CSE525 Monsoon 2020 — Pramil Panjawani (PhD19008)
Homework 8 Problem 1 — Ashima Garg (PhD19003)

You CANNOT consult any other person or online resource for solving the homework problems. You can definitely ask the instructor or TAs for hints and you are encourage to do so (in fact, you will get useful hints if you ask for help at least 1-2 days before the due date). If we find you guilty of academic dishonesty, penalty will be imposed as per institute guidelines.

Solution:

- **Problem Definition**: Generate new $endGap[1 \dots n]$ that stores $i + Gap[i]$ $\forall i \in 1 \dots n$. Given i, result[i]=maxscore of the gap sequence S[i ... n] that begins with S[i] whose next element to be considered is $endGap[i] + 1$.
- **Formula for P**: $gapSequence[n].v = S[n]$. For $i < n$, $gapSequence[i].v = max(S[i] + gapSequence[j].v$: $\forall j \in endGap[i] + 1 \dots n)$ If $endGap[i] + 1 \geq n$ then $gapSequence[i] = S[i]$. There is no element after nth index, therefore, maxscore of the sequence starting with $S[n]$ is $S[n]$ itself. For all $i < n$, we consider maximum scores of all the positions that have atleast gap of $endGap[i] + 1$ and therefore, for max score starting with $S[i]$ will be S[i] + maximum score so far for the allowable indices. Hence, gap sequence of $S[1 \dots n]$ given $Gap[1 \dots n]$ is S[0](=0) + maximum score so far for the all the allowable indices i.e. 1 to n.
- **Memoization Structure**: 1-D array result[0 ... n]. result[i].v stores the maxscore of gapSequence[i], and result[i].p stores the index of the next element in result after S[i]
- **How to fill the memo**: Initialize the $endGap[1 \dots n]$ as defined in the problem statement. Initialize $result[n].v = gapSequence[n] = S[n]$. Add sentinel $S[0] = 0$ and $endGap[0] = 1$. For $i = n \dots 0$, compute $result[i].v = gapSequence[i]$ using the recursive formula on the sequence S and $endGap$.
- **How to solve original problem from memo**: Max score of gap sequence of S given $Gap =$ result[0].v. This is because gap sequence of S will always start with S[0], since, the endGap[0] i.e. next element to be considered is 1, and hence the rest of the sequence must be gap sequence of S.
- **Space and time complexity**: Space complexity is $O(n) + O(n) + O(n) = O(n)$ for creating new $endGap$ array, for storing the maxscore for every ith element ($result.v$) and for index of the next element after i ($result.p$). Worst case time complexity is $O(n^2)$. since computing $result[i]$ requires taking the maxscore of atmost $n - endGap[i] - 1 \leq n$ values and there are $O(n)$ entries in result and generating the $endGap$ array requires one traversal of the entire gap array, therefore it has $O(n)$.
- **To Obtain gap sequence**: First fill the memo. Use the pointers ($result.p$) to store pointers that points to index of the next element k s.t. S[k] is the next element in S for the gap sequence starting with $S[i]$ and $k >= endGap[i]+1$.

To print gap sequence of S with Gap:

```
i = 0
while(result[i].p != NULL):
    print(S[result[i].p])
    i = result[i].p
```

```
⟨⟨Generates the index of the element to be considered for computing the gap sequence, i.e. i+Gap[i]⟩⟩
def findEndPosition(Gap[1 ... n]): ⟨⟨Time complexity is O(n)⟩⟩
    endGap = [0] * n
    for i = 1 ... n
        endGap[i] = i + Gap[i]
    return endGap
def gapSequence(S[0 ... n], endGap[0 ... n], result[0ldotsn]):
    S[0] = 0, endGap[0] = 1
    for i=1 ... n:
        result[i].v ← S[i], result[i].p ← NULL
        for j = endGap[i] + 1 to n
            if (S[i] + result[j].v > result[i].v):
                result[i].v ← result[i].v + S[i]
                result[i].p ← j ⟨⟨Stores the index of next element in result after S[i]⟩⟩
def main(S[1 ... n], Gap[1 ... n]):
    endGap = findEndPosition(Gap[1 ... n])
    S[0] = 0, endGap[0] = 1 ⟨⟨Add sentinel⟩⟩
    result.v=[]*n , result.p=[]*n ⟨⟨result.v stores maxscore, results.p stores index of the next element in the sequence⟩⟩
    gapSequence(S, endGap, result)
    return result.p
```

■