FRESNO STATE
Discovery. Diversity. Distinction.

Algorithms and Data Structures (CSci 115)

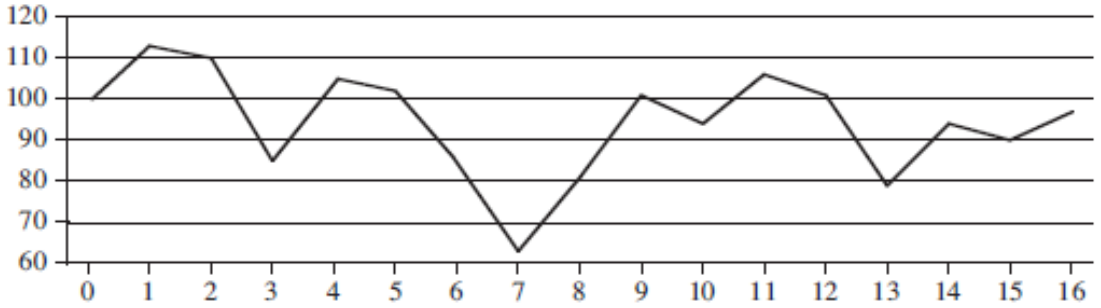
California State University Fresno
College of Science and Mathematics
Department of Computer Science
H. Cecotti

Learning outcomes

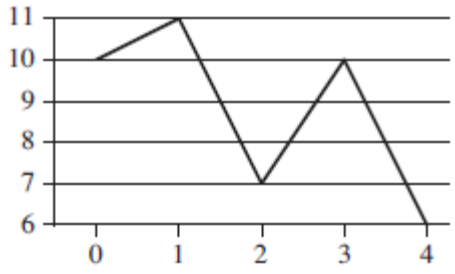
- Back to Divide & Conquer
  - Subarray problem
    - Solution + Analysis of the complexity
- Back to the array:
  - Class MyArray
    - Constructors + Destructors + Methods + Properties
- Multiple return values with C++
  - Tuple and pair

Find Max subarray

- Problem to solve
  - Buy stocks at time t1
  - Sell stocks at time t2



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



Day	0	1	2	3	4
Price	10	11	7	10	6
Change		1	−4	3	−4

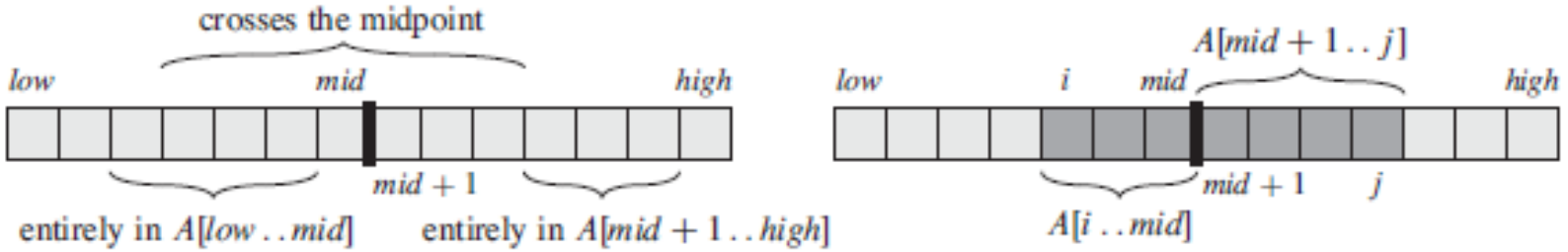
- From the sequence to the change (v(t)-v(t-1))
- 3 cases
  - In low-mid
  - Between low and high
  - In mid-high

Remark: maximum profit does not always start at the lowest price or end at the highest price

Find Max subarray

- Possible locations of the subarrays A[*low..high]*:
  - Entirely in A[*low*..*mid*]
  - Entirely in A[*mid+*1..*high]*
  - Crossing the midpoint *mid*.
- Any subarray of A[*low..high]* crossing the midpoint comprises 2 subarrays
  - A[i..*mid]*
  - A[*mid+1*..j]

    where *low* ≤i ≤ *mid* and *mid* < j ≤ *high*.



Find Max subarray

- Algorithms

```
FIND-MAX-CROSSING-SUBARRAY(A, low, mid, high)
1   left-sum = −∞
2   sum = 0
3   for i = mid downto low
4       sum = sum + A[i]
5       if sum > left-sum
6           left-sum = sum
7           max-left = i
8   right-sum = −∞
9   sum = 0
10  for j = mid + 1 to high
11      sum = sum + A[j]
12      if sum > right-sum
13          right-sum = sum
14          max-right = j
15  return (max-left, max-right, left-sum + right-sum)
```

low - mid

mid - high

```
FIND-MAXIMUM-SUBARRAY(A, low, high)
1   if high == low
2       return (low, high, A[low])          // base case: only one element
3   else mid = ⌊(low + high)/2⌋
4       (left-low, left-high, left-sum) =
            FIND-MAXIMUM-SUBARRAY(A, low, mid)
5       (right-low, right-high, right-sum) =
            FIND-MAXIMUM-SUBARRAY(A, mid + 1, high)
6       (cross-low, cross-high, cross-sum) =
            FIND-MAX-CROSSING-SUBARRAY(A, low, mid, high)
7       if left-sum ≥ right-sum and left-sum ≥ cross-sum
8           return (left-low, left-high, left-sum)
9       elseif right-sum ≥ left-sum and right-sum ≥ cross-sum
10          return (right-low, right-high, right-sum)
11      else return (cross-low, cross-high, cross-sum)
```

- Complexity
  - O(n log n)

Code

- Available on Canvas
  - Visual studio project
- Complete your own MyArray class
  - Make it as complete and useful as possible

Conclusion

- To do:
  - Class Array is fully functional
- Next sessions
  - Sort Array: Selection + Insertion + Bubble sort
  - Sort Array: Merge sort + Quick sort
- Questions?