Algorithms

20 September 2023 10:52 AM

Longest valid parenthesis.

```
Stack<int> s;   int len=0;
  s.push(-1);
for(int i=0; i<str.length(); i++)
{
    if(str[i]=='(')
    {
      s.push(i);
    }
    else
    {
      if(!s.empty())
      {
        s.pop();
      }
      if(!s.empty())
      {
        len = max(len, i-s.top());
      }
      else
      {
        s.push(i);
      }
}
```

}

Shortest path in Binary Maze

```
bool isSafe(vector<vector<int>> &mat, vector<vector<bool>> &visited, int x, int y)
{
    return (x >= 0 && x < mat.size() && y >= 0 && y < mat[0].size()) &&
           mat[x][y] == 1 && !visited[x][y];
}


void findShortestPath(vector<vector<int>> &mat, vector<vector<bool>> &visited,int i, int j, int x, int y, int &min_dist, int dist){
    if (i == x && j == y){
        min_dist = min(dist, min_dist);
        return;
    }

    visited[i][j] = true;

    if (isSafe(mat, visited, i + 1, j)) {
        findShortestPath(mat, visited, i + 1, j, x, y, min_dist, dist + 1);
    }

    if (isSafe(mat, visited, i, j + 1)) {
        findShortestPath(mat, visited, i, j + 1, x, y, min_dist, dist + 1);
    }

    if (isSafe(mat, visited, i - 1, j)) {
        findShortestPath(mat, visited, i - 1, j, x, y, min_dist, dist + 1);
    }

    if (isSafe(mat, visited, i, j - 1)) {
        findShortestPath(mat, visited, i, j - 1, x, y, min_dist, dist + 1);
    }

    visited[i][j] = false;
}


int findShortestPathLength(vector<vector<int>> &mat, pair<int, int> &src, pair<int, int> &dest){
    if (mat.size() == 0 || mat[src.first][src.second] == 0 ||
        mat[dest.first][dest.second] == 0)
        return -1;

    int row = mat.size();
    int col = mat[0].size();

    vector<vector<bool>> visited;
    visited.resize(row, vector<bool>(col));

    int dist = INT_MAX;
    findShortestPath(mat, visited, src.first, src.second, dest.first, dest.second, dist, 0);

    if (dist != INT_MAX)
        return dist;
    return -1;
}
```

Maximum Area in a Histogram.

```
   stack<int> s;
```

```
int max_area = 0;
int tp;
int area_with_top;
int i = 0;
while (i < n) {
    if (s.empty() || hist[s.top()] <= hist[i])
        s.push(i++);

    else {
        tp = s.top(); // store the top index
        s.pop(); // pop the top

        area_with_top
            = hist[tp]
              * (s.empty() ? i : i - s.top() - 1);


        if (max_area < area_with_top)
            max_area = area_with_top;
    }
}

 while (s.empty() == false) {
    tp = s.top();
    s.pop();
    area_with_top
        = hist[tp] * (s.empty() ? i : i - s.top() - 1);

    if (max_area < area_with_top)
        max_area = area_with_top;
}

return max_area;
```