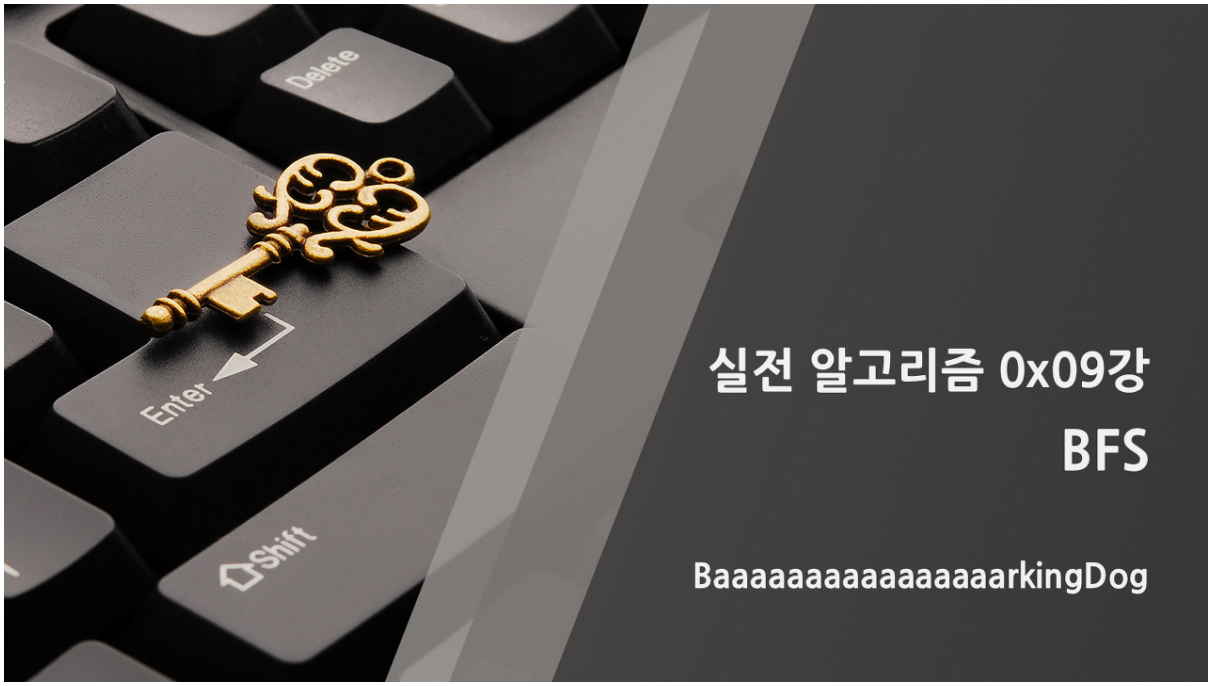
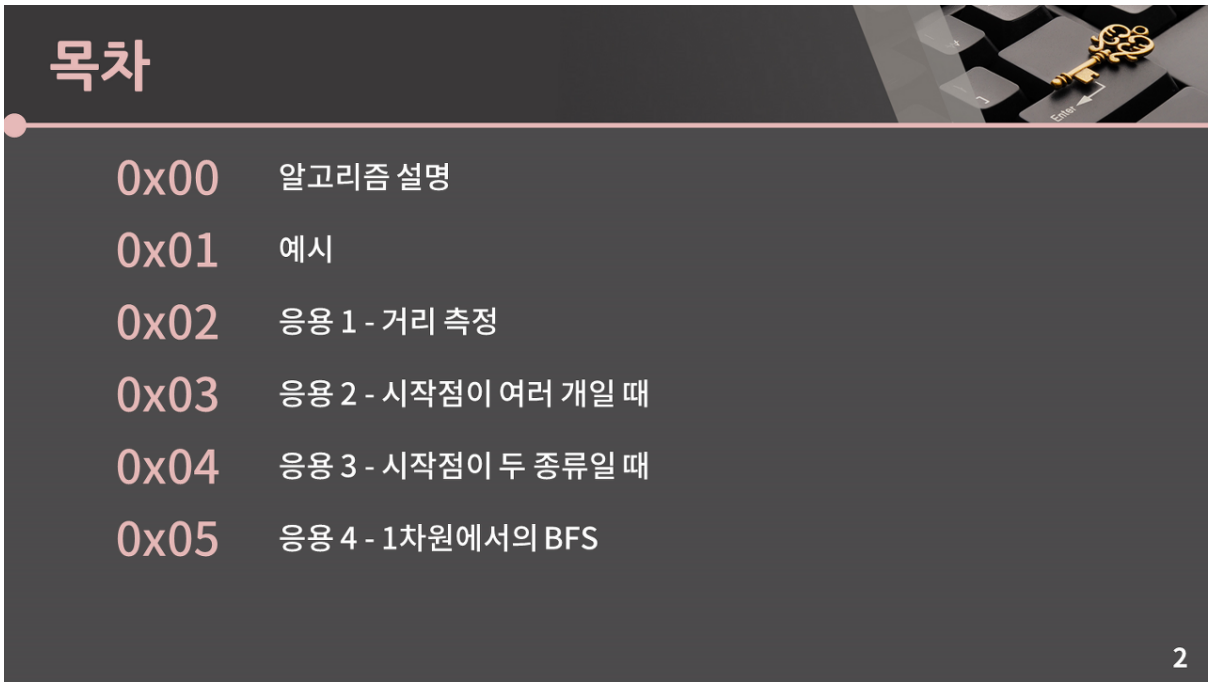
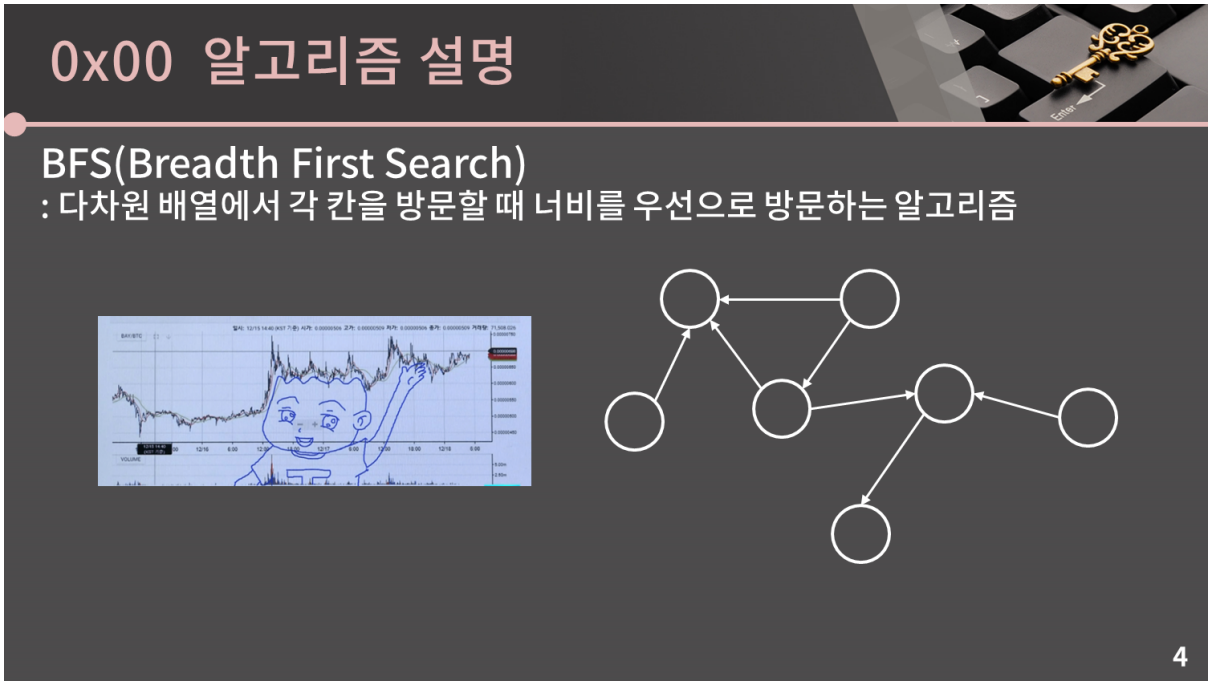
바킹독 0x09 BFS

태그	
공개여부	☑
날짜	
작성일자	





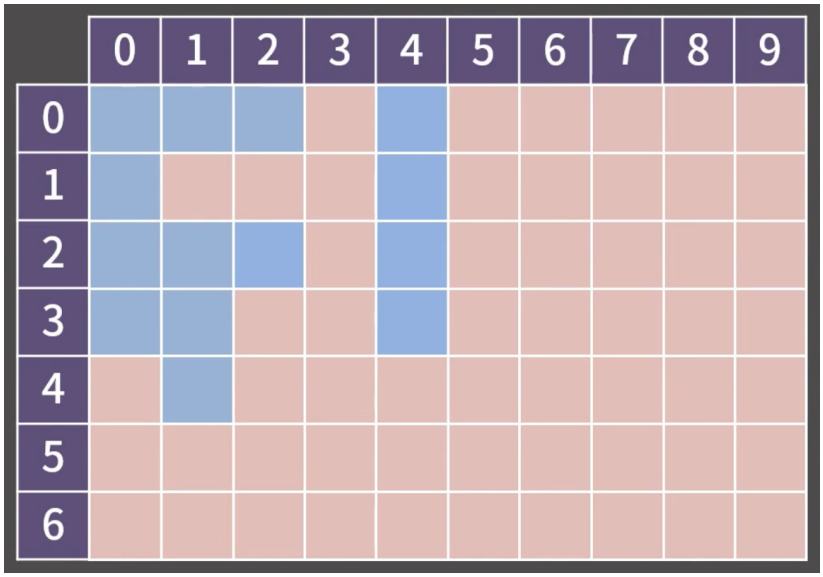


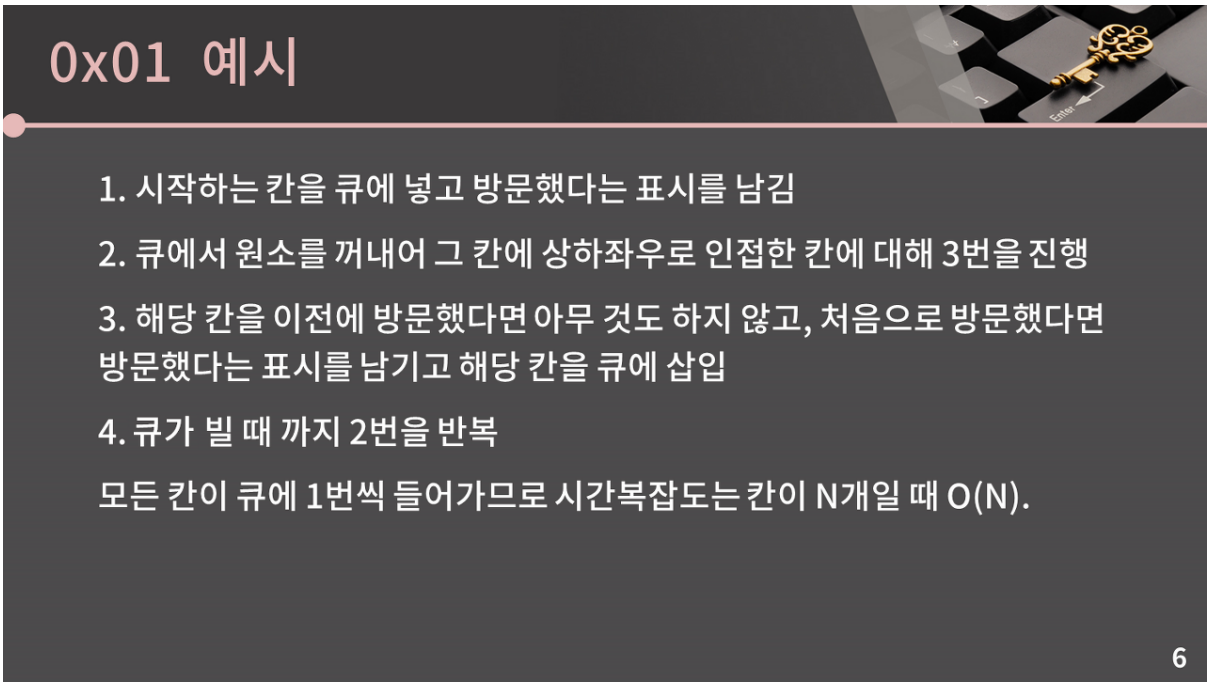
원래 BFS는 그래프라는 자료구조에서 모든 노드를 방문하기 위한 알고리즘입니다. 여기서 말하는 그래프는 우리가 흔히 아는 왼쪽과 같은 형태의 그래프가
아니라 오른쪽 모양의 그래프이고, 정확한 정의는 정점과 간선으로 이루어진 자료구조입니다.

그렇기 때문에 BFS를 정확하게
이해하려면 그래프 자료구조에 대한 이해가 선행되어야 하는데 그건 배보다 배꼽이 더 큰 느낌입니다. **그래서 BFS를 엄밀하게 정의할 수는 없지만, 실제로 어떻게 동작하는지를 보면서 다차원 배열에서의 BFS를 이해해보도록 하겠습니다.**

예시

(0, 0)과 상하좌우로 이어진 파란색 칸의 크기를 판단하는 예시를 한다고 해보겠습니다.





BFS의 **시간복잡도**를 생각해보면 방문 표시를 남기기 때문에 모든 칸은 큐에 1번씩만 들어가게 됩니다. 그렇기 때문에 시간복잡도는 칸이 N개일 때 O(N)이 됩니다. 만약 행이 R개이고 열이 C개이면 O(RC)가 될 것입니다.

BFS의 구현

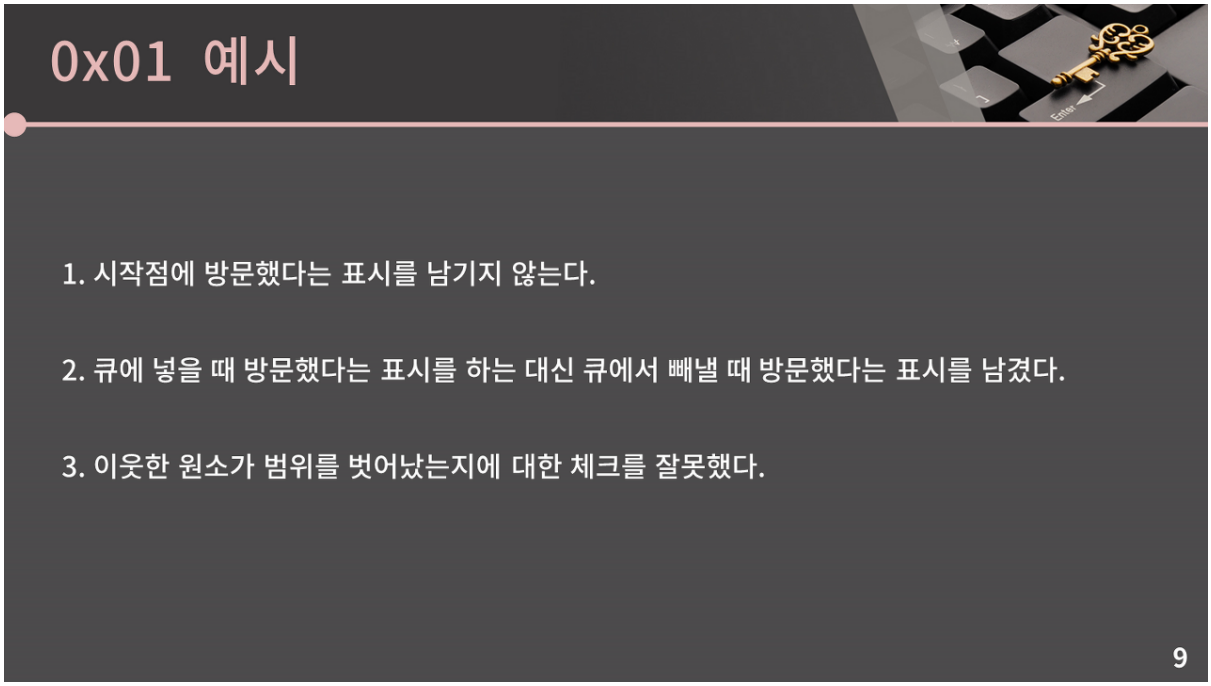
```
//BFS SKELETON
#include <bits/stdc++.h>
using namespace std;
#define X first
#define Y second // pair 함에서 first, second를 줄여서 쓰기 위해서 사용

int board[502][502] =
{{1,1,1,0,1,0,0,0,0,0},
 {1,0,0,0,1,0,0,0,0,0},
 {1,1,1,0,1,0,0,0,0,0},
 {1,1,0,0,1,0,0,0,0,0},
 {0,1,0,0,0,0,0,0,0,0},
 {0,0,0,0,0,0,0,0,0,0},
 {0,0,0,0,0,0,0,0,0,0} }; // 1이 파란 칸, 0이 빨간 칸에 대응

 bool vis[502][502]; // 해당 칸 방문여부를 저장하는 변
 int n = 7, m = 10; // n = 행의 수, m = 열의 수
 int dx[4] = {1, 0, -1, 0};
 int dy[4] = {0, 1, 0, -1}; // 상하좌우에 있는 칸을 쉽게 접근하기 위해 사용하는 변수

int main(void){

  ios::sync_with_stdio(0);
  cin.tie(0);
  queue<pair<int,int>> Q; // int값 한 쌍(한 지점, 좌표)을 저장하는 큐 선언
  vis[0][0] = 1; // (0, 0)을 방문했다
  Q.push({0,0}) // 큐에 시작점인 (0, 0) 삽입
  while(!Q.empty())
  {
```

```
     pair<int,int> cur = Q.front(); Q.pop(); // cur = 현재 위치를 저장하는 변수, 큐에서 좌표값을 가져온 뒤 큐에서 제거해준다.
      for(int dir = 0; dir < 4; dir++) // cur에서 인접한 상하좌우 칸을 살펴볼 것이다.
      {
       int nx = cur.X + dx[dir]
       int ny = cur.Y + dx[dir] // cur에서 인접한 상하좌우칸의 좌표가 nx, ny에 들어간다.

       //예외처리
       if(nx < 0 || nx >= n || ny < 0 || ny >= m) continue; // board 범위 밖일 경우(행은 n, 열은 m줄 존) 넘어감
       if(vis[nx][ny] || board[nx][ny] != 1) continue; // 이미 방문한 칸이거나 파란 칸이 아닐 경우 넘어감

       vis[nx][ny] = 1; // 예외처리에 걸리지 않는 경우! (nx, ny)를 방문했다고 명시해주고
       Q.push({nx,ny}); // 인접했던 그 칸의 좌표값을 큐에 넣어준다.
      }
   }

}



/*
about BFS
1. 시작하는 지점을 큐에 push하고 방문했다는 표시를 남김.
2. 큐에서 행렬(좌표)값이 저장된 원소를 꺼내고, 해당 지점의 상하좌우로 인접한 칸에 대하여 3번 진행
3. 해당 칸을 이전에 방문했다면 아무 동작도 하지 않는다, 만약 방문한 적이 없다면 방문표시를 남기고 해당 칸의 좌표를 큐에 삽입한다.
4. 큐가 빌 때까지 2번을 반복한다.
모든 칸이 큐에 1번씩 들어가므로 시간복잡도는 칸이 N개일 때 O(n)이 된다.
board, visit, n/m, dx/dy, pair queue, pair current, nx/ny
*/
```

BFS를 구현할 때 큐에 좌표를 넣어야 하는데, 이때 pair를 쓸 것입니다.

utility 헤더에 있는 pair인데, pair를 이용하면 두 자료형을
묶어서 가지고 다닐 수 있습니다. make_pair로 값을 넣어줄 수도 있고, C++11 이상에서는 그냥 중괄호를 써서 쉽게
해결할 수 있습니다.

값의 접근은 각각 first, second를 부름으로서 가능하고 또 pair에는 미리 대소 관계가 설정되어 있어서 편합니다. 알아서 앞쪽의 값을 먼저 비교하고, 이후 뒤쪽의 값을 비교합니다.



> BFS 구현 시 주의사항

> 1. 시작점을 큐에 넣긴하는데 정작 방문했다는 표시를 남기지 않은 채로 진행하는 경우가 있습니다. 이렇게 되면 시작점을 두 번 방문할 수가 있습니다.
>
> 2. 큐에 넣을 때 해당 칸에 방문했다는 표시를 남기지 않고 큐에서 빼낼 때 남기는 경우인데, 이렇게 되면 같은 칸이 큐에 여러 번 들어가게 되어서 시간 초과나 메모리 초과가 발생할 수 있습니다. 특히 이건 보통 예제로 주는 작은 케이스에서는 잘 돌아가다가 실제 제출을 했을 때 터지는 경우가 많기 때문에 주의해야 합니다.
>
> 3. 앞의 코드에서 있던 nx, ny가 배열 바깥으로 벗어났는지에 대한 루틴을 아예 빼먹었거나, 아니면 이상하게 구현을 한 상황을 말합니다.