The Maximum Subarray Problem
A DP Approach

The Maximum Subarray Problem: A DP solution

Input: Profit history of a company. Money earned/lost each year.

Year	1	2	3	4	5	6	7	8	9
Profit (M$)	3	2	1	-7	5	2	-1	3	-1

Problem: Find the span of years in which the company earned the most

Answer: Year 5-8 , 9 M$

Formal definition:

Input: An array of numbers $A[1 \dots n]$, both positive and negative

Output: Find the maximum $V(k, i)$, where $V(i, j) = \sum_{j=k}^{i} A[j]$

Recall

Previously learnt 4 different algorithms for solving this problem

- $\Theta(n^3)$ Brute force Algorithm
- $\Theta(n^2)$ (Reuse of Information) Algorithm
- $\Theta(n \log n)$ Divide-and-Conquer Algorithm
- $\Theta(n)$ Linear Scan Algorithm

Now

Design a $\Theta(n)$ Dynamic Programming Algorithm

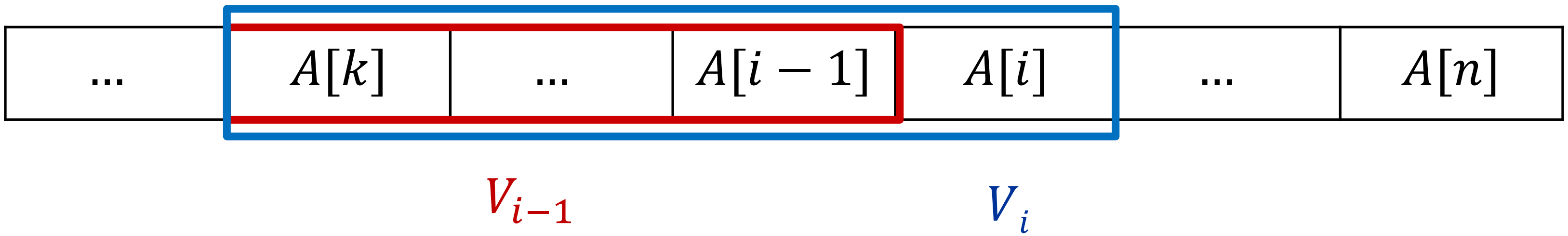
A dynamic programming ($\Theta(n)$) algorithm

Define: V_i to be max value subarray ending at $A[i]$

$$V_i = \max_{1 \le k \le i} V(k, i)$$

The main observation is that if $V_i \neq A[i] = V(i, i)$ then

$$V_i = A[i] + \max_{1 \le k < i} V(k, i-1) = A[i] + V_{i-1}$$



This immediately implies DP Recurrence

$$V_i = \begin{cases} A[1] & \text{if } i = 1 \\ \max\{A[i], A[i] + V_{i-1}\} & \text{if } i > 1 \end{cases}$$

The DP recurrence

We just saw

$$V_i = \begin{cases} A[1] & \text{if } i = 1 \\ \max\{A[i], A[i] + V_{i-1}\} & \text{if } i > 1 \end{cases} \quad \text{where} \quad V_i = \max_{1 \le k \le i} V(k, i)$$

Original problem then becomes finding i' such that

$$V_{i'} = \max_{1 \le i \le n} V_i$$

The DP recurrence permits constructing V_i in O(1) time from V_{i-1}.

$\Rightarrow$ We can construct $V_1, V_2, \dots, V_n$ in order in O(n) total time while keeping track of the largest V_i found so far

$\Rightarrow$ This finds $V_{i'}$ in O(n) total time, solving the problem.

Note: This algorithm turns out to be very similar to the linear scan algorithm we developed in class, but found using DP reasoning

Implementation

Derived recurrence that

$$V_i = \begin{cases} A[1] & \text{if } i = 1 \\ \max\{A[i], A[i] + V_{i-1}\} & \text{if } i > 1 \end{cases}$$ where $$V_i = \max_{1 \le k \le i} V(k, i)$$

and need to find i' such that

$$V_{i'} = \max_{1 \le i \le n} V_i$$

This is very straightforward.
Next slides give actual code, and a worked example

Version 1

Store V_i in a table $V[1,2,\ldots,n]$, at each step calculating $V[i]$ from $V[i-1]$

Base condition: $V[1] \leftarrow A[1]$

Recurrence: $V[i] \leftarrow \max(A[i], A[i] + V[i-1])$

```
let V[1,2,...,n] be an array storing V_i
V[1] ← A[1]
V_max ← A[1]
for i ← 2 to n do
      V[i] ← max(A[i], A[i] + V[i − 1])
      if V_max < V[i]
               then V_max ← V[i]
      end if
return V_max
```

Running time: $\Theta(n)$

i	1	2	3	4	5	6	7	8	9
$A[i]$	3	2	1	-7	5	2	-1	3	-1
$V[i]$	3	5	6	-1	5	7	6	9	8
V_{max}	3	5	6	6	6	7	7	9	9

Solution is $V[8]$

Version 2

Simplified: We only need to remember the last V_i (call it V) and V_{max}

Base condition: $V \leftarrow A[1]$

Recurrence: $V \leftarrow \max(A[i], A[i] + V)$

```
V ← A[1]
V_max ← A[1]
for i ← 2 to n do
      V ← max(A[i], A[i] + V)
      if V_max < V
            then V_max ← V
      end if
return V_max
```

Running time:
$\Theta(n)$

This gets same result as Version 1, but is simpler!

Next pages provide a detailed walk-through of how Version 1 fills in the DP table.

Version 1

Store V_i in a table $V[1,2,\ldots,n]$, at each step calculating $V[i]$ from $V[i-1]$

Base condition: $V[1] \leftarrow A[1]$ Recurrence: $V[i] \leftarrow \max(A[i], A[i] + V[i-1])$

```
let V[1,2,...,n] be an array storing V_i
V[1] ← A[1]
V_max ← A[1]
for i ← 2 to n do
      V[i] ← max(A[i], A[i] + V[i-1])
      if V_max < V[i]
               then V_max ← V[i]
      end if
return V_max
```

Running time:
$\Theta(n)$

i	1	2	3	4	5	6	7	8	9
$A[i]$	3	2	1	-7	5	2	-1	3	-1
$V[i]$	3								
V_{max}	3								

$V_{max} = V[1] = A[1] = 3$

Version 1

Store V_i in a table $V[1, 2, \ldots, n]$, at each step calculating $V[i]$ from $V[i-1]$

Base condition: $V[1] \leftarrow A[1]$ Recurrence: $V[i] \leftarrow \max(A[i], A[i] + V[i-1])$

```
let V[1,2,…,n] be an array storing V_i
V[1] ← A[1]
V_max ← A[1]
for i ← 2 to n do
      V[i] ← max(A[i], A[i] + V[i − 1])
      if V_max < V[i]
                then V_max ← V[i]
      end if
return V_max
```

Running time:
$\Theta(n)$

i	1	2	3	4	5	6	7	8	9
$A[i]$	3	2	1	-7	5	2	-1	3	-1
$V[i]$	3	5							
V_{max}	3	5							

$V_{max} = \max(A[2], A[2] + V[1]) = \max(2, 2 + 3) = 5$

Version 1

Store V_i in a table $V[1, 2, \ldots, n]$, at each step calculating $V[i]$ from $V[i-1]$

Base condition: $V[1] \leftarrow A[1]$ Recurrence: $V[i] \leftarrow \max(A[i], A[i] + V[i-1])$

```
let V[1,2,...,n] be an array storing V_i
V[1] ← A[1]
V_max ← A[1]
for i ← 2 to n do
      V[i] ← max(A[i], A[i] + V[i − 1])
      if V_max < V[i]
            then V_max ← V[i]
      end if
return V_max
```

Running time: $\Theta(n)$

i	1	2	3	4	5	6	7	8	9
$A[i]$	3	2	1	-7	5	2	-1	3	-1
$V[i]$	3	5	6						
V_{max}	3	5	6						

$V_{max} = \max(A[3], A[3] + V[2]) = \max(1, 1 + 5) = 6$

Version 1

Store V_i in a table $V[1,2,\ldots,n]$, at each step calculating $V[i]$ from $V[i-1]$

Base condition: $V[1] \leftarrow A[1]$ Recurrence: $V[i] \leftarrow \max(A[i], A[i] + V[i-1])$

```
let V[1,2,...,n] be an array storing V_i
V[1] ← A[1]
V_max ← A[1]
for i ← 2 to n do
      V[i] ← max(A[i], A[i] + V[i − 1])
      if V_max < V[i]
               then V_max ← V[i]
      end if
return V_max
```

Running time:
$\Theta(n)$

i	1	2	3	4	5	6	7	8	9
$A[i]$	3	2	1	-7	5	2	-1	3	-1
$V[i]$	3	5	6	-1					
V_{max}	3	5	6	6					

$V_{max} = 6 > \max(A[4], A[4] + V[3]) = \max(-7, -7+6) = -1$

Version 1

Store V_i in a table $V[1, 2, \ldots, n]$, at each step calculating $V[i]$ from $V[i-1]$

Base condition: $V[1] \leftarrow A[1]$ Recurrence: $V[i] \leftarrow \max(A[i], A[i] + V[i-1])$

```
let V[1,2,...,n] be an array storing V_i
V[1] ← A[1]
V_max ← A[1]
for i ← 2 to n do
      V[i] ← max(A[i], A[i] + V[i − 1])
      if V_max < V[i]
               then V_max ← V[i]
      end if
return V_max
```

Running time: $\Theta(n)$

i	1	2	3	4	5	6	7	8	9
$A[i]$	3	2	1	-7	5	2	-1	3	-1
$V[i]$	3	5	6	-1	5				
V_{max}	3	5	6	6	6				

$V_{max} = 6 > \max(A[5], A[5] + V[4]) = \max(5, 5-1) = 5$

Version 1

Store V_i in a table $V[1,2,\ldots,n]$, at each step calculating $V[i]$ from $V[i-1]$

Base condition: $V[1] \leftarrow A[1]$ Recurrence: $V[i] \leftarrow \max(A[i], A[i]+V[i-1])$

```
let V[1,2,...,n] be an array storing V_i
V[1] ← A[1]
V_max ← A[1]
for i ← 2 to n do
      V[i] ← max(A[i], A[i] + V[i − 1])
      if V_max < V[i]
                then V_max ← V[i]
      end if
return V_max
```

Running time:
$\Theta(n)$

i	1	2	3	4	5	6	7	8	9
$A[i]$	3	2	1	-7	5	2	-1	3	-1
$V[i]$	3	5	6	-1	5	7			
V_{max}	3	5	6	6	6	7			

$V_{max} = \max(A[6], A[6]+V[5]) = \max(2, 2+5) = 7$

Version 1

Store V_i in a table $V[1,2,\ldots,n]$, at each step calculating $V[i]$ from $V[i-1]$

Base condition: $V[1] \leftarrow A[1]$ Recurrence: $V[i] \leftarrow \max(A[i], A[i] + V[i-1])$

```
let V[1,2,...,n] be an array storing V_i
V[1] ← A[1]
V_max ← A[1]
for i ← 2 to n do
      V[i] ← max(A[i], A[i] + V[i-1])
      if V_max < V[i]
               then V_max ← V[i]
      end if
return V_max
```

Running time:
$\Theta(n)$

i	1	2	3	4	5	6	7	8	9
$A[i]$	3	2	1	-7	5	2	-1	3	-1
$V[i]$	3	5	6	-1	5	7	6		
V_{max}	3	5	6	6	6	7	7		

$V_{max} = 7 > \max(A[7], A[7] + V[6]) = \max(-1, -1 + 7) = 6$

Version 1

Store V_i in a table $V[1, 2, \dots, n]$, at each step calculating $V[i]$ from $V[i-1]$

Base condition: $V[1] \leftarrow A[1]$ Recurrence: $V[i] \leftarrow \max(A[i], A[i] + V[i-1])$

```
let V[1,2,…,n] be an array storing V_i
V[1] ← A[1]
V_max ← A[1]
for i ← 2 to n do
      V[i] ← max(A[i], A[i] + V[i − 1])
      if V_max < V[i]
                then V_max ← V[i]
      end if
return V_max
```

Running time:
$\Theta(n)$

i	1	2	3	4	5	6	7	8	9
$A[i]$	3	2	1	-7	5	2	-1	3	-1
$V[i]$	3	5	6	-1	5	7	6	9	
V_{max}	3	5	6	6	6	7	7	9	

$V_{max} = \max(A[8], A[8] + V[7]) = \max(3, 3+6) = 9$

Version 1

Store V_i in a table $V[1,2,\ldots,n]$, at each step calculating $V[i]$ from $V[i-1]$

Base condition: $V[1] \leftarrow A[1]$ Recurrence: $V[i] \leftarrow \max(A[i], A[i] + V[i-1])$

```
let V[1,2,…,n] be an array storing V_i
V[1] ← A[1]
V_max ← A[1]
for i ← 2 to n do
      V[i] ← max(A[i], A[i] + V[i − 1])
      if V_max < V[i]
               then V_max ← V[i]
      end if
return V_max
```

Running time: $\Theta(n)$

i	1	2	3	4	5	6	7	8	9
$A[i]$	3	2	1	-7	5	2	-1	3	-1
$V[i]$	3	5	6	-1	5	7	6	9	8
V_{max}	3	5	6	6	6	7	7	9	9

$V_{max} = 9 > \max(A[9], A[9] + V[8]) = \max(-1, -1 + 9) = 8$

Version 1

Store V_i in a table $V[1,2,\dots,n]$, at each step calculating $V[i]$ from $V[i-1]$

Base condition: $V[1] \leftarrow A[1]$ Recurrence: $V[i] \leftarrow \max(A[i], A[i] + V[i-1])$

```
let V[1,2,...,n] be an array storing V_i
V[1] ← A[1]
V_max ← A[1]
for i ← 2 to n do
      V[i] ← max(A[i], A[i] + V[i − 1])
      if V_max < V[i]
               then V_max ← V[i]
      end if
return V_max
```

Running time: $\Theta(n)$

i	1	2	3	4	5	6	7	8	9
$A[i]$	3	2	1	-7	5	2	-1	3	-1
$V[i]$	3	5	6	-1	5	7	6	**9**	8
V_{max}	3	5	6	6	6	7	7	9	9

Solution is $V[8]$

$V_{max} = 9 > \max(A[9], A[9] + V[8]) = \max(-1, -1 + 9) = 8$