백준 문제 풀이 - 단계별로 풀어보기3

상태	진행 중
작성일시	@2023년 9월 1일 오후 6:13
유형	**분할정복에 대한 백준 문제풀이 노트. (ver. 파이썬)**
텍스트	백준을 풀어보자!! https://www.acmicpc.net/user/yuyunjae03

백준 2630번 색종이 만들기 https://www.acmicpc.net/problem/2630
li[0] += a[0]
li[1] += a[1]
아래 코드와 차이점은 이부분인 것 같다. 나는 재귀호출 때마다 li라는 리스트를 만들어주고 4방향으로 잘라서 그 곳의 색깔들을 각각 저장한 후, 리턴을 해줬다. 하지만, 아래 코드를 보면 정말 간단하게도 외부에서 list를 선언한 후, 내가 [1, 0], [0, 1]을 리턴하는 경우에 그냥 해당하는 인덱스에 += 1을 해주면서 깔끔하게 처리했다. 따라서, 더 짧은 실행 시간이 소요된 것 같다.

```
#my code 64m/s
from sys import stdin

n = int(stdin.readline().rstrip())
paper = [list(map(int, stdin.readline().split())) for _ in ra


def cut_paper(left: int, right: int, top: int, bottom: int):
    li = [0, 0]
    flag = 0
    if paper[top][left] == 1:
        flag = 1
    for i in range(top, bottom):
        for j in range(left, right):
            if (flag and paper[i][j] != 1) or (flag == 0 and
                flag = -1
                break
        if flag == -1:
            break
    if flag == 0:
```

```
        return [1, 0]
    elif flag == 1:
        return [0, 1]
    else:
        a = cut_paper(left, (left + right) // 2, top, (top +
        li[0] += a[0]
        li[1] += a[1]
        a = cut_paper((left + right) // 2, right, top, (top +
        li[0] += a[0]
        li[1] += a[1]
        a = cut_paper(left, (left + right) // 2, (top + botto
        li[0] += a[0]
        li[1] += a[1]
        a = cut_paper((left + right) // 2, right, (top + bott
        li[0] += a[0]
        li[1] += a[1]
    return li


print(*cut_paper(0, n, 0, n), sep='\n')
```

```
#other good code 44m/s
# 2630 - 색종이 만들기

import sys
input = sys.stdin.readline


def divide(i, j, n):

    if n == 1:
        cnt[paper[i][j]] += 1
        return

    is_match = 1
    color = paper[i][j]
    for r in range(i, i+n):
        for c in range(j, j+n):
```

```
            if paper[r][c] != color:
                is_match = 0
                break
        if not is_match:
            break

    if not is_match:
        divide(i, j, n//2)
        divide(i+n//2, j, n//2)
        divide(i, j+n//2, n//2)
        divide(i+n//2, j+n//2, n//2)

    else:
        cnt[color] += 1
        return


N = int(input())
paper = [list(map(int, input().split())) for _ in range(N)]

cnt = [0, 0]
divide(0, 0, N)

print(cnt[0])
print(cnt[1])
```

백준 1992번 쿼드트리 https://www.acmicpc.net/problem/1992
2630 색종이 만들기 문제를 응용해서 매우 빠르게 풀 수 있었다. 그런데, 정답 코드를 보며 공부 중에, 파이썬의 특징을 이용한 기가막힌 풀이방법이 있어서 가져왔다. 리스트의 덧셈 연산을 이용해서 깔끔하게 풀었음을 볼 수 있다.

```
#my code 52m/s
from sys import stdin

n = int(stdin.readline().rstrip())
paper = [stdin.readline().rstrip() for _ in range(n)]
```

```
def cut_paper(left: int, right: int, top: int, bottom: int):
    flag = 0
    if paper[top][left] == '1':
        flag = 1
    for i in range(top, bottom):
        for j in range(left, right):
            if (flag and paper[i][j] != '1') or (flag == 0 an
                flag = -1
                break
        if flag == -1:
            break
    if flag == 0:
        print(0, end='')
    elif flag == 1:
        print(1, end='')
    else:
        print('(', end='')
        cut_paper(left, (left + right) // 2, top, (top + bott
        cut_paper((left + right) // 2, right, top, (top + bot
        cut_paper(left, (left + right) // 2, (top + bottom) /
        cut_paper((left + right) // 2, right, (top + bottom) 
        print(')', end='')


cut_paper(0, n, 0, n)
print()
```

```
#other good code 40m/s
import sys
n = int(input())
a = []
for _ in range(n):
    a.append(sys.stdin.readline().strip())

def zi(x,y,k):
    if k==1:return a[x][y]
    nk = k//2
```

```
    d1,d2,d3,d4 = zi(x,y,nk),zi(x,y+nk,nk),zi(x+nk,y,nk),zi(x
    if d1 =='0' or d1 =='1':
        if d1 == d2 and d2 == d3 and d3 == d4:
            return d1
    return '('+d1+d2+d3+d4+')'

print(zi(0,0,n))
```

백준 1780번 종이의 개수 https://www.acmicpc.net/problem/1780
처음에 답을 제출하고 실행시간을 보면서 깜짝 놀랐다.. 1초가 초과하는 시간이 걸릴 줄 몰랐기 때문이다. 결과적으로는 통과였지만, 다른 pypy3 제출 코드중에서 빠른 코드를 찾아보니, 색깔이 다르다는 것을 발견하면 break가 아닌 재귀를 그 위치에서 해버리고 return을 해서 굳이 2중 반복문을 break로 빠져나가는 수고를 줄였다. 또한, size > 1일 경우에만 반복문을 실행해서 1x1일 경우는 빠르게 해당 위치에 적혀있는 숫자의 개수 += 1을 처리했다. 이런 방식이 나보다 2배정도 코드의 속도를 향상시킨 것 같다.

```
#my code pypy3(1308m/s)
from sys import stdin

n = int(stdin.readline().rstrip())
paper = [list(map(int, stdin.readline().split())) for _ in ra
count = [0, 0, 0]


def cut_paper(row: int, col: int, size: int):
    num = paper[row][col]
    for i in range(row, row + size):
        for j in range(col, col + size):
            if paper[i][j] != num:
                num = 10
                break
        if num == 10:
            break
    if -1 <= num <= 1:
        count[num + 1] += 1
        return
    elif num == 10:
```

```
        size //= 3
        cut_paper(row, col, size)
        cut_paper(row, col + size, size)
        cut_paper(row, col + 2 * size, size)
        cut_paper(row + size, col, size)
        cut_paper(row + size, col + size, size)
        cut_paper(row + size, col + 2 * size, size)
        cut_paper(row + 2 * size, col, size)
        cut_paper(row + 2 * size, col + size, size)
        cut_paper(row + 2 * size, col + 2 * size, size)
    return


cut_paper(0, 0, n)
print(*count, sep='\n')
```

```
#other good code pypy3(712m/s)
import sys


def check_color(size, start_x, start_y):
    color = paper[start_y][start_x]
    if size > 1:
        for y in range(start_y, start_y + size):
            for x in range(start_x, start_x + size):
                if paper[y][x] != color:
                    new_size = size // 3
                    for i in range(3):
                        for j in range(3):
                            check_color(new_size, start_x + n
                    return None
    res[color + 1] += 1
    return None


N = int(sys.stdin.readline().rstrip())
paper = [list(map(int, sys.stdin.readline().rstrip().split())
res = [0, 0, 0]
```

```
check_color(N, 0, 0)
print(res[0])
print(res[1])
print(res[2])
```

백준 1629번 곱셈 https://www.acmicpc.net/problem/1629
예전에 우연히 빠른 거듭제곱 알고리즘을 공부하게 됐었는데, 그와 관련된 문제라 빠르게 풀 수 있었던 것 같다. 나는 반복문을 사용했고, 아래 코드는 재귀를 이용해 구했다.

https://jehwanyoo.net/2022/01/24/빠른-거듭제곱-알고리즘-fast-power-algorithm/
→ 빠른 거듭제곱 관련 사이트..
내 노션에도 정리가 되어 있다...

```
#my code 40m/s
from sys import stdin

a, b, c = map(int, stdin.readline().split())
ans = 1
while b > 1:
    if a >= c:
        a = a % c
    if b % 2 == 0:
        b = b // 2
    elif b % 2 == 1:
        ans = ans * a % c
        b = b // 2
    a = a * a % c
if b == 1:
    ans = (ans * a) % c
print(ans)
```

```
#other good code 32m/s
a, b, c = map(int, input().split())
def check(num):
    if num == 1:
        return a % c
```

```
    else:
        left = check(num // 2)
        if num % 2 == 0:
            return (left ** 2) % c
        else:
            return (left ** 2 * a) % c
print(check(b))
```

백준 11401번 이항 계수 3 https://www.acmicpc.net/problem/11401
처음에 감을 못잡아서 nCk = n-1Ck-1 + n-1Ck를 dp로 풀다가 메모리 초과가 발생했다. 최대(4e+6) ^ 2 만큼의 공간이 필요하기 때문이다. 그래서 다른 질문요청 글을 보다가 페르마의 소정리를 이용하면 풀 수 있다는 듯한 내용을 보게 되었다. 이를 이용해 빠른 거듭제곱 알고리즘, dp를 이용해 해결했다.

https://ko.wikipedia.org/wiki/페르마의_소정리 → 페르마의 소정리
a : 정수, p : 소수 일때 a^(p - 1) % p = 1
ex) a = 3, p = 7일때, 3^6 = 729이고 이를 7로 나누면 놀랍게도 1이 나온다..
1000000007이 소수라는 것은 알고리즘 내역을 보고 알게 되었다.. 에라토스테네스의 체를 응용해서 다시 한번 구해봐야겠다.
이를 이용하면 nCk = n! / (k! * (n-k)!) = n! / (k! * (n-k)!) * 1 = (n! / (k! * (n-k)!)) * (k! * (n-k)!)^(p-1) % p = n! * (k! * (n-k)!)^(p-2) % p가 된다. 정말 충격적인 계산이다.. 역시 사람은 수학을 잘해야 하는 것 같다.
아래 코드도 비슷한 방식이나, 조금 더 깔끔하게 해결한 것 같다.

```
#my code 1000m/s pypy3(144m/s)
from sys import stdin
mod = 1000000007


def a_pow(a: int, x: int):
    if a == 1:
        return 1
    elif x <= 1:
        return (a ** x) % mod
    else:
        p = a_pow(a, x // 2)
        if x % 2 == 0:
```

```
            return p * p % mod
        else:
            return (p * p % mod) * a % mod


n, k = map(int, stdin.readline().split())
li = [0, 0]
fact = 1
if k > n - k:
    k = n - k
if n == 1 or k == 0:
    print(1)
elif k == 1:
    print(n)
else:
    for i in range(2, n + 1):
        fact = fact * i % mod
        if i == k:
            li[0] = fact
        if i == n - k:
            li[1] = fact
    print(fact * a_pow((li[0] * li[1] % mod), mod - 2) % mod)
```

```
#other good code pypy3(120m/s)
import sys
input = sys.stdin.readline
p = 1000000007
N, K = map(int, input().split())
minimum = min(N-K, K)
A, B = 1, 1
for i in range(minimum):
    A = A*(N-i) % p
    B = B*(minimum-i) % p


def power(n):
    if n == 1:
        return B % p
```

```
    else:
        temp = power(n//2)
        if n % 2 == 0:
            return (temp*temp) % p
        else:
            return (temp*temp)*B % p


res_B = power(p-2)
print((A*res_B) % p)
```

백준 10830번 행렬 제곱 https://www.acmicpc.net/problem/10830
행렬 곱셈과 관련된 문제를 처음 풀어보다보니 너무 어려웠다. 에러도 너무 많이 발생하고 빠른 거듭제곱과 행렬제곱 함수를 만들어서 해결하긴 했지만, 거의 3시간 가까이 풀지를 못했고, 힌트를 얻고 나서야 해결할 수 있었다. c++코드도 복습해보고 행렬 제곱 관련 알고리즘도 공부해보자..
#진짜 백준을 풀때마다 def를 잘 사용하지 않다보니 함수에서 생성된 list가 c처럼 함수 종료 후, 사라질 줄 알았다. 그런데, 파이썬은 2차원 list조차 return을 해보니 함수가 종료 되었어도 그대로 넘겨 줄 수 있다는 것을 다시 알게 되었다. 진짜 파이썬은 최강이다..

```
def mul_matrix(m1, m2):
    al = [[0] * n for _ in range(n)]
    for i in range(n):
        for j in range(n):
            for k in range(n):
                al[i][j] += m1[i][k] * m2[k][j]
            al[i][j] %= mod
    return al
```

행렬 곱셈에 해당하는 이부분은 정말 빠른 거듭제곱처럼 유용한 부분인 것 같다. 잘 공부해서 다음에 써먹을 수 있어야 할 것 같다.
밑의 ios_base::sync_with_stdio(false) 이부분이 궁금해서 사이트를 가져왔다.

https://nextcoder.tistory.com/10

https://codingfriend.tistory.com/21

```
#my code 40m/s
from sys import stdin
```

```
mod = 1000

n, b = map(int, stdin.readline().split())
map_n = [list(map(int, stdin.readline().split())) for _ in ra
ans = [[0] * n for _ in range(n)]
for i in range(n):
    for j in range(n):
         if i == j:
             ans[i][j] = 1


def mul_matrix(m1, m2):
    al = [[0] * n for _ in range(n)]
    for i in range(n):
        for j in range(n):
            for k in range(n):
                al[i][j] += m1[i][k] * m2[k][j]
            al[i][j] %= mod
    return al


def pow_map(m, x: int):
    if x == 1:
        return m
    else:
        if x % 2 == 1:
            k = mul_matrix(m, m)
            return mul_matrix(pow_map(k, x // 2), m)
        else:
            k = mul_matrix(m, m)
        return pow_map(k, x // 2)


ans = pow_map(map_n, b)
for i in range(n):
    for j in range(n):
        ans[i][j] = ans[i][j] % mod
```

```
for i in ans:
    print(*i, sep=' ')
```

```
//other good code
#include <iostream>
#include <vector>
using namespace std;

typedef long long ll;
typedef vector<vector<ll> > matrix;

matrix operator * (const matrix &a, const matrix &b) {
    ll size = a.size();
    matrix res(size, vector<ll>(size));
    for (ll i = 0; i < size; i++) {
        for (ll j = 0; j < size; j++) {
            for (ll k = 0; k < size; k++) {
                res[i][j] += a[i][k] * b[k][j];
            }
            res[i][j] %= 1000;
        }
    }
    return res;
}

matrix power(matrix a, ll n) {
    ll size = a.size();
    matrix res(size, vector<ll>(size));
    for (ll i = 0; i < size; i++) { // 단위 행렬
        res[i][i] = 1;
    }
    while (n > 0) {
        if (n % 2 == 1) {
            res = res * a;
        }
        n /= 2;
        a = a * a;
    }
```

```
    return res;

}

void PrintMatrix(const matrix& a) {
    ll size = a.size();
    for (ll i = 0; i < size; i++) {
        for (ll j = 0; j < size; j++) {
            cout << a[i][j] << " ";
        }
        cout << '\n';
    }
}
int main(void) {
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);
    cout.tie(NULL);
    ll n, b;
    cin >> n >> b;
    matrix a(n, vector<ll>(n));
    for (ll i = 0; i < n; i++) {
        for (ll j = 0; j < n; j++) {
            cin >> a[i][j];
        }
    }
    PrintMatrix(power(a,b));

    return 0;
}
```

백준 11444번 피보나치 수 6 https://www.acmicpc.net/problem/11444
처음 문제를 봤을 때 많이 당황했다. n의 범위가 10^18 이었기 때문이다. 무조건 O(logn)인 방법을 찾아야 풀 수 있다는 것을 알 수 있었고, dp를 사용한 O(n) 풀이는 사용하면 안된다는 것을 알 수 있었다. 수식을 이리저리 이동시키다보니,
F(2n+1) = (F(n+1))^2 + (F(n))^2
F(2n) = F(n) * (F(n) + 2 * F(n - 1))
로 바꿀 수 있다는 것을 알게되었다. 빠른 거듭제곱 형태로 나온다는 것이다! 여기 언저리까

지는 구했었는데, 이를 빠른 거듭제곱처럼 해결할 생각을 하지 못해서 고민을 하다 결국 힌트를 얻고나서 해결할 수 있었다...
밑의 방식은 행렬곱셈을 이용한 풀이다. 선형대수학은 정말 유용한 수학 과목이었구나...ㅎㅎ

https://velog.io/@ledcost/백준-11444-파이썬-피보나치-수-6-골드3-분할-정복 → 행렬곱셈을 이용한 피보나치 수 풀이.

```
#my code 44m/s
from sys import stdin
n = int(stdin.readline().rstrip())
f_set = {0: 0, 1: 1, 2: 1}
mod = 1000000007


def fast_pow(x: int):
    if x in f_set:
        return f_set[x]
    else:
        b = fast_pow(x // 2)
        if x % 2 == 1:
            a = fast_pow(x // 2 + 1)
            f_set[x] = (a ** 2 + b ** 2) % mod
            return f_set[x]
        else:
            a = fast_pow(x // 2 - 1)
            f_set[x] = b * (b + 2 * a) % mod
            return f_set[x]


print(fast_pow(n) % mod)
```

```
#other good code 36m/s
import sys

def mat_mult(a, b):
    answer = [[0, 0], [0, 0]]
    for i in range(2):
```

```
            for j in range(2):
                for k in range(2):
                    answer[i][j] += a[i][k] * b[k][j]
                answer[i][j] %= 1000000007
        return answer


    def solution(n):
        matrix = [[1, 1], [1, 0]]
        if n == 1:
            return matrix

        temp = solution(n//2)
        if n % 2 == 0:
            return mat_mult(temp, temp)
        else:
            return mat_mult(mat_mult(temp, temp), matrix)

    n = int(sys.stdin.readline())
    print(solution(n)[0][1])v
```

백준 6549번 히스토그램에서 가장 큰 직사각형

https://www.acmicpc.net/problem/6549

이 문제는 생의 첫 플레 문제이다.. 정말 호기롭게 도전했는데, 시간초과 및 런타임에러로 문제를 푸는데, 2일이 넘게 걸리게 되었다. 분할정복으로 해결하려했지만, 내 처음 코드는 최악의 경우 O(nh)가 나오게 된다. 그런데, 재귀를 n번 할 수도 없고 h는 엄청나게 큰 수기 때문에, 높이를 이용한 방식은 고칠 필요가 있었다. 정말 많이 고민을 하다가 인터넷의 힘을 통해 분할정복으로 어떻게 접근을 해야하는지 알 수 있었다. 답을 보고도 이해하는데 오래 걸렸던 것 같다.. 모두 지우고 다시 코드를 짜보면서 비록 지금의 코드가 빠르지도 않고 다소 더러워 보이지만, 그래도 내 나름의 최선의 코드였다. O(nlogn)이고, 나중에 분할정복 알고리즘을 다시 공부하게 되면 꼭 다시 공부해야 할 문제 유형인것 같다.

아래 코드는 stack을 이용한 방식이다. 정말 기발하고 속도도 빠른 방식이라 진짜 많이 놀랐던 것 같다. 갈길이 멀다. ㅎㅎ..

stack으로 해결한 코드 출처:

https://velog.io/@e_juhee/python-백준-6549-히스토그램에서-가장-큰-직사각형

```
#my code 2288m/s
from sys import stdin


def histogram(left: int, right: int):
    if right == left:
        return graph[left]
    if right - left == 1:
        return max(min(graph[left], graph[right]) * 2, graph[
    mid = (right + left) // 2
    mid_l, mid_r = mid, mid + 1
    h = min(graph[mid_l], graph[mid_r])
    maxi = max(histogram(left, mid), histogram(mid + 1, right
    while left < mid_l and mid_r < right:
        if (mid_l == left and mid_r < right) or (graph[mid_l
            mid_r += 1
            h = min(h, graph[mid_r])
        elif (mid_r == right and mid_l > left) or (graph[mid_
            mid_l -= 1
            h = min(h, graph[mid_l])
        maxi = max(maxi, h * (mid_r - mid_l + 1))
    while left == mid_l and mid_r < right:
        mid_r += 1
        h = min(h, graph[mid_r])
        maxi = max(maxi, h * (mid_r - mid_l + 1))
    while right == mid_r and left < mid_l:
        mid_l -= 1
        h = min(h, graph[mid_l])
        maxi = max(maxi, h * (mid_r - mid_l + 1))
    return maxi


while 1:
    graph = list(map(int, stdin.readline().split()))
    if graph[0] == 0:
        break
    print(histogram(1, graph[0]))
```

```
#other good code 408m/s
from sys import stdin


while True:
    graph = list(map(int, stdin.readline().split()))

    # 0이 입력되면 반복문을 종료합니다.
    if graph[0] == 0:
        break

    # 스택과 최대 직사각형 넓이를 저장할 변수를 초기화합니다.
    stack = []
    max_result = 0

    for i, height in enumerate(graph):
        if i == 0:  # 첫 번째 i는 막대기의 개수를 의미하므로 넘어갑니다
            continue

        # 스택의 가장 위쪽 막대기보다 현재 막대기의 높이가 작으면
        if stack and stack[-1][1] > height:
            while stack:  # 스택에서 빼내며 최대 직사각형 넓이를 계산
                stack_i, stack_height = stack.pop()
                width_start = 1
                if stack:
                    width_start = stack[-1][0]+1
                result = (i - width_start) * stack_height
                max_result = max(result, max_result) # 최대값 갱
                # 스택에 들어있는 막대 중에서 현재 막대의 길이보다 큰 것
                if not stack or stack[-1][1] <= height:
                    break

        # 스택이 비어 있거나 스택의 가장 위쪽 막대기보다 현재 막대기의 높
        if not stack or stack[-1][1] <= height:
            stack.append((i, height))  # 스택에 현재 막대기를 추가

    # 반복이 종료되고, 스택에 남은 막대기가 있다면 계산합니다.
    while stack:
```

```
        stack_i, stack_height = stack.pop()
        width_start = 1
        if stack:
            width_start = stack[-1][0]+1
        result = (graph[0]+1 - width_start) * stack_height
        max_result = max(result, max_result) # 최대값 갱신

    # 최대 직사각형 넓이를 출력합니다.
    print(max_result)
```

백준 1920번 수찾기 https://www.acmicpc.net/problem/1920
이분탐색으로 해당 문제를 해결했다. 그런데, 정답을 훑어보던중, set을 이용해 정말 빠르게 해결한 풀이가 있어서 가지고 왔다. set을 이용해서 nlogn(sort → nlogn, binary_search → logn)의 시간복잡도를 n(make_set → n, search → 1)으로 줄였다.

```
#my code 420m/s
from sys import stdin

n = int(stdin.readline().rstrip())
a_li = list(map(int, stdin.readline().split()))
a_li.sort()
m = int(stdin.readline().rstrip())
num_li = list(map(int, stdin.readline().split()))


def binary_tree(val: int):
    left = 0
    right = n - 1
    while left <= right:
        mid = (left + right) // 2
        if a_li[mid] == val:
            return 1
        elif a_li[mid] > val:
            right = mid - 1
        else:
            left = mid + 1
    return 0
```

```
for i in num_li:
    print(binary_tree(i))
```

```
#other good code 100m/s
import sys
input = sys.stdin.readline

def run():
    N = input()
    nums = set(input().split())
    M = input()
    target = input().split()

    ans = []
    for i in target:
        if i in nums:
            ans.append("1")
        else:
            ans.append("0")

    print ('\n'.join(ans))

run()
```

백준 10816번 숫자 카드 2 https://www.acmicpc.net/problem/10816
dictionary를 이용해서 풀어봤다. 이분탐색을 이용하지 않아도 정말 쉽게 풀 수 있는 것 같다. 역시 자료구조는 파이썬이 짱!

```
#my code 756m/s
from sys import stdin

n = int(stdin.readline().rstrip())
n_li = list(map(int, stdin.readline().split()))
n_dict = {}
```

```
for i in n_li:
    if i in n_dict:
        n_dict[i] += 1
    else:
        n_dict[i] = 1

m = int(stdin.readline().rstrip())
m_li = list(map(int, stdin.readline().split()))

ans = []
for i in m_li:
    if i in n_dict:
        ans.append(n_dict[i])
    else:
        ans.append(0)

print(*ans, sep=' ')
```