Тривиално можем да считаме, че $1\leq lq\leq rq\leq x\_{lq}+ x\_{lq+1}+ …+ x\_{rq}$

**Решение за 5 точки**

Симулация с $bitset$ на всяка заявка.

Сложност: $O(\frac{qnS}{32})$

**Решение за 10 точки**

С $bitset$ можем да намерим възможните суми за всяка двойка $(l, r)$ и да отговорим на заявките с равни $(l, r)$ наведнъж.

Сложност: $O(\frac{\left(n^{2}+q\right)\*S}{32})$, като може да се модифицира и да хване 25 точки.

**Решение за 15 точки**

Нека поддържаме $r$ и $dp$, като$ dp[s]$ означава “в $x\_{dp[s]}, … , x\_{r}$ има подмножество със сума $s$, но в $x\_{dp\left[s\right]+1}, … , x\_{r}$ няма подмножество със сума $s$”. Движейки $r-1\rightarrow r$ е достатъчно да актуализираме $dp\\_new\left[s\right]=max⁡(dp\\_old\left[s\right], dp\\_old[s-x\_{r}])$ и $dp\\_new\left[x\_{r}\right]=r$

Заявката става “намери броя числа $lq\leq j\leq rq$ с $dp\left[j\right]\geq l$”

Сложност: $O\left((n+q\right)\*S)$

*Можем да забележим, че е достатъчно да актуализираме само до* $x\_{1}+ … + x\_{r}$*, като това може по префиксите или по суфиксите (достатъчно е да обърнем масива и да намерим еквивалентните заявки), като сумата им е* $\left(n+1\right)\*S$ *и можем да изберем начина, по който тя е по-малка, като така ще има* $\leq \frac{(n+1)S}{2}$ *актуализации общо.*

**Решение за 50 точки**

Ако сортираме $dp[s]$ и заявките по намаляващо $l$, ще имаме $2$ типа събития - “направи стойността в позиция $x$ единица” и “намери сумата на числата в позициите от $lq$ до $rq$”. За тях можем да поддържаме дърво на Фенуик.

Сложност: $O((\frac{nS}{2}+q)\*log\_{2}(S))$

**Решение за 80 точки**

Понеже $update$ събитията са много повече от $query$ събитията, можем да поддържаме първите за $O(1)$, а вторите за $O(\sqrt{S})$.

Сложност: $O(\frac{nS}{2}+q\*\sqrt{S})$

**Решение за 100 точки**

Понеже поддържаме само стойности $0$ и $1$ в позициите, отново можем да използваме $bitset$ и да намалим времето за $query$ на $\sqrt{\frac{S}{32}}$.

Сложност: $O(\frac{nS}{2}+q\*\sqrt{\frac{S}{32}})$

.

 Автор: Мартин Копчев