Week 21

2024-12-05

소수 상근수

에라토스테네스의 체로
소수를 구하며
상근수인지 체크를 함

소수 상근수

상근수인지를 check 배열에 기록을 해 두면 더 빠르게 구할 수 있음

```
int go(int now) {
    if(now == 1 || check[now] == 1) return 1;
    if(check[now] == 0) return 0;

    check[now] = 0;

    int tmp = 0, x = now;
    while(x != 0) {
        tmp += (x % 10) * (x % 10);
        x /= 10;
    }

    return check[now] = go(tmp);
}
```

상근수인지 체크하는 함수

-1 => 아직 안해봄

0 => 상근수 아님

1 => 상근수

소수 상근수

go(700)
check[700] = 0
700 => 49
check[700] = go(49)

go(49)
check[49] = 0;
49 => 97
check[49] = go(97)

go(97)
check[97] = 0;
97 => 130
check[97] = go(130)

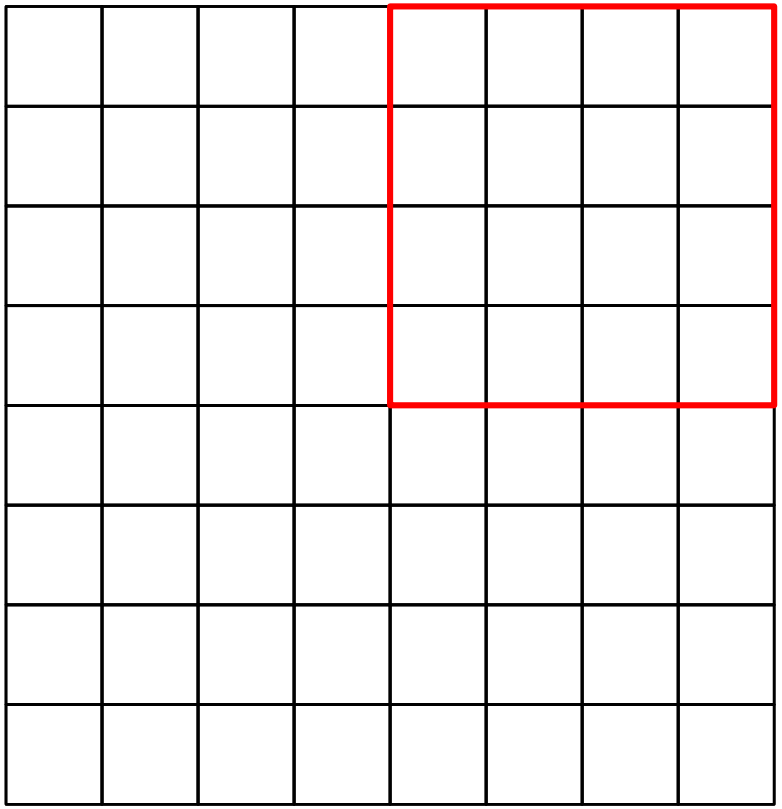
go(130)
check[130] = 0;
130 => 10
check[130] = go(10);

go(10)
check[10] = 0;
10 => 1
check[10] = go(1);

```
int go(int now) {
    if(now == 1 || check[now] == 1) return 1;
    if(check[now] == 0) return 0;

    check[now] = 0;

    int tmp = 0, x = now;
    while(x != 0) {
        tmp += (x % 10) * (x % 10);
        x /= 10;
    }

    return check[now] = go(tmp);
}
```

상근수인지 체크하는 함
-1 => 아직 안해봄
0 => 상근수 아님
1 => 상근수

Z



¼ 구역의
변의 길이 = (1 << (n – 1))

칸 수 = 변의길이 제곱
= (1 << ((n – 1) * 2))

```
int ans = 0;
while(n > 0) {
    int rc = 1 << (n - 1);
    int cnt = 1 << ((n - 1) * 2);
    if(r >= rc) {
        if(c >= rc) {
            ans += cnt * 3;
            r -= rc;
            c -= rc;
        } else {
            ans += cnt * 2;
            r -= rc;
        }
    } else {
        if(c >= rc) {
            ans += cnt;
            c -= rc;
        } else {

        }
    }
    n -= 1;
}
```

테트로미노

이런식으로 다 만들어놓고 5 중 for 문돌리기

테트로미노 개수
N
M
테트로미노 [i].size
테트로미노 [i][0].size()

```
vector<vector<vector<int>>> pos = {
    {
        {1, 1, 1, 1}
    },
    {
        {1},
        {1},
        {1},
        {1}
    },
    {
        {1, 1},
        {1, 1}
    },
    {
        {1, 1, 1},
        {0, 0, 1}
    },
    {
        {1, 1, 1},
        {1, 0, 0}
    },
    {
        {1, 1, 1},
        {0, 1, 0}
    },
```

테트로미노

```
int ans = 0;
for(vector<vector<int>> v : pos) {
    int r = v.size();
    int c = v[0].size();
    for(int i = 0; i <= n - r; i++) {
        for(int j = 0; j <= m - c; j++) {
            int tmp = 0;
            for(int x = 0; x < r; x++) {
                for(int y = 0; y < c; y++) {
                    tmp += a[i + x][j + y] * v[x][y];
                }
            }
            ans = max(ans, tmp);
        }
    }
}
cout << ans << '\n';
```

소수의 곱

pq 에 수들 전부 넣고
pq 의 탑 => 제일 작은 값을 꺼냄
=> 제일 작은 값과 수들을 곱해서 넣음

꺼낼때는 바로 직전에 꺼낸 수보다 작은 수가 나오면 무시

메모리 초과

K - 1;

```
ll now = -1;
for(int i = 0; i < k; i++) {
    while(pq.top() <= now) {
        pq.pop();
    }
    ll top = pq.top();
    pq.pop();
    now = top;
    for(ll x : a) {
        pq.push(top * x);
    }
}
```

소수의 곱

이전 방법은 겹치는 수들을 계속
넣기때문에 메모리가부족해지는것

pq 에 값을 넣을 때
현재 꺼낸 값 % 소수 == 0
전까지만 넣는 방법
으로 메모리 초과를 해결할 수 있음

```
for(int i = 0; i < k - 1; i++) {
    ll top = pq.top();
    pq.pop();
    for(ll x : a) {
        pq.push(top * x);
        if(top % x == 0) break;
    }
}
cout << pq.top() << '\n';
```

소수의 곱

2, 5, 7 이 주어졌다고 하면

2 x 2, 2 x 5, 2 x 7
5 x 2, 5 x 5, 5 x 7
7 x 2, 7 x 5, 7 x 7

if(now % x == 0) break;

특정 수를 소수의 곱으로 표현할 때
→ a1 x a2 x a3 x a4 일때
a1, a2 등의 값이 내림차순으로
정렬되도록 강제하는것

```
for(int i = 0; i < k - 1; i++) {
    ll top = pq.top();
    pq.pop();
    for(ll x : a) {
        pq.push(top * x);
        if(top % x == 0) break;
    }
}
cout << pq.top() << '\n';
```

Bumped!!

다익스트라인데 공짜인 단방향 간선들이 중간중간 있고

이 공짜 단방향간선들은 한 번만 사용할 수 있음
Dist[50000][2];

Dist[x][cnt]
x 에 도착하는데 걸리는 시간인데 , 비행기를 cnt 번 탔을때 : 라고생각하면됨

dist 는 큰 수로 초기화하고 dist[start][0] 만 0 으로 초기화하고 다익스트라
우선순위큐에 넣는 값을 [거리 , 현재까지 비행기 탄 수 , 노드번호]