Введение

Описание алгоритма

задача сводиться к тому состоянию, когда нам нужно «эффективно разложить по пустым палатам» (в процессе считывания мы дозополняем те палаты, в которых уже кто-то есть, ибо — особо и вариантов нет (перемещать нельзя же. Да и будто бы если бы можно было, то мы бы все равно пришли бы к той же задаче (достали бы, палаты стали пустыми, ну и теперь по пустым фасовать))

может ли получиться так, что эффективнее было бы Не до заполнить таким образом? — нет (ибо : … )

в итоге мы имеем: разбросанные пустые палаты И оставшихся больных, которых нам нужно эффективно их разложить.

Идея такая — делаем массив размером {сумма всех свободных мест(далее sum) + 1(далее поймёте почему)}

Что этот массив отображает? — index элемента массива И значения элемента по этому index описывает «можно ли число index представить в виде суммы каких либо из палат?». К примеру: у нас есть пустые палаты 2, 3, 5. мы их можем по разному сложить : 2, 3, 5, 2 + 3, 2 + 5, 3 + 5, 2+3+5. если сложить 3+ 5 и 5+3 — то это тоже самое И это всё учитывается в один случай. т. е. , мы можем получить числа 0 (0 , 1 … sum — т. е. нам нужен массив размером sum + 1), 2,3,5, 2+3=5(уже было), 2+5=7, 3+5 = 8, 2+3+5 = 10

0 2 3 5 7 8 10 — такие элементы мы будем обозначать единичками

т. е., наш заполненный массив выглядит так  
0 1 2 3 4 5 6 7 8 9 10

1 0 1 1 0 1 0 1 1 0 1

далее мы берём наше А , пусть оно будет равно 5 , а Б пусть равно 5

обращаемся к массиву array по индексу A = 5

получаем, что — да, есть. т. е. — есть такая комбинация комнат, в которые можно поместить полностью больных А

далее — у нас остаётся 10 — 5 = 5 свободных мест

и получается так, что — у нас точно остались комнаты, по которым мы спокойно сможем разместить наших больных В

итого у нас все заполнено — оптимально!

Вопрос: Почему точно остались свободные палаты И мы точно сможем по ним(оставшимся местам) раскидать наших больных Б?

Причина вот в чём: вспомним, что такое array[index] = 1 — это значит, что index может быть представлен в виде суммы каких то из комнат (0 — нет такой комбинации-суммы комнат, чтобы получить такое число). т. е. a1 + a2 + … +an = y, где мы установили, что (без ограничения общности — иначе просто переномировать) A = a1 + … + aj, т. е. — aj+1 + … + an = y — A — столько мест осталось на В. И мы тут видим aj+1 + … + an — комбинацию-сумму.

Рассмотрим случай, когда «в нашем array нет места для А».

Что это вообще значит? — нет такой комбинации комнат, сумма вместимостей которых даст нам А == нет такой комбинации комнат, которыми можно полностью забить А-ами. И у нас есть только 2 выхода — либо некоторые А «гуляют», либо — ложаться все, Просто у нас теперь одна из палат не до конца забита — что может сыграть с нами «злую шутку»

Как это выглядит на примере? Пусть А = 6

0 1 2 3 4 5 6 7 8 9 10

1 0 1 1 0 1 0 1 1 0 1

как мы видим, array[6] = 0 — нет такой комбинации комнат, сумма вместимостей которых даст нам 6

Но — ближайшими вариантами является 5 и 7 (в данном случае они близко, а так — могут хоть сколько угодно дальше стоять)

и теперь мы смотрим два эти варианта. Пусть В = 6. если мы выберем 5, то на В-шки останется 10 — 5 = 5 мест. Итого у нас заполнено по максимуму. Если мы возьмём 7, то мало того, что мы не всё место занимаем (только 6 из 7), так ещё и на В осталось 10-7= 3. итого 6+3=9.

Заметим, что — если выбрать 5(нижнюю грань, когда А-ки гуляют), то мы уже сразу говорим — всех не заселим.

Далее поговорим о реализации И как восстановить путь

Реализация

это ДП задача И нам надо вернуть путь. Вернуть путь — последовательность номеров палат, в которые помещены А. С этим учётом мы и делаем наш массив

если мы заговорили о массиве, то — пусть массив «хранит index», хранит 1/0 , Какую комнату последнюю прибавили И — предыдущее значение, Из которого мы пришли в это значение. Это значение = Предыдущее значение + rooms[номер последней конматы (ну которой прибавили)].

Круто, а как это вообще заполнять? — берём наши комнаты 2 5 3 (я специально перетусовал) И нашу эту таблицу 0 … 10

базовый наш массив выглядит так:

0 1 2 3 4 5 6 7 8 9 10 -- index

1 0 0 0 0 0 0 0 0 0 0 -- могут достичь или нет

0 0 0 0 0 0 0 0 0 0 0 -- какую комнату занёс

-1 0 0 0 0 0 0 0 0 0 0 — предыдущее значение

берём 2 — это комната под номером 1. проходимся по всему нашему дп-массиву, И — там где 1 и номер последней добавленной комнаты не равной текущей, вот к тому числу добавляем 2 и по полученному индексу ставим 1 (т. е. — мы можем достичь это число)

ну вот к примеру после таких махинаций с 2

0 1 2 3 4 5 6 7 8 9 10

1 0 1 0 0 0 0 0 0 0 0

0 0 1 0 0 0 0 0 0 0 0

-1 0 0 0 0 0 0 0 0 0 0

потом 5

0 1 2 3 4 5 6 7 8 9 10

1 0 1 0 0 1 0 1 0 0 0

0 0 1 0 0 2 0 2 0 0 0

-1 0 0 0 0 0 0 2 0 0 0

потом 3

0 1 2 3 4 5 6 7 8 9 10

1 0 1 1 0 1/1 0 1 1 0 1

0 0 1 3 0 2/3 0 2 3 0 3

-1 0 0 0 0 0/2 0 2 5 0 7

что тут интересного? — а что делать, если на одно и тоже значение в таблице у нас два варината событий? — ничего не менять (от перестановки слогаемых сумма не меняется)

Как восстановить путь?

Ну пусть у нас А = 6, В = 5. Я понимаю, что у нас не все сядут, но мы все равно покажем путь А. Мы с вами определили, что лучше между 5 и 7 будет 5

0 1 2 3 4 5 6 7 8 9 10

1 0 1 1 0 1/1 0 1 1 0 1

0 0 1 3 0 2/3 0 2 3 0 3

-1 0 0 0 0 0/2 0 2 5 0 7

мы с вами договрились ничего не менять, поэтому 1 2 0. первые данные нам уже ничего не дают ( на этом этапе). Потом записываем куда-то номер последней добавленной комнаты (у нас 2). потом уже переходим по index на элемент по «принимавшее значение До». И так далее, пока не прийдём в 0

итого у нас комната под номер 2

(если бы мы взяли 1 3 2, то храним «3», потом переходим по двойке. Там 1 1 0. записываем 1 : «3 1». и потом оттуда в 0 И всё на этом. Итого 1 3)

Хочется ещё обсудить такие моменты: если была комната 50 50 0 (заведомо заполненная) И она заполнена А — её не в счёт. 0 0 0 — такая же история. Так же ситуация, когда A = 0 — её вообще забыть надо

Проблемы реализации и их решение

Массим мы сделали круто — но как его быстро обходить? — вот и я не придумал и воспользовался std::set.