**Анализ**

**Подзадача № 1**

В тази подзадача са тестовите примери. Тя е за обратна връзка от системата.

**Подзадача № 2**

Трябва да се започне от някъде, а именно от проверка за това дали един скобов низ е правилен. Един вариант е да се направи пълно изчерпване, следващо дефиницията от условието. Tо няма да е много приятно за писане от един шестокласник, а и ще е тромаво за изпълнение от един компютър (в най-добрия случай би работило достатъчно бързо за ). Другият вариант е да се използва стандартната проверка за правилен скобов низ от един и същи вид скоби. Нека в един скобов низ заменим всяка отваряща скоба с , а всяка затваряща скоба с 1. Трябва да важат следните две условия, за да е правилен:

* Сборът на всички числа е . (броят на отварящите скоби е равен на броя на затварящите)
* Минималната префиксна сума е точно равна на . (магия)

Тогава може лесно да се напише решение, обхождащо всеки един подниз и проверяващо дали е правилен.

Постигната сложност:

Имплементация: brackets\_35p.cpp

**Подзадача № 3**

Решението на третата подзадача е дребна оптимизация на горното решение. Нека проверим дали е правилен всеки подниз с начало . Може да се забележи, че поднизът ще има същите префикси като подниза , изключвайки префикса, завършващ на -вия елемент. Така ще трябва само да проверим за -вия префикс, защото предишните ще са проверени.

Постигната сложност:

Имплементация: brackets\_55p.cpp

**Подзадача № 4**

Нека в оригиналния низ заменим скобите с числа. Нека е префиксната сума, завършваща на -тия елемент, като . Тогава задачата бързо се свежда до това да преброим двойките (), за които са в сила условията:

* Няма , за което и

Ако горното условие важи за някоя двойка , то поднизът е образува правилен скобов израз. Защо? Нека, като сме направили описаната проверка за някой подниз, да сме получили префиксни суми , където е префиксната сума до -тия символ на подниза. Тогава може да се забележи, че , , …, . За да бъде поднизът правилен скобов израз, то трябва и всички да са . Горните две условия проверяват това.

Нека за момент игнорираме второто условие. Как да преброим двойките за които ? Тук има множество подходи, но този, който ще ни бъде най-полезен, е да съобразим, че числата са достатъчно малки, за да поддържаме counting. Въпросът е как да модифицираме counting-а, така че да реши нашата задача.

Нека в един масив cnt[value] поддържаме броя леви краища на поднизове, за които и няма по-малък префикс след тях. Ние ще обхождаме възможните десни краища отляво-надясно, като ще променяме cnt спрямо -тия префикс. Тогава има два случая:

* Текущата скоба е . Няма как скобов подниз, завършващ на , да е правилен. Тогава текущият префикс става възможен ляв край за следващите десни краища, заради това увеличаваме с . Тъй като префиксната сума се увеличава, то няма леви краища, за които е първата по-малка следваща префиксна сума.
* Текущата скоба e . Тогава за всеки ляв край с префиксна сума, равна на се появява по-малък следващ префикс. Заради това трябва да занулим . Всички леви краища с префиксна сума със сигурност са по-големи от , заради това те вече са занулени. Към отговора добавяме .

И остана да отпечатаме верния отговор ☺.

Постигната сложност:

Имплементация: brackets\_100p.cpp

*Автор: Борис Михов*