

2800 & 2841

1) 2800

```
import sys
from itertools import combinations

inp=list(map(str,sys.stdin.readline().rstrip()))
save=[] #인덱스 순서쌍 저장
stk=[] #" 저장할 공간

# 1) 인덱스 순서쌍을 저장하는 과정

for i in range(len(inp)) :
    if inp[i]=="(" : #"(" 를 발견하면 스택에 "("의 인덱스를 넣어놔다가
        stk.append(i)
    elif inp[i]==")" :
        save.append([stk.pop(),i])
        #" ) 만나면 스택에 들어가있는 최신 "("와 쌍이므로
        # 스택에 있는 인덱스와 짝 인덱스를 저장

res=[] #결과 담을 리스트
# print(save)
for i in range(1,len(save)+1): #나온 순서쌍 갯수만큼 돌면서
    # 조합의 경우를 구해줄거임
    #세쌍의 괄호 순서쌍이 있다면
    # 하나 뽑을 때, 두개 뽑을 때, 세개 뽑을 때,, 각 조합의 경우 구해주기
    aftercombination= list(combinations(save, i))
    # combination (리스트, 갯수)
    # 리스트에서 갯수만큼 조합으로 뽑바줘!
    #print(list(aftercombination))
    for j in aftercombination : # 각 조합의 경우를 돌면서, 조합 경우에 해당 인덱스를 발견한다면
        # print(j)
        inpcpy2 = inp.copy()
        #원본 인풋을 복사해놓은 인풋의 값에서 조합 인덱스 해당하는 괄호 없애주기
        #print(inpcpy2)
        for e,s in j : 6,10 / 3, 11
            #print(e,s)
            inpcpy2[e]="" #인덱스저장, 뺀 문자만 저장 bb => 메모리 효율적
            inpcpy2[s]=""
            res.append("".join(inpcpy2))

#print(res)
res=list(set(res))
#중복 제거해주기 위해 집합으로 바꿨다가
#정렬 위해 다시 리스트로 변경
res.sort()
#사전 순 정렬을 위해 진행 (안해주면 틀리더라)
for i in res:
    print(i)
```

print(save) 결과 (순서쌍
의 종류)

```
y
(1+(2*(3+4)))
[[6, 10], [3, 11], [0, 12]]
([6, 10],)
```

각 조합의 경우

이 조합의 인덱스들에 해
당하는 놈들 찾으면
시작 인덱스와 , 끝 인덱
스를 "" 로 처리해서 없애
주기

```
[[6, 10], [3, 11], [0, 12]]
([6, 10],)
([3, 11],)
([0, 12],)
([6, 10], [3, 11])
([6, 10], [0, 12])
([3, 11], [0, 12])
([6, 10], [3, 11], [0, 12])
(1+(2*(3+4)))
```

2) 2841

```
import sys
# 손가락의 가장 적게 움직이는 수를 구하는 프로그램
n,p = map(int, sys.stdin.readline().split())
lis=[]
lis2=[[ ] for _ in range(7)] #각 줄
cnt =0 #손가락을 움직 횟수

# 입력값 저장
for i in range(n) :
    lineno, pno = map(int, sys.stdin.readline().split())
    lis.append([lineno, pno])

# 이전 음이 나보다 낮은 음이면 안 움직여도 되고, 새 손가락 한번만 움직이니 cnt+=1
# 같은 음이면 cnt조차 안더해도 됨
# 높은 음이면 나랑 같거나 낮은 애 나올 때까지 손가락 움직이고(cnt 더해나가기),
# 나보다 작은 애, 같은 애 발견하거나 lis2에 암것도 없어도 없어도 마지막에 새 손가락 더해주기 cnt+=1
for l,p in lis :
    #print(l,p)
    if not lis2[l] : # lis2[l] 에 아무것도 없음 내가 처음이니깐 바로 넣어주면 된다
        lis2[l].append(p)
        cnt+=1

    else : #lis2[l] 에 뭔가 존재한다면!?
        if lis2[l][-1]<p: #lis2의 최신 아이가 지금 내 플랫폼보다
            lis2[l].append(p)
            cnt+=1

        elif lis2[l][-1] == p:
            continue

        else : #이전 음이 나보다 높은 음이면 움직여
            for j in range(len(lis2[l])-1, -1, -1) :
                if lis2[l][j] < p :# 내가 더 큰경우
                    lis2[l].append(p)
                    cnt+=1
                    break

            elif lis2[l][j]> p :
                #print(lis2[l])
                lis2[l].pop()
                cnt+=1

            else : #때때보니깐 나랑 똑같은 수 => 안움직여도됨
                break

    if not lis2[l] : #텅 비었으면~~
        lis2[l].append(p)
        cnt+=1

    #print("lis2 : " , lis2)
print(cnt)
```

1	2	3	4	5	6
플	5	3,5	4	5	6

```
lis2 : [[ ], [5], [ ], [ ], [ ], [ ], [ ]]
lis2 : [[ ], [5], [3], [ ], [ ], [ ], [ ]]
lis2 : [[ ], [5], [3, 5], [ ], [ ], [ ], [ ]]
lis2 : [[ ], [5], [3, 5, 7], [ ], [ ], [ ], [ ]]
lis2 : [[ ], [5], [3, 4], [ ], [ ], [ ], [ ]]
lis2 : [[ ], [3], [3, 4], [ ], [ ], [ ], [ ]]
9
```

3) top 틀린 풀이

```
import sys

n=int(sys.stdin.readline())
inp=list(sys.stdin.readline().rstrip().split())

withIndex=[]
for i in range(n) :
    withIndex.append([int(inp[i]), i+1]) #높이, 몇번째 탑인지 번호 저장
#print(withIndex)
res=[]
stk=[]
for i in range(n) :
    #print(stk)

    if stk :
        #만약 스택에 들어있는 높이 중 최댓값이 내 높이보다 작다?
        # 그럼 내가 보낸 레이저 아무도 못받아
        if max(stk)[0] < withIndex[i][0] :
            res.append(0)

        #스택 애들 중 나보다 큰 아이 중 최신 아이
        #최신 아이 찾아야 되니깐 맨 뒤부터 검사 (역순)
        else :
            for j in range(len(stk)-1,-1,-1) :
                if stk[j][0]>withIndex[i][0] :
                    res.append(stk[j][1])
                    break
            else : #맨 첫번째 아이
                res.append(0)
            stk.append(withIndex[i])
            #print("res" ,res)

for i in res:
    print(i, end=" ")
```

3,5,7

5,7 ← 4

[]

<https://ywtechit.tistory.com/204>

1. `stack` 이 존재하는 동안에 이전에 들어온 `stack[-1][1]` 의 값과 지금 들어올 `top[i]` 의 값을 비교한다.
2. 만약, 이전에 들어온 `stack[-1][1]` 의 값이 `top[i]` 보다 크면 레이저 신호를 수신할 수 있기 때문에 그때의 `stack[-1][0](index)` 를 저장한다.
3. 그렇지 않으면, 다음으로 존재하는 `stack` 값을 확인하기 위해 `pop()` 한다.
4. `stack` 에 아무것도 남지 않으면 `[i, top[i]]` 값을 넣는다.(이때, `i` 를 넣는 이유는 `index` 를 저장