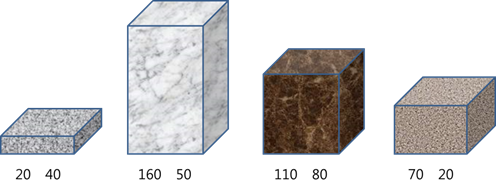
**(중) [기출A-0040] 석탑**

|  |  |
| --- | --- |
| 시간 제한 | **20**개의 Test Input 입력시 C/C++ **1**초 / Java **1.5**초 |
| 메모리 제한 | Stack : **1** Mbytes  /  Total : **256** Mbytes |
| 제출횟수 제한 | **10** 회 |
| 채점 | 답안을 제출하면 Test Input에 대한 결과를 판정해서 실시간으로 알려주며  그 의미는 다음과 같다.  **Pass : Test Input에 대하여 모두 정답 처리된 경우**  **Fail : Test Input에 대하여 일부 혹은 모두 오답 처리된 경우**  ※ Test Input : 검정 시스템에서 제출된 코드 실행 時 자동으로 입력되는 Input Data |
| 평가기준 | 정답 처리된 Test Input의 개수 |

다양한 높이의 직육면체 모양의 돌들을 쌓아 구조물을 만들려 한다. 돌들은 무게가 무거워 중장비를 이용하여 들어올려야 하며 무게가 다양하기 때문에 각 돌들을 들어올려 쌓을 때 소모되는 중장비의 연료량 또한 다양하다. 또한 소모되는 연료량은 **들어올리는 높이에 비례하여 증가**한다. 아래와 같이 4개의 돌들을 쌓아 구조물을 만드는 경우를 보도록 하자.



위 예에서 각 돌들의 아래에 존재하는 두 숫자는 각 돌들의 높이(cm)와 그 돌을 10cm 들어올려 쌓을 때 소모되는 연료량을 의미한다. 좌측 돌부터 쌓을 경우 첫 번째 돌은 들어올릴 필요가 없으므로 연료가 소모되지 않고 두 번째 돌의 경우 20cm 높이 위에 쌓아야 하므로 100(2 \* 50)의 연료가 소모된다. 세 번째 돌의 경우 180cm 높이에 쌓아야 하므로 1440(18 \* 80)의 연료가 소모되고 네 번째 돌의 경우 580(29 \* 20)의 연료가 소모되어 총 2120의 연료가 소모된다.

직육면체 돌들의 수 N과 각 돌들의 높이 및 10cm 들어올릴 때 소모되는 연료량이 주어질 때 N개의 돌들을 모두 쌓아 구조물을 만들 수 있는 **최소한의 연료 소모량의 합**을 출력하는 프로그램을 작성하시오.

**[입력]**  
최초 테스트케이스의 개수 T가 주어지며 다음 줄부터 T개의 테스트 케이스가 주어진다. 각 테스트 케이스는 여러 줄로 구성이 되며 첫 줄에는 돌들의 수 N(4 ≤ N ≤ 1000)이 주어진다. 다음 N줄에는 각 돌들의 높이와 10cm 높이 당 연료 소모량이 공백으로 구분되어 주어진다. 단, 높이는 10 이상 1000 이하의 10의 배수인 정수 값으로 주어지며 연료 소모량은 1 이상 100 이하의 정수 값으로 주어진다.

**[출력]**  
각각의 테스트 케이스에 대하여 #x(x는 테스트 케이스 번호를 의미)를 출력하고 공백을 하나 둔 다음 모든 돌을 쌓을 수 있는 **최소한의 연료 소모량의 합**을 출력한다.

**[입출력 예]**(입력)  
4  
4  
20 40  
160 50  
110 80  
70 20  
4  
800 13  
250 41  
1000 74  
630 79  
6  
290 99  
220 50  
190 29  
880 51  
230 99  
170 91  
8  
180 42  
650 41  
970 3  
550 100  
160 79  
740 10  
170 85  
980 31

(출력)  
#1 1390  
#2 10931  
#3 17342  
#4 21195

(sample\_input.txt에 대한 출력)  
#1 1390  
#2 10931  
#3 17342  
#4 21195  
#5 63271  
#6 264242  
#7 1098597  
#8 2376067  
#9 4918585  
#10 6597399

**(중상) [기출P-0067] 가장 큰 최단 경로**

|  |  |
| --- | --- |
| 시간 제한 | **20** 개의 Test Input 입력시 C/C++ **2** 초, Java **3** 초 |
| 메모리 제한 | Stack : **5** Mbytes  /  Total : **256** Mbytes |
| 코드길이 제한 | **128** Kbytes |
| 채점 | 답안을 제출하면 Test Input에 대한 결과를 판정해서 실시간으로 알려준다.  모든 케이스에 대해 정답일 경우 **Pass**, 그 외의 경우 **Fail** 이다. |

1번부터 N번까지 번호가 부여된 N개의 정점이 있고, 서로 다른 두 정점을 잇는 간선에는 가중치(길이)가 있는 무방향 그래프를 생각해 보자. 이 그래프의 연결된 임의의 두 정점 사이의 최단 경로는 쉽게 구할 수 있다. **그래프 전체에서 연결된 임의의 두 정점 사이의 최단 경로들 중 가장 큰 값을 그래프의“가장 큰 최단 경로”**라고 정의하자. 이 때, 임의의 두 정점 A, B 를 선택 시 A와 B가 연결되어 있지 않다면 최단 경로는 존재하지 않지만, **A와 B가 서로 다를 필요는 없다는 것에 유의한다.**

이제, 주어진 그래프에서 정점이 하나씩 제거된다고 하자. 물론 **정점이 제거될 때 그 정점에 연결된 간선들도 모두 제거**된다. 그렇다면 당연히 그래프의“가장 큰 최단 경로”도 변할 수 있다.

그래프의 정보와, 그 정점들이 제거되는 순서를 입력으로 받아“가장 큰 최단 경로”의 변화를 출력하는 프로그램을 작성하라.

**[입력]**   
맨 첫 줄에는 테스트 케이스의 개수 T가 주어진다. 각 케이스의 첫째 줄에는 정점의 개수 N이 주어진다. (2 ≤ N ≤ 500) 다음 줄에 정점의 번호를 의미하는 1, 2, 3,…, N의 값이 정점들이 지워지는 순서대로 주어진다. 다음 줄에 1번 정점과 2번, 3번, … N번 정점 간의 간선의 길이가 N-1개의 정수로 차례대로 주어진다. 그 다음 줄에 2번 정점과 3번, 4번, …, N번 정점 간의 간선의 길이가 N-2개의 정수로 차례대로 주어진다. 이렇게 반복되어 마지막에는 N-1번 정점과 N번 정점 간의 간선의 길이가 1개의 정수로 주어진다. 길이 값이 0으로 주어진 경우는 해당하는 간선이 존재하지 않음을 뜻한다. 간선의 길이 값은 0 이상, 1,000,000,000 이하이다.

**[출력]**   
각 테스트 케이스의 답을 순서대로 표준출력으로 출력하며, 각각의 테스트 케이스에 대하여 #x (x는 테스트 케이스 번호, 1부터 시작) 을 출력하고 공백을 하나 둔 다음, N개의 정수를 공백으로 구분하여 출력한다. 첫번째 수는 **최초 그래프에서 “가장 큰 최단 경로”**이어야 한다. 두번째 수는 **입력에 지정된 첫번째 정점까지 제거된 후“가장 큰 최단 경로”**이어야 한다. **세번째 수는 입력에 지정된 두번째 정점까지 제거된 후 “가장 큰 최단 경로”**이어야 한다. 이런 식으로, **마지막 N번째 수는 그래프에 정점이 단 하나 남았을 때의“가장 큰 최단 경로”**이어야 한다.

**[입출력 예]**   
(입력)   
2                                              ← 2 test cases in total   
3                                              ← 1st case   
2 3 1   
1 2   
4   
4                                              ← 2nd case   
3 2 1 4   
1 0 2   
1 0   
2

(출력)   
#1 3 2 0   
#2 3 3 2 0

(sample\_input.txt 의 출력)   
=> sample\_input의 정답은 지면 관계상 sample\_input.txt 하단에 함께 작성하여 공유드리니, 풀이에 참고하여 주시기 바랍니다.