

---

오늘은 백준 실버 3 문제


```
// C99
#include <stdio.h>

int main(void)
{
    int N; // 사람의 수
    int P[1000] = { 0, }; // 각 사람이 돈을 인출하는데 걸리는 시간 / 배열의 모든 요소를 0으로
    int temp1, temp2; // 값을 임시로 저장할 변수
    int sum = 0; // 각 사람이 돈을 인출하는데 필요한 시간의 합의 최솟값

    scanf("%d", &N);
    for (int i = 0; i < N; i++) // 입력한 값들을 비내림차순으로 정렬, 그림 2 참고
    {
        scanf("%d", &temp1);

        for (int j = 0; j < i; j++)
        {
            if (temp1 < P[j])
            {
                for (int k = j; k < i + 1; k++)
                {
                    temp2 = P[k];
                    P[k] = temp1;
                    temp1 = temp2;
                }
                break;
            }
        }
        if (P[i] == 0) // temp1이 P 배열에서 가장 큰 값일 때
            P[i] = temp1;
    }
    for (int i = 0; i < N; i++) // 그림 1 참고
        sum += P[i] * (N - i);
    printf("%d", sum);

    return 0;
}
```



# BAE<K>JOON>

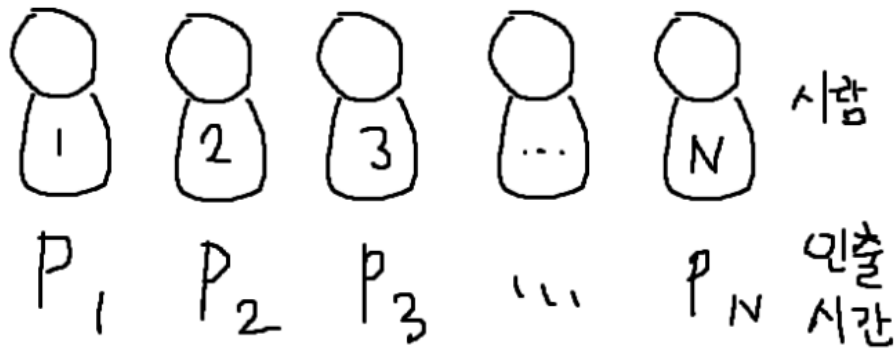
ONLINE JUDGE

## 11399번: ATM

11399번 제출 맞은 사람 숏코딩 재채점 결과 채점 현황 강의 ATM 분류 ...

[www.acmicpc.net](http://www.acmicpc.net)

문제를 풀 때 그림으로 정리하면서 푸는 습관을 들이려고 합니다  
무지성 프로그래밍은 손이 기억해도 머리가 기억하질 못합니다  
각설하고 설명을 시작해보겠습니다



비내림차순 정렬  
(같은 것이 있는 오름차순 정렬)

$$P'_1 \quad P'_2 \quad P'_3 \quad \dots \quad P'_N$$

각 사람이 돈을 인출하는데 필요한 시간의 합(의 최솟값)

$$\begin{aligned}
 &= \underbrace{(P'_1)}_{N\text{-개}} + \underbrace{P'_1 + P'_2}_{N-1\text{개}} + \underbrace{P'_1 + P'_2 + P'_3}_{N\text{번이 돈을 인출하는데 걸리는 시간}} + \dots \\
 &+ \underbrace{(P'_1 + P'_2 + P'_3 + \dots + P'_N)}_{\text{N번이 돈을 인출하는데 걸리는 시간}} \\
 &= P'_1 \times N + P'_2 \times (N-1) \dots
 \end{aligned}$$

그림 1

앞 사람이 돈을 뺄 때까지 기다려야 하기 때문에  
돈을 인출하는 데 걸리는 시간이 짧은 사람이 줄 앞에 있어야  
각 사람이 돈을 인출하는데 필요한 시간의 합은 작아집니다

인출 시간이 서로 같은 사람이 있을 수 있으므로  
비내림차순으로 인출 시간을 정렬하여 계산한 값이 최솟값이 되겠습니다  
쉽게 말해 돈 빨리 뺄 사람 줄 앞으로 보내서 계산하는 겁니다

참고로 P 배열 요소와 N의 최댓값은 각각 1000이기 때문에  
 sum의 최댓값은 P 배열의 모든 요소와 N이 1000일 때의 값인 500500000입니다  
 만약 최댓값이 int 범위를 초과한다면 sum을 다른 자료형으로 선언해야겠지요

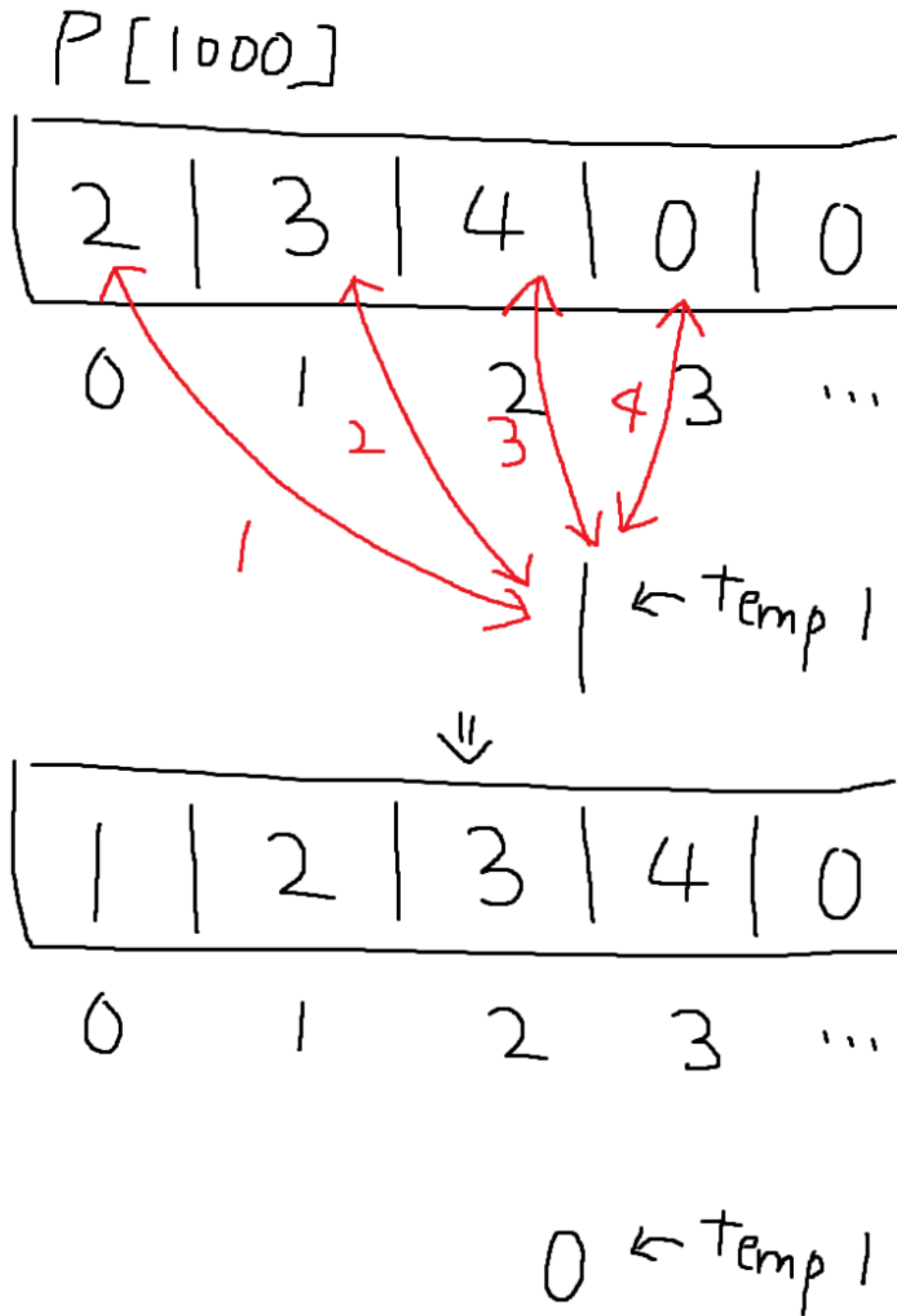


그림 2

temp1 값이 입력될 때마다 다음과 같은 동작이 진행됩니다  
 저는 값이 들어올 때마다 새롭게 정렬하는 방식으로  
 코드를 작성했습니다

P의 0 인덱스부터 시작하여 temp1보다 큰 값을 만나면  
 그 자리에 temp1을 넣고 나머지 값들은 뒤로 밀어냅니다  
 자신보다 큰 값이 없으면 맨 뒤에 temp1이 자리잡습니다

