Week 17

2024-10-31

알파벳

BFS + Set

큐에 들어가는 정보는 x, y, state

dist[20][20] 에서 거리를 재고

set<int> st[20][20] 에서는 해당 좌표에서 가능한 알파벳 묶음 종류 (state) 를 캐시

알파벳

```
queue<tuple<int,int,int>> q;
q.push({0, 0, 1 << (a[0][0] - 'A')});
st[0][0].insert(1 << (a[0][0] -'A'));
dist[0][0] = 1;
```

알파벳

```
int ans = 1;
while(!q.empty()) {
    auto [x, y, state] = q.front();
    q.pop();
    for(int k = 0; k < 4; k++) {
        int nx = x + dx[k];
        int ny = y + dy[k];

        if(nx < 0 || nx >= n || ny < 0 || ny >= m) continue;

        int c = a[nx][ny] - 'A';

        if(state & 1 << c) continue;

        int nstate = state | 1 << c;
        if(st[nx][ny].count(nstate)) continue;

        st[nx][ny].insert(nstate);
        dist[nx][ny] = max(dist[nx][ny], dist[x][y] + 1);
        ans = max(ans, dist[nx][ny]);
        q.emplace(nx, ny, nstate);
    }
}
cout << ans << '\n';
```

알파벳

A	B	C	D	E	F	G	..
B	C	D	E	F	G	..	
C	D	E	F	G	..		
D	E	F	G	..			
E	F	G	..				
F	G	..					
G	..						
...							

트리

규칙을 찾아서 재귀로 트리 그래프를 새로 만듦

3	6	5	4	8	7	1	2

5	6	8	4	3	1	2	7

go(0, 8, 0, 8)

→ preorder(0, 8) 과 inorder(0, 8)

트리

규칙을 찾아서 재귀로 트리 그래프를 새로 만듦

3	6	5	4	8	7	1	2

5	6	8	4	3	1	2	7

go(0, 8, 0, 8)
→ preorder(0, 8) 과 inorder(0, 8)

트리의 전위순회의 첫 번째 값은 루트임
해당 루트를 중위순회에서 찾으면
해당 루트의 왼쪽 부분 트리와
오른쪽 부분 트리를 찾을 수 있음

전위순회에서 루트 바로 다음은 왼쪽 자식 ,
왼쪽 자식개수만큼 이동하고 나오는
바로 다음 자식은 오른쪽 자식

→ 왼쪽 자식 트리와 오른쪽 자식 트리에 재귀적으로 들어갈 수 있음

트리

3	6	5	4	8	7	1	2

5	6	8	4	3	1	2	7

3	6	5	4	8	7	1	2

5	6	8	4	3	1	2	7

go(0, 8, 0, 8)

→ preorder(0, 8) 과 inorder(0, 8) 가 매치

L

R

3	6	5	4	8	7	1	2

5	6	8	4	3	1	2	7

3 의 왼쪽 자식은 preorder 왼쪽의 첫번째 3 → 4

오른쪽 자식은 preorder 오른쪽의 첫번째 3 → 7

L 4 개 , R 3 개

왼쪽 자식 부분 트리로 이동

트리

3	6	5	4	8	7	1	2

5	6	8	4	3	1	2	7

3	6	5	4	8	7	1	2

5	6	8	4	3	1	2	7

왼쪽 자식 트리로 이동

go(0, 8, 0, 8)

→ preorder(0, 8) 과 inorder(0, 8)

L

R

6	5	4	8

5	6	8	4

go(1, 5, 0, 4)

→ preorder(1, 5) 과 inorder(0, 4) 가 매치

6 의 왼쪽 자식은 preorder 왼쪽 첫번째 6 → 5

오른쪽 자식은 preorder 오른쪽 첫번째 6 → 4

트리

3	6	5	4	8	7	1	2

5	6	8	4	3	1	2	7

6	5	4	8

5	6	8	4

왼쪽자식으로 이동

go(1, 5, 0, 4)

→ preorder(1, 5) 과 inorder(0, 4) 가 매치

L

R

5

go(2, 3, 0, 1)

→ preorder(2, 3) 과 inorder(0, 1) 가 매치

5

자식없음 끝

트리

3	6	5	4	8	7	1	2

5	6	8	4	3	1	2	7

6	5	4	8

5	6	8	4

오른쪽자식으로 이동

go(1, 5, 0, 4)

→ preorder(1, 5) 과 inorder(0, 4) 가 매치

4	8

8	4

go(3, 5, 2, 4)

→ preorder(3, 5) 과 inorder(2, 4) 가 매치

왼쪽 자식 하나만 있는 걸 확인할 수 있음

4 의 왼쪽 자식은 8

오른쪽 자식은 없음

8

8

하나라서 자식 없음

트리

```
void go(int l1, int r1, int l2, int r2) {
    if(r1 - l1 == 1) return ;
    int mid = find(in.begin(), in.end(), pre[l1]) - in.begin()
    int lsize = mid - l2;
    int rsize = r2 - mid - 1;

    if(lsize > 0) {
        g[pre[l1]].push_back(pre[l1 + 1]);
        go(l1 + 1, l1 + 1 + lsize, l2, mid);
    }

    if(rsize > 0) {
        g[pre[l1]].push_back(pre[l1 + lsize + 1]);
        go(l1 + 1 + lsize, r1, mid + 1, r2);
    }

}
```