MUA

Maratones uniandes

MATRIX EXPONENTIATION

ISIS 2801

Matrix power

Given a square matrix M we want to calculate its **p** power fast

$$M^0 = I$$

$$M^p = \prod_{i=1}^{p} M$$

Matrix power

Given a square matrix M we want to calculate its **p** power fast

$$M^0 = I$$

$$M^p = \prod_{i=1}^{p} M$$

$$O(pn^\omega)$$

Number exponentiation

$a^0 = I$

$a^1 = a$

$a^p = a^{p-1} \times a$ if p is odd

$a^p = (a^{p/2})^2$ if p is even

Number exponentiation

$a^0 = I$

$a^1 = a$

$a^p = a^{p-1} \times a$ if p is odd

$a^p = (a^{p/2})^2$ if p is even

$O(log(p))$

Number exponentiation

```
double exp(double base, double p) {
    if( p == 0) return 1;
    else if(p == 1) return base;
    else {
       double res = exp(base, p/2);
       res *= res;
       if(p % 2 == 1) res *= base;
       return res;
    }
}
```

Matrix exponentiation

```
struct Matrix { int mat[MAX_N][MAX_N]; };

Matrix matrixMultiplication(Matrix a, Matrix b) {
    Matriz ans;
    for(int i=0; i < MAX_N; i++)
        for(int j=0; j< MAX_N; j++
            for(ans.mat[i][j] = k = 0; k< MAX_N; k++)
                 ans.mat[i][j] += a.mat[j][k] * b.mat[k][j];
    return ans;
}

Matrix matrixPow(Matrix base, int p) {
    Matrix ans;
    for(int i=0; i<MAX_N; i++)
        for(int j=0; j<MAX_N; j++)
            ans.mat[i][j] = (i==j);
    while(p){
        if(p & 1) ans = matrixMultiplication(ans, base);
        base = matrixMultiplication(base, base);
        p >>= 1;
    }
    return ans;
}
```

Matrix exponentiation

```
struct Matrix { int mat[MAX_N][MAX_N]; };

Matrix matrixMultiplication(Matrix a, Matrix b) {
    Matriz ans;
    for(int i=0; i < MAX_N; i++)
        for(int j=0; j< MAX_N; j++
            for(ans.mat[i][j] = k = 0; k< MAX_N; k++)
                 ans.mat[i][j] += a.mat[j][k] * b.mat[k][j];
    return ans;
}

Matrix matrixPow(Matrix base, int p) {
    Matrix ans;
    for(int i=0; i<MAX_N; i++)
        for(int j=0; j<MAX_N; j++)
            ans.mat[i][j] = (i==j);
    while(p){
        if(p & 1) ans = matrixMultiplication(ans, base);
        base = matrixMultiplication(base, base);
        p >>= 1;
    }
    return ans;
}
```

$O(n^3log(p))$