АНАЛИЗ НА РЕШЕНИЕТО НА ЗАДАЧА КОНСТРУКТОР

В основата на решението на задачата лежи следното твърдение:

Нека са дадени N отсечки с дължини $d_1, d_2,....,d_N$. Необходимо и достатъчно условие от тях да може да се състави изпъкнал N-ъгълник е дължината на най-дългата от тях да е по-малка от сумата от дължините на останалите или, което е същото,

$$2.max\{ d_1, d_2,....,d_N\} < d_1 + d_2 + ... + d_N.$$

Това твърдение е обобщение на условието от три отсечки да може да се състави триъгълник. Доказателството на твърдението не е тривиално, но авторите на задачата считат, че то е достатъчно интуитивно ясно и може да се използва за решаване на задачата без да се доказва.

И така N е броя на отсечките, а нека с M означим най-голямата дължина на отсечка, която може да се срещне в задачата (в конкретния случай $M=10^9-1$).

Най-простият начин да се реши задачата е да се направи изчерпване по всички подмножества от отсечки и, използвайки горното твърдение, за всяко да се проверява дали образува изпъкнал многоъгълник. Сложността на такова решение е $O(2^N.N)$ и то ще получи 40 точки.

Решение на задачата може да се получи така:
Сортираме масива с дължините на отсечките. Нека се е получила следната последователност от дължини $d_1 \leq d_2 \leq \ldots \leq d_N$ и $S_N = d_1 + d_2 + \ldots + d_N$. Тогава можем да разсъждаваме така: ако $2.d_N < S_N$, т.е. изпълнено е необходимото и достатъчно условие всичките N отсечки да образуват изпъкнал многоъгълник, то задачата е решена, иначе отсечката с дължина d_N не може да участва в изпъкнал многоъгълник, образуван от които и да са от дадените отсечки. Премахваме я и към останалите $N-1$ отсечки с дължини $d_1 \leq d_2 \leq \ldots \leq d_{N-1}$ и сума от дължините $S_{N-1}=S_N-d_N$ прилагаме абсолютно същите разсъждения. Това продължава докато не стигнем до някое k, за което $2.d_k < S_k$ (тогава k е решението на задачата) или не свършат всички отчечки – тогава не може да се построи никакъв изпъкнал многоъгълник). Това е линеен алгоритъм и като сложим сложност на сортирането $O(NlogN)$, то получаваме обща сложност $O(NlogN)$.

Трябва да се внимава при първоначалното формиране на сумата след сортирането, да не се получи число, което ще надвиши максималното число за тип long long. Тъй като дължините на отсечките са по-малки от 10^9, то ако пресмятайки сумата $d_1 + d_2 + \ldots + d_N$ се достигне до препълване на суматора, който е от тип long long, то тогава ще е ясно, че $2.d_N < S_N$ и че всички отсечки могат да образуват изпъкнал многоъгълник.

Да разгледаме още един алгоритъм за решаване на задачата. Той се основава на следното твърдение:

Преди описаният по-горе алгоритъм да завърши своята работа, той ще отхвърли най-много $[logM]+2$ отсечки.

Доказателство: Нека алгоритъмът е отхвърлил l отсечки $d_N, d_{N-1}, \ldots, d_{N-l+1}$, $l \leq N$. Ако $l<N$, то, тъй като $d_1 \geq 1, d_2 \geq 1, \ldots, d_{N-l} \geq 1$ и алгоритъмът е отхвърлил тези числа, то можем да напишем:

$d_{N-l+1} \geq d_1 + d_2 + \ldots + d_{N-l} \geq N-l$,

$d_{N-l+2} \geq d_1 + d_2 + \ldots + d_{N-l} + d_{N-l+1} \geq (N-l)+(N-l) = 2(N-l)$,

$d_{N-l+3} \geq d_1 + \ldots + d_{N-l} + d_{N-l+1} + d_{N-l+2} \geq (N-l)+(N-l)+2(N-l) = 2^2(N-l)$

....

$d_N \geq (1+1+2+2^2+....+2^{l-2})(N-l)=2^{l-1}(N-l)$

Тъй като $M \geq d_N \geq 2^{l-1}(N-l) \geq 2^{l-1}$, то $l \leq logM+1$.

Ако $l=N$, то имаме $d_1 \geq 1$, $d_2 \geq 1$ и по-нататък

$d_3 \geq d_1+d_2 \geq 2$,
$d_4 \geq d_1+d_2+d_3 \geq 1+1+2=2^2$,
$d_5 \geq d_1+d_2+d_3+d_4 \geq 1+1+2+2^2=2^3$
.....

$d_N \geq 1+1+2+2^2+....2^{N-3}=2^{N-2}=2^{l-2}$

Тъй като $M \geq d_N \geq 2^{l-2}$, то $l \leq logM+2$.

Твърдението е доказано.

Тогава можем да построим алгоритъм, при който дължините на отсечките не се сортират, а на всяка стъпка се търси най-дългата отсечка. Нека тя е с индекс *imax*. Ако $2.d_{imax}<S_N$ то сме намерили решението, ако не махаме тази отсечка (и от сумата) и отново търсим най-дългата. Тъй като такива стъпки може да има най-много *O(logM)* и, при обикновено, линейно търсене на максимален елемент, неговата сложност е *O(N)*, то общата сложност на алгоритъма е *O(NlogM)*.

Задачата може да се реши и за почти линейно време. Въпреки, че тестовете не го изискват, алгоритъмът е поучителен.

Първата стъпка в такъв алгоритъм е да се подредят елементите в масива така, че на последните $m=[logM]+3$ места да застанат, макар и в произволен ред, ония елементи, които биха застанали на тези места и при пълно сортиране. Това може да се постигне за линейно време *O(N)* чрез някой от алгоритмите за намиране на *m*-тия най-малък елемент в N елементен масив. След това да сортираме само тези *m* елемента за време *mlogm* и да започнем да изпълняваме алгоритъма със сортирания масив. Тъй като, според доказаното твърдение, до края на работата на алгоритъма могат да бъдат отхвърлени максимум *logM+2* отсечки, то алгоритъмът въобще няма да стигне до несортираната част от масива. Сложността на алгоритъма е *O(N+mlogm)* или *O(N+logMloglogM)*, което е почти линейно по *N*.