```
from sys import stdin, stdout

rta = []

tests = int(stdin.readline())

for _ in range(tests):
    tam_arr = int(stdin.readline())

    num_arr = [int(elem) for elem in stdin.readline().split()]

    maximo = num_arr[0]
    suma_actual = num_arr[0]

    for I in range(1,len(num_arr)):
        # busco la suma maxima de subsecuencia, algoritmo de kadane
        suma_actual = max(suma_actual + num_arr[I], num_arr[I])

        maximo = max(suma_actual, maximo)

    acumulado = 0
    aparecidos = {}
    aparecidos[0] = 1
    ocurrencias = 0

    for num in num_arr:
        acumulado += num # acumulado
        diferencia = acumulado - maximo

        if diferencia in aparecidos:
            ocurrencias += aparecidos[diferencia]

        if acumulado in aparecidos:
            aparecidos[acumulado] += 1
        else:
            aparecidos[acumulado] = 1

    rta.append("{} {}".format(maximo, ocurrencias))

stdout.write("\n".join(rta) + "\n")
```

MAXSUMSQ

Maximum Sum Sequences

Resuelto por
Luciano Agustin Macias

Enunciado

Dado un arreglo con **N** elementos, donde **X** el la suma máxima de cualquier subsecuencia continua en el arreglo. ¿**Cuantas** subsecuencias continuas en dicho arreglo dan la suma máxima?

Entrada

La primera línea contiene ***T** que es el número de casos de prueba*. A esto le siguen ***2T lineas***, 2 por cada caso. En cada caso, la primer línea corresponde con ***N** que es el número de elementos que hay en el arreglo*, siendo la segunda línea los enteros ***Ai i=1..N*** separados por un espacio.

Salida

La salida consiste de **T** lineas, una por cada caso. En cada línea irán 2 enteros separados por un espacio; el valor de **X** que es *la suma máxima de cualquier subsecuencia*, y la *cantidad de subsecuencias que producen ese valor*.

Ejemplo

Entrada

2 (T)

3 (N1)

-1 -1 -1 (Arreglo 1)

4 (N2)

2 0 -2 2 (Arreglo 2)

Salida

-1 3 (Salida arreglo 1)

2 4 (Salida arreglo 2)

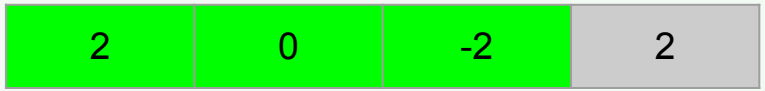
Ideas	Implementación
1. RSQ y luego comparar todos los rangos.	Precalcular el RSQ → O(n) Comparar todos los rangos → $O(n^2)$ TLE
2. Modificar RSQ para iterar las sumas cada vez que entra un valor.	Comparar todos los rangos a medida que entran → $O(n^2)$ TLE
3. Algoritmo de ***Kadane*** (Visto en ***AYED*** pero no con ese nombre, lo tuve que buscar)	No podemos contar las subsecuencias, sólo obtener el máximo en O(n)... Quizás con algo más o una variación se pueda

Algoritmo de Kadane para obtener el máximo

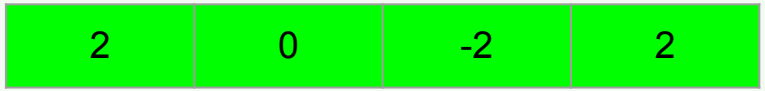
En el algoritmo de kadane, en resumen lo que ocurre es que vamos analizando sumas parciales, donde en cada nuevo elemento incorporado, evaluamos si es conveniente agregarlo al acumulador o comenzar un nuevo acumulado desde la posición actual.

Acumulador = 0 ⇒ Acumulador = 2
Actual = 2 Maximo = 2

Acumulador = 2 ⇒ Acumulador = 2
Actual = 0 Maximo = 2
Actual < Acumulador + Actual ⇒ sigo acumulador



Acumulador = 2 ⇒ Acumulador = 0
Actual = -2 Maximo = 2
Actual < Acumulador + Actual ⇒ sigo acumulador



Acumulador = 0 ⇒ Acumulador = 2
Actual = 2 Maximo = 2
Actual == Acumulador + Actual

Ejemplo con [-1, -1, -1]

En cada paso se descarta el acumulador ya que siempre ocurre que actual es mayor al acumulador incluyendo al actual.

Este es un caso trivial porque el arreglo es todo negativo.

Acumulador = 0 ⇒ Acumulador = -1
Actual = -1 Maximo = -1

Acumulador = -1 ⇒ Acumulador = -1
Actual = -1 Maximo = -1
Actual > Acumulador + Actual ⇒ sigo actual

Acumulador = -1 ⇒ Acumulador = -1
Actual = -1 Maximo = -1

Dado un arreglo con **N** elementos, donde **X** el la suma máxima de cualquier subsecuencia continua en el arreglo. ¿**Cuantas** subsecuencias continuas en dicho arreglo dan la suma máxima?

Con el algoritmo de Kadane tenemos solucionado encontrar el valor máximo.

```
    tam_arr = int(stdin.readline())

    num_arr = [int(elem) for elem in stdin.readline().split()]

    maximo = num_arr[0]
    suma_actual = num_arr[0]

    for I in range(1,tam_arr):
        # continuamos la búsqueda con el acumulado que sea mayor
        suma_actual = max(suma_actual + num_arr[I], num_arr[I])

        maximo = max(suma_actual, maximo)
```

Pero…
¿Como contamos la cantidad de subarreglos cuya suma es el valor de la suma máxima X?

Luego de investigar y pensar por horas…

Opciones

1. Usar el valor máximo obtenido e iterar todos subarreglos con 2 for

Implementación

Volvemos al problema inicial, tenemos $O(n^2)$

TLE

2. Utilizar el algoritmo de suma de prefijos para ir contando los subarreglos que suman el máximo

O(n), utilizamos un diccionario para recordar las sumas acumuladas y comprobar si significa que es válido

Suma de prefijos para contar los subarreglos de suma K

Se busca los arreglos desde el comienzo avanzando progresivamente por las posiciones del arreglo acumulando los valores para comprobar si ya existen acumulados anteriores que puedan indicar un subarreglo de suma K, en nuestro caso de suma máxima previamente calculada.

```
acumulado = 0
acumulados_pasados = {}
acumulados_pasados[0] = 1
ocurrencias = 0

for num in num_arr:
    # acumulado para las sumas parciales entre el comienzo
    # y la posición actual
    acumulado += num
    diferencia = acumulado - maximo

    # si existe una diferencia en acumulados pasados quiere decir
    # que ya existe un acumulado que junto al acumulado actual
    # obtiene el máximo
    if diferencia in acumulados_pasados:
        ocurrencias += acumulados_pasados[diferencia]

    # registramos los acumulados y contamos cuántas veces aparecen
    # esto nos dará el número de subarreglos si llegara a existir
    # una diferencia que con el acumulado nos da el máximo
    if acumulado in acumulados_pasados:
        acumulados_pasados[acumulado] += 1
    else:
        acumulados_pasados[acumulado] = 1
```

2	0	-2	2

acumulados_pasados → { 0:1 }; maximo = 2;

2	0	-2	2

acumulado = 2;

diferencia = acumulado - máximo = 2 - 2 = 0 → existe

acumulados_pasados → { 0:1 }

ocurrencias += acumulados_pasados[diferencia] → ocurrencias = 1

acumulado no existe en acumulados_pasados → se registra

acumulados_pasados → { 0:1, 2:1 }

acumulados_pasados → { 0:1, 2:1 }; maximo = 2;

acumulado = 2 + 0 = 2;

diferencia = acumulado - máximo = 2 - 2 = 0 → existe

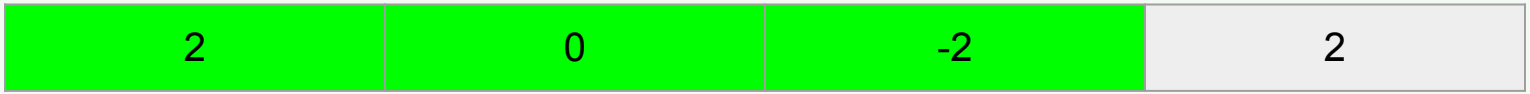
acumulados_pasados → { 0:1, 2:1 }

ocurrencias += acumulados_pasados[diferencia] → ocurrencias = 2

acumulado existe en acumulados_pasados → se registra

acumulados_pasados → { 0:1, 2:2 }

acumulados_pasados → { 0:1, 2:2 }; maximo = 2;



acumulado = 2 + (-2) = 0;

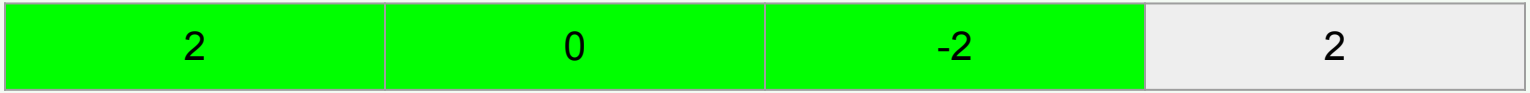
diferencia = acumulado - máximo = 0 - 2 = -2 → no existe

acumulados_pasados → { 0:1, 2:1 }

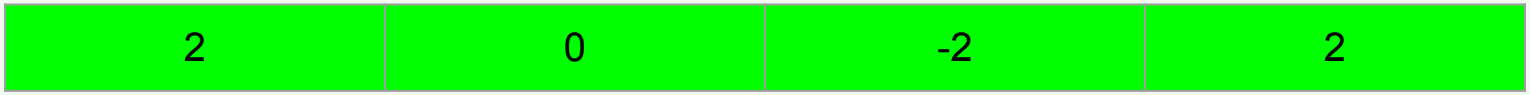
ocurrencias = 2 → no se modifica ya que no existe un acumulado anterior

acumulado existe en acumulados_pasados → se registra

acumulados_pasados → { 0:2, 2:2 }



acumulados_pasados → { 0:2, 2:2 }; maximo = 2;



acumulado = 0 + 2 = 2;

diferencia = acumulado - máximo = 2 - 2 = 0 → existe

acumulados_pasados → { 0:1, 2:2 }

ocurrencias += acumulados_pasados[diferencia] → ocurrencias = 2+2 = 4

acumulado existe en acumulados_pasados → se registra

acumulados_pasados → { 0:2, 2:3 }

Resultado

Como podemos ver, en ocurrencias ahora nos quedó correctamente que hay 4 subarreglos cuyo resultado de la suma es el máximo 2.

Dichos subarreglos son:

[2] (del inicio)

[2, 0]

[2,0,-2,2]

[2] (del final)

Aceptado con 930ms de 1000ms limites
El tiempo puede mejorar si cambiamos a C por ej

33930626	☐	2024-12-09 14:14:07	Maximum Sum Sequences	**accepted** edit ideone it	0.93	99M	PYPY3

Código completo

```python
from sys import stdin, stdout
rta = []
tests = int(stdin.readline())
for _ in range(tests):
    tam_arr = int(stdin.readline())
    num_arr = [int(elem) for elem in stdin.readline().split()]
    maximo = num_arr[0]
    suma_actual = num_arr[0]
    for I in range(1,tam_arr):
        # continuamos la busqueda con el acumulado que sea mayor
        suma_actual = max(suma_actual + num_arr[I], num_arr[I])
        maximo = max(suma_actual, maximo)
    # hago suma de prefijos para contar arreglos con cierta propiedad
    acumulado = 0
    acumulados_pasados = {}
    acumulados_pasados[0] = 1
    ocurrencias = 0
    for num in num_arr:
        # acumulado para las sumas parciales entre el comienzo
        # y la posición actual
        acumulado += num
        diferencia = acumulado - maximo
        # si existe una diferencia en acumulados pasados quiere decir
        # que ya existe un acumulado que junto al acumulado actual
        # obtenemos el máximo
        if diferencia in acumulados_pasados:
            ocurrencias += acumulados_pasados[diferencia]
        # registramos los acumulados y contamos cuantas veces aparecen
        # esto nos dará el número de subarreglos si llegara a existir
        # una diferencia que con el acumulado nos da el máximo
        if acumulado in acumulados_pasados:
            acumulados_pasados[acumulado] += 1
        else:
            acumulados_pasados[acumulado] = 1
    rta.append("{} {}".format(maximo, ocurrencias))
stdout.write("\n".join(rta) + "\n")
```

Algoritmo de Kadane

Algoritmo contar los subarreglos de suma X

Fuentes

- Maximum subarray problem - Wikipedia
- Prefix sum - Wikipedia
- Count Subarray sum Equals K - takeuforward
- Material de AYED 2023

¿Preguntas?