**2. Labyrinth**

**Code is perfect but there is tle or mle as I used backtracking method**

1. **#include** <bits/stdc++.h>
2. **using** **namespace** std;
3. **bool** issafe(**int** x,**int** y,**int** n,**int** m,**vector**<**vector<char>**>& ma ,**vector**<**vector<int>**>& visited){
4. **if**(x>=0 && y>=0 && x<n && y<m && ma[x][y] != **'#'** && visited[x][y] == 0){
5. **return** **true**;
6. }
7. **return** **false**;
8. }
10. **void** solve(**int** x1,**int** y1,**int** x2,**int** y2,**int** n,**int** m,**vector**<**vector<char>**>& ma ,**vector**<**vector<int>**>& visited, string path,**vector<string>**& ans){
12. **if**(x1 == x2 && y1==y2){
13. ans.push\_back(path);
14. **return** ;
15. }
17. **int** dx[] = {-1,0,0,1};
18. **int** dy[] = {0,-1,1,0};
19. **char** dir[] = {**'U'**,**'L'** ,**'R'** ,**'D'**};
21. **for**(**int** i=0;i<4;i++){
22. **int** nx = x1 + dx[i];
23. **int** ny = y1 + dy[i];
25. **if**(issafe(nx,ny,n,m,ma,visited)){
26. visited[nx][ny] = 1;
27. solve(nx,ny,x2,y2,n,m,ma,visited,path+dir[i],ans);
28. visited[nx][ny] = 0; **//backtracking**
29. }
30. }
31. }
33. **int** main(){
34. **int** n,m;
35. cin>>n>>m;
36. **vector**<**vector<char>**>ma(n,**vector<char>**(m));
37. **int** x1,x2,y1,y2;
38. **for**(**int** i=0;i<n;i++){
39. **for**(**int** j=0;j<m;j++){
40. cin>>ma[i][j];
41. **char** x = ma[i][j];
42. **if**(x == **'A'**){
43. x1 = i;
44. y1 = j;
45. }**else** **if** (x == **'B'**){
46. x2 = i;
47. y2= j;
48. }
49. }
50. }
51. **vector**<**vector<int>**>visited(n,**vector<int>**(m,0));
52. string path = **""**;
53. **vector<string>**ans;
55. **if**(ma[x1][y1] == **'A'** && ma[x2][y2] == **'B'**){
56. visited[x1][y1] = 1;
57. solve(x1,y1,x2,y2,n,m,ma,visited,path,ans);
58. }**else**{
59. cout<<**"no"**<<endl;
60. **return** 0;
61. }
62. **if**(ans.size() == 0){
63. cout<<**"no"**<<endl;
64. }**else**{
65. cout<<**"yes"**<<endl;
66. }
68. **return** 0;
70. }