ECS 122A: Algorithm Design and Analysis

Week 3 Discussion

© Ji Wang

Apr 14, 2021

A few words on logistics

- Homework 2 due tomorrow (Apr 15) at midnight
- Homework 3 is released, due next Tuesday (Apr 20) at midnight
- When submitting, please select page(s) to each question, preferably in PDFs
- Homework is graded by attempting (50%) + one selected problem (50%)
- Midterm 1 is next Thursday, will cover up to and including homework 3

Outline

- Divide and Conquer: Key idea
- Solve Divide and Conquer Recurrence: Master Theorem
- Variant of Maximum-Subarray Problem: Stock Investment

Divide and Conquer: Key idea

1. **Divide** the problem into a number of subproblems that are smaller instances of the same problem.
2. **Conquer** by solving the subproblems recursively.
3. **Combine** the solutions to the subproblems to produce the solution to the original problem.

Divide and Conquer: Key idea

1. **Divide** the problem into a number of subproblems that are smaller instances of the same problem.
2. **Conquer** by solving the subproblems recursively.
3. **Combine** the solutions to the subproblems to produce the solution to the original problem.

$$T(n) = aT(\frac{n}{b}) + f(n)$$

So far, we demonstrated three problems that use divide and conquer paradigm:

- Merge Sort
- Maximum Subarray
- Matrix-matrix multiply (Strassen's algorithm)

Using Master Theorem

What is the asymptotic bound for the given recurrence?

$$T(n) = 3T(\frac{n}{4}) + n$$

$a =$
$b =$
$f(n) =$

Using Master Theorem

What is the asymptotic bound for the given recurrence? Does Master Theorem apply to this recurrence?

$$T(n) = 2T(\frac{n}{2}) + n \lg n$$

Stock Investment

Problem Statement:

We're doing a simulation in which we look at n consecutive days of a given stock, at some point in the past. Let's number the days $i = 1, 2, ..., n$ and $p(i)$ is the price per share for the stock on that day. We want to know: When should we have bought and sold in order to have made as much money as possible? (If there was no way to make money during the n days, we should report this instead.)

For example, $n = 4$, $p(1) = 9$, $p(2) = 1$, $p(3) = 5, p(4) = 3$. Then we should answer "buy on day 2, sell on day 3".

Stock Investment

Problem Statement:

We're doing a simulation in which we look at n consecutive days of a given stock, at some point in the past. Let's number the days $i = 1, 2, ..., n$ and $p(i)$ is the price per share for the stock on that day. We want to know: When should we have bought and sold in order to have made as much money as possible? (If there was no way to make money during the n days, we should report this instead.)

For example, $n = 4$, $p(1) = 9$, $p(2) = 1$, $p(3) = 5, p(4) = 3$. Then we should answer "buy on day 2, sell on day 3".

Rephrase the problem:

Input: array p of length n

Stock Investment

Problem Statement:

We're doing a simulation in which we look at n consecutive days of a given stock, at some point in the past. Let's number the days $i = 1, 2, ..., n$ and $p(i)$ is the price per share for the stock on that day. We want to know: When should we have bought and sold in order to have made as much money as possible? (If there was no way to make money during the n days, we should report this instead.)

For example, $n = 4$, $p(1) = 9$, $p(2) = 1$, $p(3) = 5, p(4) = 3$. Then we should answer "buy on day 2, sell on day 3".

Rephrase the problem:

Input: array p of length n

Output: $\texttt{argmax}\{p(j) - p(i)\}$ where $i \leq j$

Stock Investment

Revisit maximum subarray problem:

1. Divide $A[low \cdots high]$ into two subarrays of as equal size as possible by finding the midpoint mid
2. Conquer:
   a. finding maximum subarrays of $A[low \cdots mid]$ and $A[mid + 1 \cdots high]$
   b. finding a max-subarray that crosses the midpoint
3. Combine: returning the max of the three

Stock Investment

Revisit maximum subarray problem:

1. Divide $A[low \cdots high]$ into two subarrays of as equal size as possible by finding the midpoint mid
2. Conquer:
   a. finding maximum subarrays of $A[low \cdots mid]$ and $A[mid + 1 \cdots high]$
   b. finding a max-subarray that crosses the midpoint
3. Combine: returning the max of the three

Can we apply the same strategy on this problem?

Stock Investment

How does the conquer part work?

1. The optimal solution to $A[low \cdots mid]$
2. The optimal solution to $A[mid + 1 \cdots high]$
3. $\texttt{argmax}\{p(j) - p(i)\}$ where $low \leq i \leq mid$ and $mid + 1 \leq j \leq high$

Stock Investment

How does the conquer part work?

1. The optimal solution to $A[low \cdots mid]$
2. The optimal solution to $A[mid + 1 \cdots high]$
3. $\texttt{argmax}\{p(j) - p(i)\}$ where $low \leq i \leq mid$ and $mid + 1 \leq j \leq high$ (equivalent to finding the index of `min` of $A[low \cdots mid]$ and that of `max` of $A[mid + 1 \cdots high]$)

Stock Investment

How does the conquer part work?

1. The optimal solution to $A[low \cdots mid]$
2. The optimal solution to $A[mid + 1 \cdots high]$
3. $\texttt{argmax}\{p(j) - p(i)\}$ where $low \le i \le mid$ and $mid + 1 \le j \le high$ (equivalent to finding the index of `min` of $A[low \cdots mid]$ and that of `max` of $A[mid + 1 \cdots high]$)

How to describe the design of an algorithm in English

- First point out what strategy/method used, e.g. divide-and-conquer, binary search.
- Separate paragraphs if necessary, e.g. branches.
- Bullet-point format is also a good practice.

Stock Investment: Pseudocode [1]

```
STOCK-INVESTMENT(A, low, high)
// Base case: only one element
if low == high
    return low, high, 0
else mid = low + ⌊(high − low)/2⌋
    leftBuy, leftSell, leftGain = STOCK-INVESTMENT(A, low, mid)
    rightBuy, rightSell, rightGain = STOCK-INVESTMENT(A, mid, high)
    // Find the index of min(leftArray)
    crossBuy = MIN-INDEX(A, LOW, MID)
    // Find the index of max(rightArray)
    crossSell = MAX-INDEX(A, MID, HIGH)
    crossGain = A[crossSell] − A[crossBuy]
    if max(leftGain, rightGain, crossGain) < 0
        return "no gain"
    elseif leftGain ≥ rightGain and leftGain ≥ crossGain
        return leftBuy, leftSell, leftGain
    elseif rightGain ≥ leftGain and rightGain ≥ crossGain
        return rightBuy, rightSell, rightGain
    else return crossBuy, crossSell, crossGain
```

[1] See pp.[20-22] in the textbook for pseudocode conventions

Stock Investment: Time complexity

$$
\begin{aligned}
T(n) &= 2T(\frac{n}{2}) + \Theta(n) + \Theta(1) \\
&= \Theta(n \log n)
\end{aligned}
$$

1. $2T(\frac{n}{2})$: the first two optimal solutions
2. $\Theta(n)$: the third solution is equivalent to finding the `min` and `max`, e.g. linear scan
3. $\Theta(1)$: compare the results of three solutions