HQ9

// we assume that array indices start from 1

// $S[1...n]$ contains a sequence of positive integers

// $G[1...n]$ contains non negative integers.

1. Given k, compute $P(k)$ = maximum score among all gap subsequences using indices $S[k...n]$

2. $P(n) = S[n]$ // base case

for $k<n$, $P(k) = \max\begin{cases} S[k] + P(1+k+G[k]) \\ P(k+1) \end{cases}$

If $P(1+k+G[k])$ doesn't exist ie., $1+k+G[k]$ exceeds n, then we take zero value, i.e.,

$$P(k) = \max\begin{cases} S[k] + 0 \\ P(k+1) \end{cases}$$

3. Let M be the maximum score among all gap subsequences using indices $S[k...n]$ starting with $S[k]$

now, $m = \max\begin{cases} S[k] + P(j + G[k]) \\ P(j) \end{cases}$

Where $P(j + G[k])$ and $P(j)$ returns the max score among all subsequences using indices $S[j+G[k]...n]$ and $S[j...n]$ respectively where $j>k$ and $j+G[k]>k$

also we can write the above as-

$$m = \max\begin{cases} S[k] + m_1 \\ m_2 \end{cases}$$

where m_1 and m_2 are the respective score of the above two case.

<u>claim</u>: m_1 and m_2 must be the maximum score of $P[j]$ and $P[j+G[k]]$ where $j>k$.

If m_1 and m_2 are not the maximum score among all subsequences for $P(j)$ and $P(j+G[k])$ and instead there were m_1' and m_2' as the maximum score among all the subsequences for $P(j)$ and $P(j+G[k])$ respectively such that $m_1' > m_1$ and $m_2' > m_2$, Then there would exist a

$$\text{maxscore} = \max\begin{cases} S[k] + m_1' \\ m_2' \end{cases}$$

So, maxscore would be greater than m which contradicts our assumption that m is the maximum score of all gap subsequence using indices $S[k \ldots n]$ starts with $S[k]$.

Thus $$P(k) = \max\begin{cases} S[k] + P(j+G[k]) \\ P(j) \end{cases}$$

where max is taken over all j such that $j>k$.

This justifies the recursive formula. If there is no such $P(j+G[k])$ existing we take a zero.

Also for a single element that starts with index k, i.e, $S[k]$ we simply put $P(k) = S[k]$ as max score for all subsequence for a single element is itself.

(4) We are taking a 1D-array $GS[1..n]$ as memo where $GS[i]$ will store $P(i)$.

(5) initialize $GS[k] = P(n) = S[n]$

for $i = n-1$ to 1

compute $GS[i] = P(i)$ using the recursive formula on given array S

6. Maximum score among all gap sub sequences in $S[1 \dots n] = P(1)$

7. Time Complexity $= O(n)$
   Space Complexity $= O(n)$

as computing $GS[j]$ where $1 \le j \le n$ will require $O(1)$ time complexity and there are $O(n)$ entries in the array GS.

8. To compute the list of indices which contribute to the max score of all gap subsequence from $S[1 \dots n]$. We will also store a pointer in GS array that point to other indices of GS array. ~~that point to other~~ We denote the pointer associated with $GS[j]$ as $GS[j].p$.

Let m_j be the max score among all gap subsequence from $S[j \dots n]$ where $GS[j].p$ stores the index of k such that $GS[k]$ is the next element contributing to the max score among all gap subsequences.

The pointer can be computed while calculating values of GS array.

$GS[j].p = null$ when $P(1+j+GS[j])$ and $P(j+1)$ does not exist, and $GS[j].p = \max \begin{cases} P(1+j+GS[j]) + S[j] \\ P(j+1) \end{cases}$

i.e., if $S[j] + P(1+j+GS[j])$ is maximum then $GS[j].p$ will point to index $(1+j+GS[j])$ of $GS[]$ and if $P(j+1)$ is maximum then $GS[j].p$ will point to index $(j+1)$ of $GS[]$.

Finally to print the subsequence which has the max score of all gap subsequence.

```
i = 1
while ( GS[i]. p != NULL)
{   print ( S[GS[i].p])
    i = GS[i].p
}
```

This will print the required subsequence starting from GS[1] untill we hit a null.