

## Print all path with max gold In C++

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

struct Pair {
    int i, j;
    string psf;

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

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

    // dp array to store maximum gold
    // collected to reach each cell
    vector<vector<int>> dp(m,
vector<int>(n, 0));

    // Initialize dp array for the last column
    for (int i = 0; i < m; i++) {
        dp[i][n - 1] = arr[i][n - 1];
    }

    // Fill dp array using dynamic
    // programming approach
    for (int j = n - 2; j >= 0; j--) {
        for (int i = 0; i < m; i++) {
            int maxGold = dp[i][j + 1]; //
            Maximum gold by going right from current
            cell
            if (i > 0) {
                maxGold = max(maxGold, dp[i -
1][j + 1]); // Maximum gold by going
                diagonal-up-right
            }
            if (i < m - 1) {
                maxGold = max(maxGold, dp[i +
1][j + 1]); // Maximum gold by going
                diagonal-down-right
            }
            dp[i][j] = arr[i][j] + maxGold; //
            Total gold collected to reach current cell
        }
    }
}
```

**Given Input Matrix (arr):**

```
3 2 3 1
2 4 6 0
5 0 1 3
9 1 5 1
```

**Step 1: Initialize dp Table**

- **Copy the last column (j = 3) from arr to dp:**

```
0 0 0 1
0 0 0 0
0 0 0 3
0 0 0 1
```

**Step 2: Fill dp Table from Right to Left**

**Column 2 (j = 2)**

Each  $dp[i][j] = arr[i][j] + \max(dp[i][j+1], dp[i-1][j+1], dp[i+1][j+1])$

```
0 0 4 1 → 3 + max(1) = 4
0 0 9 0 → 6 + max(3,0) = 9
0 0 6 3 → 1 + max(3,1) = 6
0 0 8 1 → 5 + max(3) = 8
```

**Column 1 (j = 1)**

```
0 11 4 1 → 2 + max(4,9) = 11
0 13 9 0 → 4 + max(9,6) = 13
0 9 6 3 → 0 + max(6,8) = 9
0 14 8 1 → 1 + max(8) = 14
```

**Column 0 (j = 0)**

```
13 11 4 1 → 3 + max(11,13) = 13
15 13 9 0 → 2 + max(13,9) = 15
18 9 6 3 → 5 + max(9,14) = 18 ✓ (Expected
max value)
23 14 8 1 → 9 + max(14) = 23
```

**Step 3: Find Maximum Gold in Column 0**

|   |   |
|---|---|
| <pre> }  // Find the maximum gold collected in the first column int maxGold = dp[0][0]; int maxRow = 0; for (int i = 1; i &lt; m; i++) {     if (dp[i][0] &gt; maxGold) {         maxGold = dp[i][0];         maxRow = i;     } }  // Print the maximum gold collected cout &lt;&lt; maxGold &lt;&lt; endl;  // Queue to perform BFS for path tracing queue&lt;Pair&gt; q; q.push(Pair(maxRow, 0, to_string(maxRow))); // Start from the cell with maximum gold in the first column  // BFS to print all paths with maximum gold collected while (!q.empty()) {     Pair rem = q.front();     q.pop();      if (rem.j == n - 1) {         cout &lt;&lt; rem.psf &lt;&lt; endl; // Print path when reaching the last column     } else {         int currentGold = dp[rem.i][rem.j];         int rightGold = dp[rem.i][rem.j + 1];         int diagonalUpGold = (rem.i &gt; 0) ? dp[rem.i - 1][rem.j + 1] : -1;         int diagonalDownGold = (rem.i &lt; m - 1) ? dp[rem.i + 1][rem.j + 1] : -1;          // Add paths to queue based on the direction with maximum gold         if (rightGold == currentGold - arr[rem.i][rem.j + 1]) {             q.push(Pair(rem.i, rem.j + 1, rem.psf + " H")); // Move horizontally to the right         }         if (diagonalUpGold == currentGold - arr[rem.i - 1][rem.j + 1]) {             q.push(Pair(rem.i - 1, rem.j + 1, rem.psf + " LU")); // Move diagonally up- right         }     } } </pre> | <ul style="list-style-type: none"> <li>• The <b>maximum gold collected</b> is 18 at <b>row 2</b>.</li> </ul> <p><b>Step 4: Find All Paths (Using BFS)</b></p> <p>Starting from dp[2][0] = 18:</p> <ol style="list-style-type: none"> <li>1. dp[2][1] = 9</li> <li>2. dp[3][1] = 14</li> <li>3. dp[3][2] = 8</li> <li>4. dp[3][3] = 1</li> </ol> <p><b>Valid Path:</b></p> <p>2 → LD → 3 → LU → 3 → H → 1</p> <p><b>Final Output</b></p> <p>Maximum Gold: 18<br/>Path: 2 LD 3 LU 3 H 1</p> |
|---|---|

```

        if (diagonalDownGold ==
currentGold - arr[rem.i + 1][rem.j + 1]) {
            q.push(Pair(rem.i + 1, rem.j + 1,
rem.psf + " LD")); // Move diagonally down-
right
        }
    }
}
}

int main() {
    vector<vector<int>> arr = {
        {3, 2, 3, 1},
        {2, 4, 6, 0},
        {5, 0, 1, 3},
        {9, 1, 5, 1}
    };

    printMaxGoldPath(arr);

    return 0;
}

```

**Output:-**

18