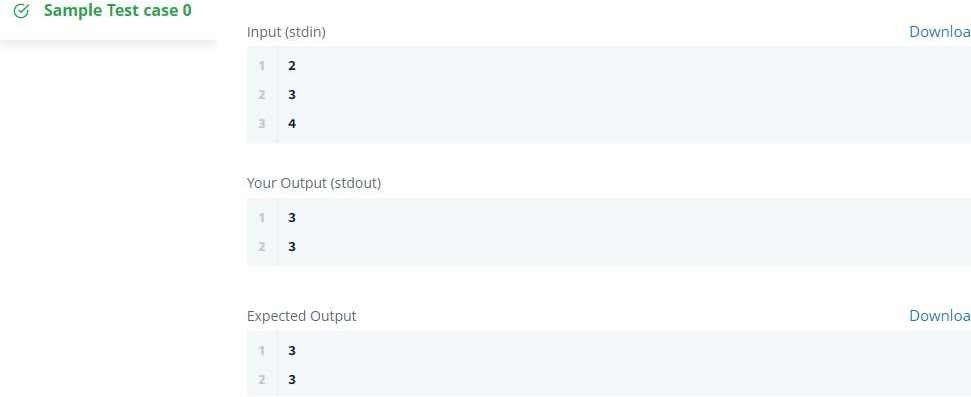
}

### Observations/Discussions/ Complexity Analysis:

Time Complexity : O(n).

Space Complexity : O(1) because no extra space has been used.

### Result/Output/Writing Summary:



**Learning outcomes (What I have learnt):**

* 1. Learnt about stacks.
  2. Got an overview of the type of questions on hacker-rank.
  3. Get to know about crucial test cases.
  4. Got an understanding about referencing of stack.

**Evaluation Grid (To be created as per the SOP and Assessment guidelines by the faculty):**

|  |  |  |  |
| --- | --- | --- | --- |
| Sr. No. | Parameters | Marks Obtained | Maximum Marks |
| 1. |  |  |  |
| 2. |  |  |  |
| 3. |  |  |  |
|  |  |  |  |