Print all path with minimum Cost In C++

```cpp
#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 Minimum Cost Path Problem

We will compute the **dynamic programming (DP) table** step-by-step to ensure that we get the minimum cost sum **46** for the given matrix.

Given Input Matrix (`arr`):

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

Step 1: Understanding the DP Approach

1. **Base Case**: The last cell (`dp[3][3]`) is the same as `arr[3][3] = 16`.
2. **Filling Last Row (Right to Left)**:
   - `dp[i][j] = arr[i][j] + dp[i][j+1]`
3. **Filling Last Column (Bottom to Top)**:
   - `dp[i][j] = arr[i][j] + dp[i+1][j]`
4. **Filling the Rest (Bottom-Up, Right-to-Left)**:
   - `dp[i][j] = arr[i][j] + min(dp[i+1][j], dp[i][j+1])`

Step 2: Construct DP Table Step-by-Step

1. Initialize `dp[3][3]` (Bottom-Right Cell)

```
dp[3][3] = arr[3][3] = 16
```

2. Fill the Last Row (Right to Left)

dp[i][j]=arr[i][j]+dp[i][j+1]dp[i][j] = arr[i][j] + dp[i][j+1]dp[i][j]=arr[i][j]+dp[i][j+1]

i=3	**j=3**	**j=2 (15+16)**	**j=1 (14+31)**	**j=0 (13+45)**

```
    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;
}
```

arr	16	15	14	13
dp	16	**31**	**45**	**58**

3. Fill the Last Column (Bottom to Top)

dp[i][j]=arr[i][j]+dp[i+1][j]dp[i][j] = arr[i][j] + dp[i+1][j]dp[i][j]=arr[i][j]+dp[i+1][j]

i=2	**j=3 (12+16)**	**j=2**	**j=1**	**j=0**
arr	12	11	10	9
dp	**28**	-	-	-
i=1	**j=3 (8+28)**	**j=2**	**j=1**	**j=0**
arr	8	7	6	5
dp	**36**	-	-	-
i=0	**j=3 (4+36)**	**j=2**	**j=1**	**j=0**
arr	4	3	2	1
dp	**40**	-	-	-

4. Fill the Rest of the DP Table

dp[i][j]=arr[i][j]+min(dp[i+1][j],dp[i][j+1])dp[i][j] = arr[i][j] + min(dp[i+1][j], dp[i][j+1])dp[i][j]=arr[i][j]+min(dp[i+1][j],dp[i][j+1])

i=2	**j=2 (11+min(31,28))**	**j=1 (10+min(41,38))**	**j=0 (9+min(45, 40))**
arr	11	10	9
dp	**39**	**38**	**40**
i=1	**j=2 (7+min(39, 36))**	**j=1 (6+min(38, 44))**	**j=0 (5+min(45, 43))**
arr	7	6	5
dp	**43**	**44**	**45**

i=0	j=2 (3+min(43, 40))	j=1 (2+min(41, 44))	j=0 (1+min(45, 43))
arr	3	2	1
dp	**43**	**45**	**46**

Final DP Table

46	45	43	40
45	44	43	36
40	38	39	28
58	45	31	16

✅ **Minimum Cost Path Sum = `46` (Matches G++ Output)**

Step 3: Extracting All Paths

Now, we use BFS (`queue<Pair>`) to **trace all paths** from `(0,0)` to `(3,3)` following the minimum cost. The paths may vary but should sum up to `46`.

1. **Move Right `H`** if `dp[i][j+1]` is smaller.
2. **Move Down `V`** if `dp[i+1][j]` is smaller.
3. **If both are equal, try both paths (`H` and `V`)**.

Possible Paths (`psf` values in BFS)

```
V V V H H H  (Down-Down-Down-Right-Right-Right)
```

Output:-
46
HHHVVV