

Experiment-3.1

Student Name: Mukund Jaiswal

UID: 21BCS3407

Branch: CSE

Section/Group: 602-B

Semester: 5th

Date of Performance: 19/10/23

Subject Name: AP-I

Subject Code: 21CSP-314

Aim- Dynamic Programming: Implement the problems based on Dynamic Programming

Objectives- The objective of this experiment is to understand the concept of dynamic programming problems.

Problem1: <https://www.hackerrank.com/challenges/construct-the-array/problem?isFullScreen=true>

Problem2: <https://www.hackerrank.com/challenges/equal/problem?isFullScreen=true>

Description-

Dynamic programming is a powerful technique in computer science and mathematics for solving problems by breaking them down into smaller subproblems and solving each subproblem only once, storing the results for future reference.

The definition of dynamic programming says that it is a technique for solving a complex problem by first breaking into a collection of simpler subproblems, solving each subproblem just once, and then storing their solutions to avoid repetitive computations.

Code:

1.

Change Theme Language C++14



```
1  #include <bits/stdc++.h>
2
3  using namespace std;
4
5  string ltrim(const string &);
6  string rtrim(const string &);
7  vector<string> split(const string &);
8  const int MOD = 1000000007;
9
10 long countArray(int n, int k, int x) {
11     long ways[2][2];
12     ways[0][0] = 1;
13     ways[0][1] = 0;
14     bool fillSecond = true;
15     for (int i = 0; i < n-1; i++) {
16         ways[fillSecond][0] = (ways[!fillSecond][1] * (k - 1)) % MOD;
17         ways[fillSecond][1] = (ways[!fillSecond][1] * (k - 2) + ways[!fillSecond][0]) % MOD;
18         fillSecond = !fillSecond;
19     }
20
21     long answer;
22     if (x == 1) {
23         answer = (ways[fillSecond][1] * (k - 1)) % MOD;
24     }
25     else {
26         answer = (ways[fillSecond][1] * (k - 2) + ways[fillSecond][0]) % MOD;
27     }
28     return answer;
29 }
30
```

```
32 int main()
33 {
34     ofstream fout(getenv("OUTPUT_PATH"));
35
36     string first_multiple_input_temp;
37     getline(cin, first_multiple_input_temp);
38
39     vector<string> first_multiple_input = split(rtrim(first_multiple_input_temp));
40
41     int n = stoi(first_multiple_input[0]);
42
43     int k = stoi(first_multiple_input[1]);
44
45     int x = stoi(first_multiple_input[2]);
46
47     long answer = countArray(n, k, x);
48
49     fout << answer << "\n";
50
51     fout.close();
52
53     return 0;
54 }
55
56 > string ltrim(const string &str) { ...
65 }
66
67 > string rtrim(const string &str) { ...
76 }
```

```
..  
78 vector<string> split(const string &str) {  
79     vector<string> tokens;  
80  
81     string::size_type start = 0;  
82     string::size_type end = 0;  
83  
84     while ((end = str.find(" ", start)) != string::npos) {  
85         tokens.push_back(str.substr(start, end - start));  
86  
87         start = end + 1;  
88     }  
89  
90     tokens.push_back(str.substr(start));  
91  
92     return tokens;  
93 }
```

2.

Change Theme Language C++14

```
1  #include <bits/stdc++.h>  
2  
3  using namespace std;  
4  
5  string ltrim(const string &);  
6  string rtrim(const string &);  
7  vector<string> split(const string &);  
8  
9  int equal(vector<int> arr) {  
10     int ans = INT_MAX;  
11     int min_elem = *min_element(arr.begin(), arr.end());  
12     for(int base=0; base<3; base++){  
13         int temp = 0;  
14         for(int i=0; i<arr.size(); i++){  
15             int dist = arr[i] - (min_elem - base);  
16             int steps = dist/5 + (dist%5)/2 + (dist%5)%2;  
17             temp+=steps;  
18         }  
19         ans = min(ans, temp);  
20     }  
21     return ans;  
22 }  
23  
24 int main()  
25 {  
26     ofstream fout(getenv("OUTPUT_PATH"));  
27  
28     string t_temp;  
29     getline(cin, t_temp);  
30 }
```

```
31     int t = stoi(ltrim(rtrim(t_temp)));
32
33 > for (int t_itr = 0; t_itr < t; t_itr++) {...
55     }
56
57     fout.close();
58
59     return 0;
60 }
61
62 ✓ string ltrim(const string &str) {
63     string s(str);
64
65     s.erase(
66         s.begin(),
67         find_if(s.begin(), s.end(), not1(ptr_fun<int, int>(isspace)))
68     );
69
70     return s;
71 }
72
73 ✓ string rtrim(const string &str) {
74     string s(str);
75
76     s.erase(
77         find_if(s.rbegin(), s.rend(), not1(ptr_fun<int, int>(isspace))).base(),
78         s.end()
79     );
80
81     return s;
```

```
71 }
72
73 ✓ string rtrim(const string &str) {
74     string s(str);
75
76     s.erase(
77         find_if(s.rbegin(), s.rend(), not1(ptr_fun<int, int>(isspace))).base(),
78         s.end()
79     );
80
81     return s;
82 }
83
84 ✓ vector<string> split(const string &str) {
85     vector<string> tokens;
86
87     string::size_type start = 0;
88     string::size_type end = 0;
89
90     while ((end = str.find(" ", start)) != string::npos) {
91         tokens.push_back(str.substr(start, end - start));
92
93         start = end + 1;
94     }
95
96     tokens.push_back(str.substr(start));
97
98     return tokens;
99 }
```

Outcome-

Problem 1 outcome

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

✓ Sample Test case 0
Download

Input (stdin)

1	4 3 2
---	-------

Your Output (stdout)

1	3
---	---

Expected Output

1	3
---	---

Download

Problem 2 Outcome

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

✓ Sample Test case 0
Download

Input (stdin)

1	5
2	110
3	53 361 188 665 786 898 447 562 272 123 229 629 670 848 994 54 822 46 208 17 449 302 466 832 931 778 156 39 31 777 749 436 138 289 453 276 539 901 839 811 24 420 440 46 269 786 101 443 832 661 460 281 964 278 465 247 408 622 638 440 751 739 876 889 380 330 517 919 583 356 83 959 129 875 5 750 662 106 193 494 120 653 128 84 283 593 683 44 567 321 484 318 412 712 559 792 394 77 711 977 785 146 936 914 22 942 664 36 400 857
4	82
5	520 862 10 956 498 956 991 542 523 664 378 194 76 90 753 868 837 830 932 814 616 78 103 882 452 397 899 488 149 108 723 22 323 733 330 821 41 322 715 917 986 93 111 63 535 864 931 372 47 215 539 15 294 642 897 98 391 796 939 540 257 662 562 580 747 893 401 789 215 468 58 553 561 169 616 448 385 900 173 432 115 712

✓ Sample Test case 1

Learning Outcomes-

1. Learnt about the dynamic programming.
2. Learnt about the some dynamic programming problem.