Maximum Incrementing Subset

Problem

- You are given an array
  - Find the longest contiguous sub-array that is monotonically increasing

1	2	4	1	2	1	3	4	7	1	2	4	3	2	4	1	2	4	5	9	1	2	3	4	8	9	1	2	4	3	2

Problem

1	2	4	1	2	1	3	4	7	1	2	4	3	2	4	1	2	4	5	9	**1**	**2**	**3**	**4**	**8**	**9**	1	2	4	3	2

Naïve Solution

- Investigate each position of the array as a possible solution

```
def lis(array):
    besti, bestj = -1,-1
    for i in range(len(array)):
        j=i+1
        while j<len(array) and array[j]>= array[j-1]:
            j+=1
        if j-i > bestj-besti:
            besti, bestj = i, j-1
        print(i, array[i], besti, bestj)
    return besti, bestj
```

Naïve Solution

- Worst case run-time:
  - If the array is monotonically increasing:
    - Compare $(n-1)+(n-2)+\ldots+3+2+1=\dfrac{n(n-1)}{2}$ elements

Dynamic programming solution

- For each slice `A[:k]`:
  - Maintain
    - `j` such that `j` is minimal and `A[j:k]` is monotonically increasing
    - `r,s` such that `A[r:s]` is the longest monotonically increasing subarray in `A[:k]`

Dynamic programming solution

- To move from `A[:k]` to `A[:k+1]` :
  - Maintain the loop invariants
    - If `A[k] >= A[k-1]`:
      - Still increasing, so *j* does not change
    - Compare `k+1-j` with `r-s` and if the first is larger, we have found a new champion:
      - set `r,s = j,k`
    - Otherwise:
      - `r, s` does not change
      - `j` is set to `k`