# ABC PATH

## Big-O Blue - Lecture 06: DFS

# Tóm tắt đề bài

# Tóm tắt đề bài

- Bạn được cho bảng chữ cái hai chiều. Tìm đường đi **dài nhất** của các chữ cái liên tiếp, bắt đầu từ ‘A’.
- Đường đi có thể đi từ một ký tự trong bảng đến bất kí ký tự liền kề nó (ngang, dọc, chéo).

# Mô tả Input/Output

## Input

Có nhiều bộ test. Mỗi bộ test có định dạng:

- Dòng đầu tiên gồm 2 số nguyên *H*, *W* (1 ≤ H, W ≤ 50) lần lượt là chiều cao và rộng của bảng.
- H dòng kế tiếp, mỗi dòng gồm W ký tự in hoa

Kết thúc toàn bộ bộ test là cặp “0 0”.

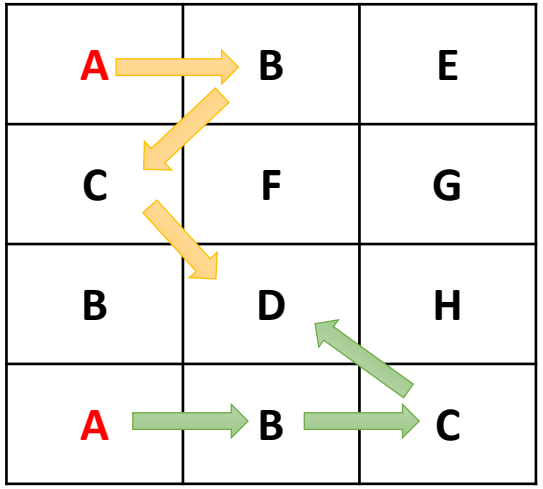
## Output

Với mỗi bộ test in ra “Case C: X” với **C** là số thứ tự bộ test và **X** là kết quả bài toán.

# Giải thích ví dụ

# Ví dụ

| Input | Output |
|---|---|
| 4 3<br>ABE<br>CFG<br>BDH<br>ABC<br>0 0 | Case 1: 4 |




- Có  nhiều cách đi từ A → D
- Không có đường nào đi từ A → E

➔ **Đường đi dài nhất là 4** (A → D).

# Hướng dẫn giải

# Nhận xét

- Việc duyệt A, sau đó duyệt tìm đến B, rồi tìm C, v.v. giống như *duyệt theo độ sâu bảng chữ cái* dựa trên các chữ cái
- Đề bài yêu cầu tìm đường đi dài nhất

➔ áp dụng DFS cho bài toán này

# Ý tưởng

- Do đường đi cần tìm *chỉ có thể bắt đầu từ 'A'* → duyệt qua từng phần tử trong bảng, nếu nó là ký tự 'A' thì thực hiện DFS từ đây
- Trong lúc DFS cần chú ý các điểm sau:
  - Từ 1 ô, nếu muốn đi qua ô liền kề ô đó (ngang, dọc, chéo) thì ô kề phải thỏa điều kiện chứa kí tự liền sau của ô đang xét
  - Trong quá trình duyệt, đếm độ sâu của phép duyệt này được bao nhiêu ký tự xem lần duyệt nào cho ra đường dài nhất

# Các bước giải (đối với 1 testcase)

B1: Đọc vào W, H và lưu lại bảng chữ cái

B2: Duyệt qua bảng chữ cái, nếu chữ cái đang xét là ‘A’

- DFS từ ô này, trong quá trình DFS:

  + Xét ô kề với ô đang được DFS:

    - Nếu không tồn tại ô kề hoặc ô kề đã được duyệt → bỏ qua
    - Nếu ô kề chứa chữ cái liền sau → DFS từ ô kề

  + Cập nhật độ dài đường đi

- So sánh đường đi vừa DFS với đường đi dài nhất để cập nhật

B3: In kết quả theo định dạng “Case C: X”

**Độ phức tạp:** **O(t * W * H)** với t là số lượng testcase, W, H lần lượt là kích thước của bảng

# Mã giả

# Mã giả

```
#main
case = 0
while True:
 case += 1
 read h, w
 if h == 0 and w == 0:
  break
 read matrix a #grid of letters
```

```
res = 0
visited = [[False]*w for I = 1 → h]
for i = 1 → h
 for j = 1 → w
  if a[i][j] == 'A'
   res = max(res, DFS(i, j))
print "Case " + case + ": " + res
printline
```

# Mã giả

```
dx = [0, -1, -1, -1, 0, 1, 1, 1]
dy = [-1, -1, 0, 1, 1, 1, 0, -1]
#dfs function
function DFS(x, y)
  st = []
  push (x,y) to st
  visited[x][y] = True
  max_dist = 1
```

```
while st is not empty():
  (x, y) = pop from st
  for i = 1 → 8:
    u = x + dx[i]
    v = y + dy[i]
    if 0 <= u < h and 0 <= v < w
      and not visited[u][v]
      and ord(a[x][y]) == ord(a[u][v])-1
        visited[u][v] = True
        push (u,v) to st
        max_dist = max(max_dist, ord(a[u][v])-65+1)
return max_dist
```

# Thank you