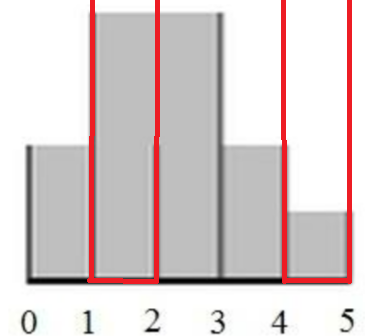
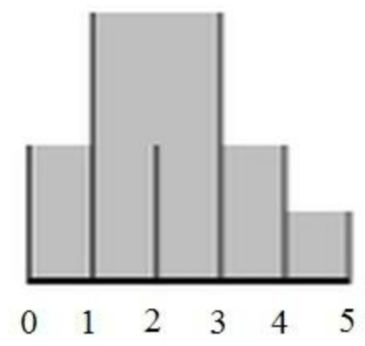
Важно уточнение да се отбележи за тази задача е, че ние реално не искаме да минимизираме сумата на сантиметрите, с които да повдигаме преградите, а само да минимизираме броя на преградите, които биват повдигани като цяло. Това обаче после ще забележим, че няма значение, понеже задачата има доста добри свойства. Алгоритъмът, който е реализиран във файла rain.cpp първо маха излишните прегради и после смята колко ще излезе водата, но ние ще разсъждаваме малко по друг начин и ще започнем първо с това, че искаме да сметнем колко най-много вода можем да поберем.

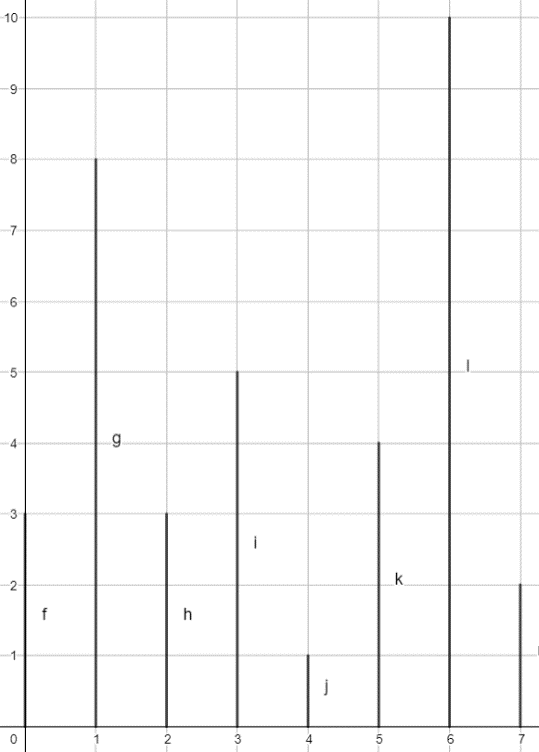
Доста близко до ума е, че ако увеличим всички прегради на макс ще получим максимално количество вода. Затова ще разгледаме ситуация, при която сме използвали всички свои увеличения. Ще разсъждаваме за всяко едно секторче, тоест за всяка лента между две стени. Тоест за нещо такова:

Тук са показани два сектора – между 1 и 2 и между 4 и 5.

Очевидно ако сумираме колко е висока водата във всеки един сектор ще намерим колко вода има общо. Тъй като водата се налива до преливане, тоест може някоя стена да бъде изцяло потопена (както например номер 2).



То е сравнително близо до акъла да се сетим, че всъщност няма значение кои са съседните стени, а всъщност какви са стените наляво и надясно. Нека го илюстрираме с малко по-конкретен пример.

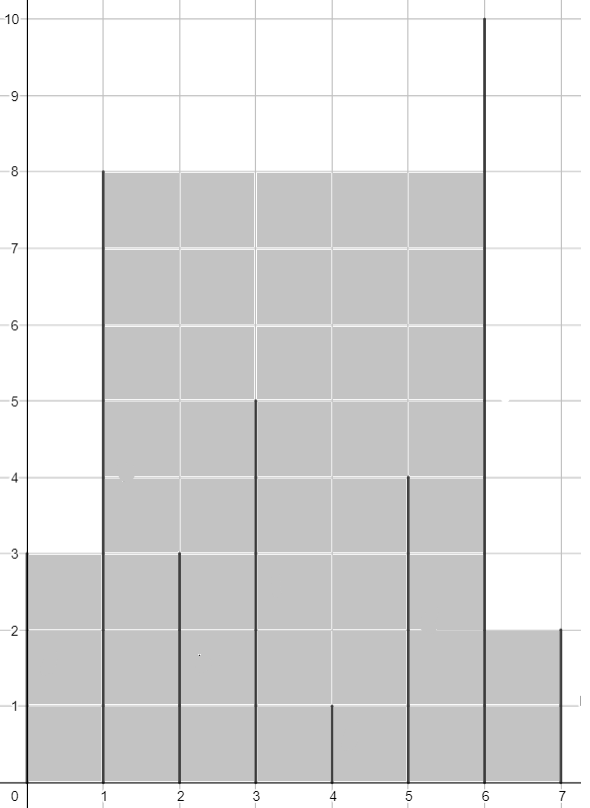
Колко ще бъде височината на стълбчето между 3 и 4? – 8

Колко ще бъде височината на стълбчето между 5 и 6? – 8

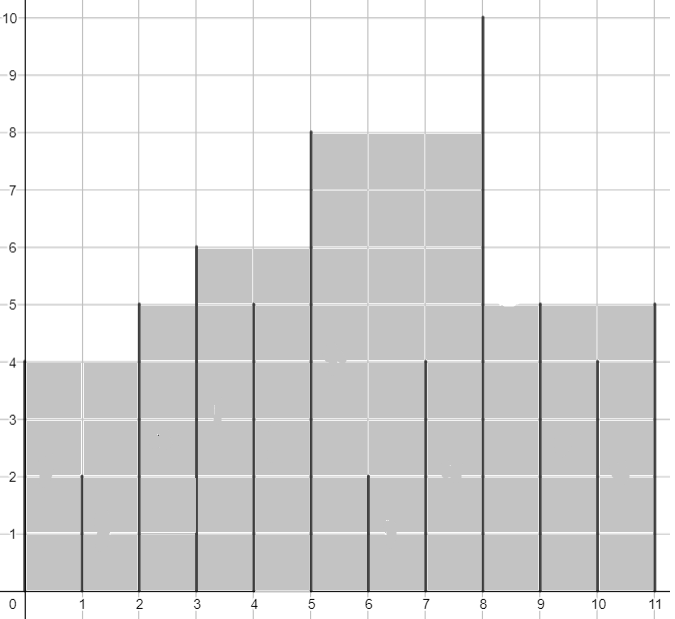
Колко ще бъде височината на стълбчето между 0 и 1? –3

Колко ще бъде височината на стълбчето между 4 и 5? –8

С малко пробване човек се убеждава, че ако наляво има много висока стена и надясно има много висока стена, то водата също ще бъде много висока, дори и съседните стени да бъдат сравнително ниски. Също се забелязва, че и ако на едната страна няма висока стена, а на другата има, то стълбът няма да е много висок. Формално това е точно зависимостта min{максимална лява стена, максимална дясна стена}. Ако се замисли човек това има доста повече логика, отколкото звучи, понеже няма как да имаме повече вода от това, тъй като тази вода би се разляла от единия край поне, защото няма да има какво да я държи. Също точно толкова вода може и да се побере, понеже имаме достатъчно високи стени. Ето и картинка на този пример, но запълнен с вода.

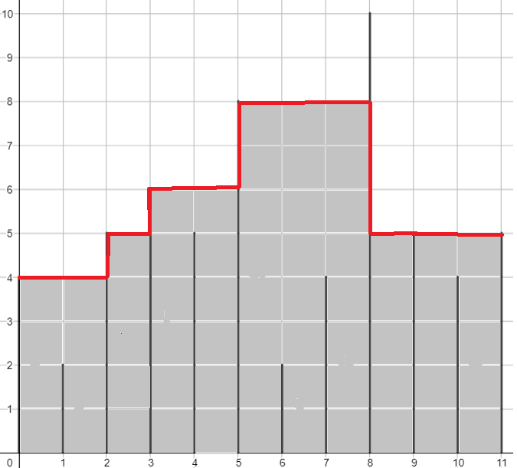


Ето и още един пример, за да се убедим в тази зависимост.

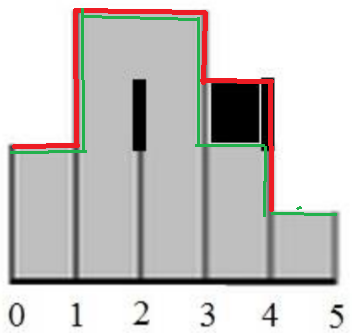


Можем да пресметнем точното количество за O(N) като използваме префиксни и суфиксни масиви, за да пазим максимумите.

Относно другата част на задачата, трябва да направим следното наблюдение. В тази задача „харченето“ на ресурс или подобрява нещата или не ги променя. Освен това за да видим колко най-малко ресурс ще използваме, ще трябва просто да видим добавки към стени са излишни. Тук идеята е, че няма как да правим нещо като стратегия за това къде да вдигаме нивото и къде да го пускаме, понеже ние искаме да постигнем максималното ниво за минимални ресурси. Тоест ние просто искаме да си запазим формата на водата. Под форма на водата се разбира контурът по „външните“ краища на стълбовете. На картинката това е червената начупена линия.



Та стратегията ще ни бъде просто да махнем излишните повишения, които не допринасят за формата на водата. Интуицията за това трябва да ни е, че колкото по-високи стени ползваме, толкова по добра форма ще имаме. И всяка форма, която можем да получим в някакъв смисъл се съдържа в максималната.

 Тук зелената форма се съдържа в червената.

По тази причина ще направим следното решение – ще ходим по всеки стълб и ще гледаме дали той допринася по какъвто и да е начин за формата на водата. Ще разгледаме три ситуации – потопена стена, стърчаща стена и стена, която е „между“ съседните максимуми.

1 сл. – потопена стена.

Това е случаят със стена номер 2 от примера на задачата. Очевидно тя не допринася с нищо, затова ще й махнем цялото увеличение.

2 сл. – стърчаща стена

Тук трябва да разделим на два подслучая – когато левия и десния максимум са равни и когато не са.

2.1 сл. – равни максимуми

Тук сме в ситуация подобна на тази. Разбира се стените може да не са съседни, ние гледаме максимумите отляво и отдясно, понеже както видяхме от миналата част на задачата само те ни интересуват.

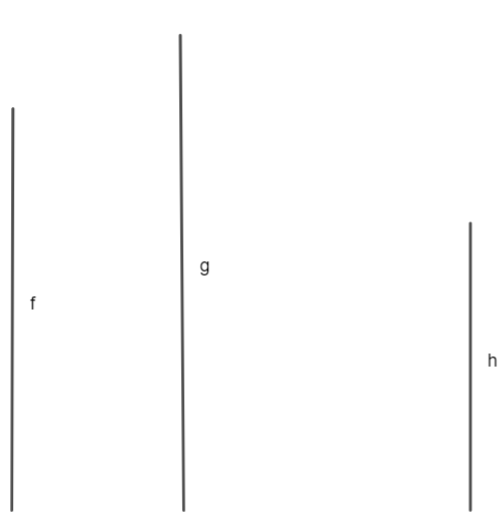
Картина, която съдържа текст, гардероб

Описанието е генерирано автоматично

Тук забелязваме, че какъвто и да е **g**, нивото на водата няма да се промени, затова му махаме цялото увеличение.

2.2 сл. – различни максимуми.

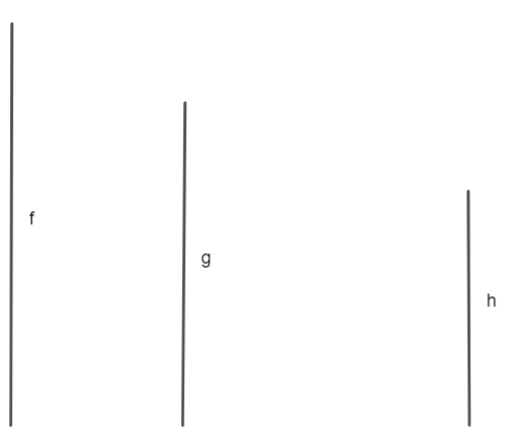
Тук сме в ситуация, подобна на тази.



Тук ако стената **g** е твърдениска, ще загубим вода, но все пак тя е излишно висока. Забелязваме, че тя няма смисъл да бъде по-висока от по-високото от двата си „съседа“ (по конкретно в нашия случай това е стената **f**). В този случай, използваме толкова увеличение, колкото ни трябва, за да достигнем по високия максимум. Разбира се, ако конкретната стена си е оригинално много висока, просто нулираме повдигането (затова е онази странна проверка в кода, ако lift[i] стане <0).

3 сл. – стена между двата максимума.

Тук сме в ситуация подобна на тази:



Както се вижда, тук няма да намаляваме **g**, понеже ще загубим вода. Затова в този случай не правим нищо.

За получим отговорът, който ни интересува, накрая просто броим кои стени биват увеличени като цяло.