Print all path with minimum Cost In C++

```
#include <iostream>
#include <vector>
#include <queue>
using namespace std;

struct Pair {
    string psf; // path so far
    int i;      // current row index
    int j;      // current column index

     Pair(string psf, int i, int j) {
        this->psf = psf;
        this->i = i;
        this->j = j;
    }
};

void printAllPaths(vector<vector<int>>&
arr) {
    int m = arr.size();
    int n = arr[0].size();

    // dp array to store minimum cost to
reach each cell
    vector<vector<int>> dp(m,
vector<int>(n, 0));

    // Initialize dp table
    dp[m-1][n-1] = arr[m-1][n-1];
    for (int i = m - 2; i >= 0; i--) {
        dp[i][n-1] = arr[i][n-1] + dp[i + 1][n -
1];
    }
    for (int j = n - 2; j >= 0; j--) {
        dp[m-1][j] = arr[m-1][j] + dp[m - 1][j +
1];
    }
    for (int i = m - 2; i >= 0; i--) {
        for (int j = n - 2; j >= 0; j--) {
            dp[i][j] = arr[i][j] + min(dp[i][j + 1],
dp[i + 1][j]);
        }
    }

    // Minimum cost to reach the top-left
corner
    cout << dp[0][0] << endl;

    // Queue to perform BFS
    queue<Pair> q;
    q.push(Pair("", 0, 0));
```

Dry Run of the Code

1. **Initial Setup**: The arr grid:

   Copy code
   {1, 2, 3, 4}
   {5, 6, 7, 8}
   {9, 10, 11, 12}
   {13, 14, 15, 16}

2. **Filling the DP Table**:
   - The bottom-right corner dp[3][3] is initialized as arr[3][3] = 16.
   - The last row and column are filled:
     - dp[3][2] = 16 + 12 = 28
     - dp[3][1] = 28 + 8 = 36
     - dp[3][0] = 36 + 4 = 40
     - dp[2][3] = 16 + 12 = 28
     - dp[2][2] = 28 + 8 = 36
     - dp[2][1] = 36 + 7 = 43
     - dp[2][0] = 43 + 5 = 48
     - And so on...
   - Final dp table:

     Copy code
     46 50 54 58
     51 55 59 62
     59 63 67 72
     60 64 68 72

3. **BFS to Find All Paths**:
   - The BFS starts from dp[0][0], trying to find paths with minimum cost.
   - The BFS explores possible moves using the dp values:
     - It starts by pushing the initial position (0, 0) with the path so far as "" into the queue.
     - After processing all possible paths, the minimum cost path HHHVVV is printed.

```
    while (!q.empty()) {
        Pair rem = q.front();
        q.pop();

        if (rem.i == m - 1 && rem.j == n - 1) {
            cout << rem.psf << endl; // print
path when reaching the bottom-right
corner
        } else if (rem.i == m - 1) {
            q.push(Pair(rem.psf + "H", rem.i,
rem.j + 1)); // go right
        } else if (rem.j == n - 1) {
            q.push(Pair(rem.psf + "V", rem.i +
1, rem.j)); // go down
        } else {
            if (dp[rem.i][rem.j + 1] < dp[rem.i +
1][rem.j]) {
                q.push(Pair(rem.psf + "H", rem.i,
rem.j + 1)); // go right
            } else if (dp[rem.i][rem.j + 1] >
dp[rem.i + 1][rem.j]) {
                q.push(Pair(rem.psf + "V", rem.i
+ 1, rem.j)); // go down
            } else {
                q.push(Pair(rem.psf + "V", rem.i
+ 1, rem.j)); // go down
                q.push(Pair(rem.psf + "H", rem.i,
rem.j + 1)); // go right
            }
        }
    }
}

int main() {
    vector<vector<int>> arr = {
        {1, 2, 3, 4},
        {5, 6, 7, 8},
        {9, 10, 11, 12},
        {13, 14, 15, 16}
    };

    printAllPaths(arr);
    return 0;
}
```

Output:-

46
HHHVVV