The Maximum Subarray Problem (MSP)

AN APPLICATION OF DIVIDE AND CONQUER

Background

- Divide-n-Conquer consists of three phases

Divide
- Break the problem into a number of subproblems that are smaller instances of the same problem

Conquer
- Solve the sub-problems recursively. Solve the smallest problems directly

Combine
- Combine the solutions to the subproblems into the solution for the original problem

A ***recurrence*** is an equation or inequality that describes a function in terms of its value on smaller inputs.

Example – Stock Trading

- Let you are interested in trading shares of a company
- You can buy one unit of stock only one time and then sell it at a later date and can learn what the price of the stock will be in the future.
- The goal is to maximize your profit by using "buy low and sell high" principle
- Within a given period, you might not be able to buy at the lowest price and then sell at the highest price

Share Price Details

- we can maximize profit by buying at the lowest price, after day 7.
- We will find the lowest price point (LPP) and highest price point (HPP)
  - The pair (LPP, HPP) will give us maximum profit
- Counter Example shows that day (2,3) gives maximum profit however
  - Price on day is neither lowest nor price at day 3 is highest

Day	0	1	2	3	4
Price	10	11	7	10	6
Change		13	-3	-25	20

Day	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Price	100	113	110	85	105	102	86	63	81	101	94	106	101	79	94	90	97
Change		13	-3	-25	20	-3	-16	-23	18	20	-7	12	-5	-22	15	-4	7

By: Dr. Sajid Iqbal, COMputer EDucation eXplaineD - COMEDXD

A brute-force solution

- Try every possible pair of buy and sell dates in which the buy date precedes the sell date
- A period of n-days has $\binom{n}{2} = \frac{n!}{(2!)(n-2)!}$ pairs. It has time complexity $\theta(n^2)$
  - $\binom{5}{2} = \frac{5!}{(2!)(5-2)!} = \frac{5!}{2!\cdot 3!} = \frac{5\cdot 4}{2} = 10 \rightarrow \{(1,2),(1,3),(1,4),(1,5),(2,3),(2,4),(2,5),(3,4),(3,5),(4,5)$
- we can hope to evaluate each pair of dates in constant time,
- this approach would take $\Omega(n^2)$ time

A transformation

- Find the sequence of days over which the net change from the first day to the last is maximum.
- Instead of looking at the daily prices, we consider the daily change in price
  - The change on day i is the difference between the prices after day $i - 1$ and after day i.
- ***Maximum subarray*:** Find the nonempty, contiguous subarray of A (last row of table) whose values have the largest sum.
  - In our example, Maximum Sub-Array is A[8-11] with sum=43
    - Thus, buy the stock just before day 8 (after day 7) and sell it after day 11, earning a profit of $43 per share.
    - There could be multiple sub-arrays with same profit

Day	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Price	100	113	110	85	105	102	86	63	81	101	94	106	101	79	94	90	97
Change		13	-3	-25	20	-3	-16	-23	18	20	-7	12	-5	-22	15	-4	7

Divide and Conquer Solution of Problem

By: Dr. Sajid Iqbal, COMputer EDucation eXplaineD - COMEDXD

Maximum Sub-Array (MSA) Problem

- Divide the subarray into two subarrays of as equal size as possible
  - $L[low, \dots, mid], R[mid + 1, \dots, high]$
  - Let $MSA[i, \dots . j]$ be a maximum sub-array, it can now lie
    - In $L[low, \dots, mid]$ such that $low \leq i \leq j \leq mid$
    - In $R[mid + 1, \dots, high]$ such that $mid + 1 \leq i \leq j \leq high$
    - Crossing the mid-point such that $low \leq i \leq mid < j \leq high$
    - MSA will have maximum sum over all sub-arrays of L, R or crossing over

Solving MSA

- Sub-Arrays L, R can be solved recursively
- How to solve cross-over MSA
  - Any subarray crossing the midpoint is itself made of two subarrays
    - $MSA[i, \dots, mid], MSA[mid + 1, \dots, j]$
    - So find these and combine them
- Hence the problem is solved using two sub-routines
  - $FIND\_MAXIMUM\_SUBARRAY(A, low, high)$
  - $FIND\_MAX\_CROSSING\_SUBARRAY(A, low, mid, high)$

FIND_MAXIMUM_SUBARRAY

	$Find\_Maximum\_Subarray(A, low, high)$
1	$if\ (high == low)$
2	$\quad return\ (low, high, A[low])$
3	$else\ mid = int(low + high)/2$
4	$\quad (left_{low}, left_{high}, left_{sum}) = Find\_Maximum\_Subarray(A, low, mid)$
5	$\quad (right_{low}, right_{high}, right_{sum}) = Find\_Maximum\_Subarray(A, mid + 1, high)$
6	$\quad (cross_{low}, cross_{high}, cross_{sum}) = Find\_Max\_Cross\_Subarray(A, low, mid, high)$
7	$\quad if\ (left_{sum} \geq right_{sum}\ and\ left_{sum} \geq cross_{sum})$
8	$\quad\quad return\ (left_{low}, left_{high}, left_{sum})$
9	$\quad else\ if\ (right_{sum} \geq left_{sum}\ and\ right_{sum} \geq cross_{sum})$
10	$\quad\quad return\ (right_{low}, right_{high}, right_{sum})$
11	$\quad else\ return(cross_{low}, cross_{high}, cross_{sum})$

By: Dr. Sajid Iqbal, COMputer EDucation eXplained - COMEDXD

Find_Max_Subarray Procedure

- Direct Solution (Non Recursive): Base case
  - **Line 1,2** : if there is only one element
  - So return that element with low and high indexes
- **Lines 3–11** handle the recursive case.
  - **Line 3**: It computes the mid point of the array
  - **Lines 4 and 5**: These lines conquer by recursively finding MSA within the left and right subarrays, respectively.
  - **Lines 6–11**: form the combine part.
  - **Line 6** : finds a MSA that crosses the midpoint.
  - **Line 7:** tests whether the left subarray contains an MSA
  - **line 8:** returns that MSA.
  - **line 9:** tests whether the right subarray contains an MSA
  - **line 10:** returns that MSA.
  - **line 11:** returns the MSA that crosses the mid point

	$Find\_Maximum\_Subarray(A, low, high)$
1	$if\,(high == low)$
2	$\quad return\ (low, high, A[low])$
3	$else\ mid = int(low + high)/2$
4	$\quad (left_{low}, left_{high}, left_{sum}) =$ $Find\_Maximum\_Subarray(A, low, mid)$
5	$\quad (right_{low}, right_{high}, right_{sum}) =$ $Find\_Maximum\_Subarray(A, mid + 1, high)$
6	$\quad (cross_{low}, cross_{high}, cross_{sum}) =$ $Find\_Max\_Cross\_Subarray(A, low, mid, high)$
7	$\quad if\,(left_{sum} \geq right_{sum}\ and\ left_{sum} \geq cross_{sum})$
8	$\quad\quad return\ (left_{low}, left_{high}, left_{sum})$
9	$\quad else\ if\,(right_{sum} \geq left_{sum}\ and\ right_{sum} \geq cross_{sum})$
10	$\quad\quad return\ (right_{low}, right_{high}, right_{sum})$
11	$\quad else\ return(cross_{low}, cross_{high}, cross_{sum})$

FIND_MAX_CROSSING_SUBARRAY

	Find_Max_Crossing_Subarray(A,low,mid,high)
1	$left\_sum = -\infty$
2	$sum = 0$
3	$for\ i = mid\ \textbf{downto}\ low$
4	$\quad sum = sum + A[i]$
5	$\quad if\ (sum > left_{sum})$
6	$\quad\quad left_{sum} = sum$
7	$\quad\quad maximum_{left} = i$
8	$right_{sum} = -\infty$
9	$sum = 0$
10	$for\ j = mid + 1\ to\ high$
11	$\quad sum = sum + A[j]$
12	$\quad if(sum > right_{sum})$
13	$\quad\quad right_{sum} = sum$
14	$\quad\quad Maximum_{right} = j$
15	$return\ (maximum_{left}, maximum_{right}, left_{sum} + right_{sum})$

- This problem is *not* a smaller instance of our original problem
  - As it has the added restriction that the subarray it chooses must cross the midpoint.
  - We know that MSA has two parts $\boldsymbol{MSA[i, \dots, mid], MSA[mid + 1, \dots, j]}$
  - **Line 1-2**: variables initialization
  - **Line 4-7**: find the maximum sum of left part of array A
    - $Left_{sum}$ contains the maximum sum found so far in left array
  - **Line 8-9**: variables initialization
  - **Line 10-14**: find the maximum sum of right part of Array A
    - $Right_{sum}$ contains the maximum sum found so far in right array
  - **Line 15**: return the maximum left and right points along with total maximum sum

FIND_MAX_CROSSING_SUBARRAY

	Find_Max_Crossing_Subarray(A,low,mid,high)
1	$left\_sum = -\infty$
2	$sum = 0$
3	$for\ i = mid\ \textbf{downto}\ low$
4	$\quad sum = sum + A[i]$
5	$\quad if\ (sum > left_{sum})$
6	$\quad\quad left_{sum} = sum$
7	$\quad\quad maximum_{left} = i$
8	$right_{sum} = -\infty$
9	$sum = 0$
10	$for\ j = mid + 1\ to\ high$
11	$\quad sum = sum + A[j]$
12	$\quad if(sum > right_{sum})$
13	$\quad\quad right_{sum} = sum$
14	$\quad\quad Maximum_{right} = j$
15	$return\ (maximum_{left}, maximum_{right}, left_{sum} + right_{sum})$

- This problem is *not* a smaller instance of our original problem
  - As it has the added restriction that the subarray it chooses must cross the midpoint.
  - We know that MSA has two parts $\boldsymbol{MSA[i, \dots, mid], MSA[mid + 1, \dots, j]}$
  - **Line 1-2**: variables initialization
  - **Line 4-7**: find the maximum sum of left part of array A
    - $Left_{sum}$ contains the maximum sum found so far in left array
  - **Line 8-9**: variables initialization
  - **Line 10-14**: find the maximum sum of right part of Array A
    - $Right_{sum}$ contains the maximum sum found so far in right array
  - **Line 15**: return the maximum left and right points along with total maximum sum

Change		13	-3	-25	20	-3	-16	-23	18	20	-7	12	-5	-22	15	-4	7

By: Dr. Sajid Iqbal, Education explained - COMEDXD

Algorithm Time Complexity

- $T(n) = 1 + 1 + \frac{n}{2} + 1 + \frac{n}{2} + \frac{n}{2} + \frac{n}{4} + \frac{n}{4} + 1 + 1 + \frac{n}{2} + 1 + \frac{n}{2} + \frac{n}{2} + \frac{n}{4} + \frac{n}{4} + 1$
- $\boldsymbol{T(n) = 7 + 4\left(\frac{n}{4}\right) + 6\left(\frac{n}{2}\right)}$
- $\boldsymbol{T(n) = 7 + n + 3n}$
- $\boldsymbol{T(n) = 4n + 7}$
- $\boldsymbol{T(n) = 4n}$
- $\boldsymbol{T(n) = \theta(n)}$

	Find_Max_Crossing_Subarray(A,low,mid,high)	T(n)
1	$left\_sum = -\infty$	1
2	$sum = 0$	1
3	$for\ i = mid\ \boldsymbol{downto}\ low$	$n/2 + 1$
4	$\quad sum = sum + A[i]$	$n/2$
5	$\quad if\ (sum > left_{sum})$	$n/2$
6	$\quad\quad left_{sum} = sum$	$n/4$
7	$\quad\quad maximum_{left} = i$	$n/4$
8	$right_{sum} = -\infty$	1
9	$sum = 0$	1
10	$for\ j = mid + 1\ to\ high$	$\frac{n}{2} + 1$
11	$\quad sum = sum + A[j]$	$n/2$
12	$\quad if(sum > right_{sum})$	$n/2$
13	$\quad\quad right_{sum} = sum$	$n/4$
14	$\quad\quad Maximum_{right} = j$	$n/4$
15	$return\ (maximum_{left}, maximum_{right}, left_{sum} + right_{sum})$	1

Algorithm Analysis

- $T(n) = 1 + 1 + 1 + T\left(\frac{n}{2}\right) + T\left(\frac{n}{2}\right) + \theta(n) + 1 + 1 + 1 + 1$
- $T(n) = 2T\left(\frac{n}{2}\right) + \theta(n) + 7$
- $T(n) = 2T\left(\frac{n}{2}\right) + \theta(n)$
- $\boldsymbol{T(n) = \theta(nlogn)}$

	$Find\_Maximum\_Subarray(A, low, high)$	Cost $T(n)$
1	$if\,(high == low)$	1
2	$\quad return\ (low, high, A[low])$	1
3	$else\ mid = int(low + high)/2$	1
4	$\quad (left_{low}, left_{high}, left_{sum}) = Find\_Maximum\_Subarray(A, low, mid)$	$T(\frac{n}{2})$
5	$\quad (right_{low}, right_{high}, right_{sum}) = Find\_Maximum\_Subarray(A, mid + 1, high)$	$T(\frac{n}{2})$
6	$\quad (cross_{low}, cross_{high}, cross_{sum}) = Find\_Max\_Cross\_Subarray(A, low, mid, high)$	$\theta(n)$
7	$\quad if\,(left_{sum} \geq right_{sum}\ and\ left_{sum} \geq cross_{sum})$	1
8	$\quad\quad return\ (left_{low}, left_{high}, left_{sum})$	1
9	$\quad else\ if\,(right_{sum} \geq left_{sum}\ and\ right_{sum} \geq cross_{sum})$	1
10	$\quad\quad return\ (right_{low}, right_{high}, right_{sum})$	1
11	$\quad else\ return(cross_{low}, cross_{high}, cross_{sum})$	1

Space Com

- $S(n) = n + 3 + 6$
- $S(n) = n + 9$
- $S(n) = n$
- 0300-3014469

	Find_Max_Crossing_Subarray(A,low,mid,high)	$S(n) = n + 3$
1	$left\_sum = -\infty$	1
2	$sum = 0$	1
3	$for\ i = mid\ \textbf{downto}\ low$	1
4	$\quad sum = sum + A[i]$	0
5	$\quad if\ (sum > left_{sum})$	0
6	$\quad\quad left_{sum} = sum$	0
7	$\quad\quad maximum_{left} = i$	0
8	$right_{sum} = -\infty$	1
9	$sum = 0$	0
10	$for\ j = mid + 1\ to\ high$	1
11	$\quad sum = sum + A[j]$	0
12	$\quad if(sum > right_{sum})$	0
13	$\quad\quad right_{sum} = sum$	0
14	$\quad\quad Maximum_{right} = j$	0
15	$return\ (maximum_{left}, maximum_{right}, left_{sum} + right_{sum})$	1

Space Complexity

- $S(n) = n + 10$
- $S(n) = n$
- $S(n) = 0$

	$Find\_Maximum\_Subarray(A, low, high)$	Cost $T(n)$
1	$if\,(high == low)$	0
2	$\quad return\ (low, high, A[low])$	0
3	$else\ mid = int(low + high)/2$	1
4	$\quad (left_{low}, left_{high}, left_{sum}) =$ $Find\_Maximum\_Subarray(A, low, mid)$	3
5	$\quad (right_{low}, right_{high}, right_{sum}) =$ $Find\_Maximum\_Subarray(A, mid + 1, high)$	3
6	$\quad (cross_{low}, cross_{high}, cross_{sum}) =$ $Find\_Max\_Cross\_Subarray(A, low, mid, high)$	3
7	$\quad if\,(left_{sum} \geq right_{sum}\ and\ left_{sum} \geq cross_{sum})$	0
8	$\quad\quad return\ (left_{low}, left_{high}, left_{sum})$	0
9	$\quad else\ if\,(right_{sum} \geq left_{sum}\ and\ right_{sum} \geq cross_{sum})$	0
10	$\quad\quad return\ (right_{low}, right_{high}, right_{sum})$	0
11	$\quad else\ return(cross_{low}, cross_{high}, cross_{sum})$	0

```
def find_max_crossing_subarray(A, low, mid, high):
    left_sum = -10000
    sum = 0;
    max_left,max_right=0,0
    for i in range(mid, low , -1):
        sum = sum + A[i]
        if (sum > left_sum):
            left_sum = sum
            max_left=i

    right_sum = -1000
    sum = 0;
    for j in range(mid + 1, high):
        sum = sum + A[j]
        if (sum > right_sum):
            right_sum = sum
            max_right=j

    return max_left,max_right,max(left_sum + right_sum, left_sum, right_sum)
```

```
def find_maximum_subarray(A, low, high):
    # Base Case: Only one element
    if (low == high):
        return low,high,A[low]

    mid = (low + high) // 2
    left_low,left_high,left_sum=find_maximum_subarray(A, low, mid)
    right_low,right_high,right_sum=find_maximum_subarray(A, mid + 1, high)
    cross_low,cross_high,cross_sum=find_max_crossing_subarray(A, low, mid, high)
    if(left_sum>=right_sum and left_sum>=cross_sum):
        return left_low,left_high,left_sum
    elif (right_sum>=left_sum and right_sum>=cross_sum):
        return right_low,right_high,right_sum
    else:
        print(cross_low,cross_high,cross_sum)
        return cross_low,cross_high,cross_sum
```

```
# Driver Code
#    0  1  2  3  4  5  6  7  8  9  10  11 12 13 14 15
A = [-13,-3,-25,20,-3,-16,-23,18,20,-7,12,-5,-22,15,-4,7]
n = len(A)

low,high,max_sum = find_maximum_subarray(A, 0, n - 1)
print("Maximum contiguous sum is ",
low+1,high+1,max_sum)
```