Eko

Link submit: http://www.spoj.com/problems/EKO/

Solution:

C++	http://ideone.com/6FaSZh
Java	https://ideone.com/h2w5EF
Python	https://ideone.com/Xj5xyl

Python khi submit sẽ bị TLE do dữ liệu của trang SPOJ chưa sẵn sàng cho ngôn ngữ Python, mọi người có thể vào algote.com để submit bài này.

Tóm tắt đề:

Có N cái cây ($1 \le N \le 10^6$), cây thứ i có độ cao là h_i ($1 \le h_i \le 10^9$). Eko muốn chọn một độ cao H nào đó dành cho toàn bộ các cây. Eko sẽ cưa các cây sao cho độ dài chiều cao của các cây không được vượt quá H. Bạn cần tìm độ cao H lớn nhất là bao nhiêu, sao cho tổng độ dài các phần gỗ bị cưa của các cây không nhỏ hơn M.

Input:

Dòng đầu tiên chứa hai số nguyên N, M ($1 \le N \le 10^6$, $1 \le M \le 2 * 10^9$) lần lượt là số lượng cây và số M được định nghĩa ở trên.

Dòng thứ hai gồm N số nguyên dương $h_1, h_2, ..., h_n$ ($1 \le h_i \le 10^9$) lần lượt là độ cao của N cây.

Tất cả những số trên cách nhau bởi dấu khoảng cách.

Output:

In ra một số nguyên là độ cao H lớn nhất mà bạn có thể tìm được để thỏa mãn điều kiện trên.

Ví dụ:

4 7 20 15 10 17	15

Giải thích ví dụ:

Bạn chọn H = 15 vì :

- Chặt cây đầu tiên có độ dài là 20 thì lượng gỗ bị chặt là 5
- Chặt cây thứ hai có độ dài là 15 thì lượng gỗ bị chặt là 0

- Chặt cây thứ ba, vì cây thứ ba độ dài đã nhỏ hơn 15 rồi nên không cần chặt nên lượng gỗ chặt là 0
- Chặt cây thứ tư có độ dài là 17 thì lượng gỗ bị chặt là 2.

Tổng lượng gỗ bị chặt: 5 + 0 + 0 + 2 = 7 ≥ 7 thỏa mãn.

Ta không thể nào tìm được một độ cao H tốt hơn 15 nữa.

Hướng dẫn giải:

Ta dựa vào một nhận xét như sau: Khi H càng tăng, rõ ràng tổng lượng gỗ bị chặt sẽ càng giảm và ngược lại. Do đó, nếu giả sử như ta có thể quy định mid là độ dài của các khúc gỗ mà sau khi chặt không được phép vượt quá mid. Nếu như tổng lượng gỗ bị chặt mà ≥ M, điều này có nghĩa rằng đáp án của ta có thể là mid, hoặc đáp án của ta sẽ nằm trong đoạn từ mid + 1 trở về sau. Còn nếu tổng lượng gỗ bị chặt mà < M thì rõ ràng ta không thể nào nhận mid làm đáp án được mà phải tìm từ mid – 1 trở về trước. Từ ý tưởng này, ta hình thành nên thuật toán chặt nhị phân như sau:

- Gọi getSum(x) là tổng lượng gỗ bị chặt khi ta quy định độ dài của các cây sau khi chặt không được vượt quá x.
- Ta đặt l = 0, r = 10^9 + 10, đại diện cho trường hợp xấu nhất là không thể chặt được và trường hợp tối đa có thể chặt được.
- Trong lúc l ≤ r, ta đặt mid = (l + r) / 2, nếu getSum(mid) >= M thì ta tiến hành cập nhật res = mid, đồng thời tìm kết quả tiếp trên đoạn [mid + 1, r], tức gán l = mid + 1. Ngược lại, nếu getSum(mid) < M thì ta tìm kết quả trên đoạn [l...mid – 1], tức gán r = mid – 1.

Độ phức tạp: **O(Nlog(max(hi))** với N là số lượng cây và max(h_i) là chiều cao lớn nhất của cây.