Week 11: Review

CSC 236:Introduction to the Theory of Computation

Summer 2024

Instructor: Lily

Announcements

IDK RULE: If Q1 (a) [2marks] ~~~

(b) [6 marks] IDK ✓

(b) [6 marks] ~~~

+ IDK ✗

- Course evaluations are available now! *(open until August 14th)*
  - We ***highly encourage*** everyone to fill it out
  - Once you filled out the evaluation, take quiz 11 and select “true” (honor system, please don’t lie)
- Office hours will continue until the 16th of August
  - Tuesday office hours are now 5~6:00pm ***online***
- For those who missed the final
  - ***There are NO make-up exams or alternative assessments***
  - Petition to write a deferred in-person final exam with the Fall offering of the course on the Arts&Sci website
- *There is no tutorial after this class*

Iterative algorithm

Given a *zero-indexed* list P containing the price of a single stock over n days we want to ***find the maximum profit when making a single buy and a single sell trade***.

$P[i]$ is the price of the stock on day i for $i \in \{0, \dots, n-1\}$. If you buy on day i and sell on day j, then *your profit would be* $P[j] - P[i]$. You can only sell *after* you buy.

The Algorithm should compute $\max\limits_{0 \le i < j < n} P[j] - P[i]$

$P = [10, 2, 9, 3, 4, 1, 6]$

buy buy max NOT sell

OPT $9-2=7$ profit 0

sell not buy or sell

profit 0

sell

OPT BUY

```
def BestBuy(P: array):
    min_price = P[0]
    max_profit = 0
    for i in range(1, len(P)):
        max_profit = max(max_profit,
                         P[i]-min_price)
        if P[i] < min_price:
            min_price = P[i]
    return max_profit
```

```
def BestBuy(P: array):
    min_price = P[0]
    max_profit = 0
    for i in range(1, len(P)):
        max_profit = max(max_profit, P[i]-min_price)
        if P[i] < min_price:
            min_price = P[i]
    return max_profit
```

loop of interest

precondition: P is a non-empty array

postcondition: max_profit returned will be max value obtained by buying at time i, selling at time j with i<j.

variable:

min_price_i : val of min-price after iter i

max_profit_i : value of max_profit after iter i

LI(i) := after iteration i, for interval I = P[0 : i+1], min_price_i is the smallest value of I and max_profit_i is the max value obtained if we buy and sell in I

LI(i) := after iteration i, for interval I = P[0 : i+1], min_price$_i$ is the smallest value of I and max_profit$_i$ is the max value obtained if we buy and sell in I

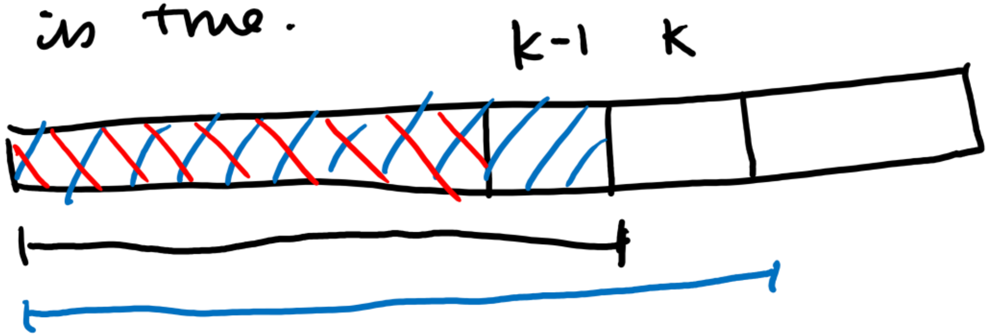
Q(i) := if $i \in \{0, \dots, n-1\}$ then LI(i) is true.

Base case: i=0, interval P[0 : 1] = P[0] min_price$_0$ = P[0] only entry

max_profit$_0$ = 0 since only looked at one value

Inductive STEP: for $k \in \{1, \dots, n-1\}$ suppose $Q(0) \wedge \dots \wedge Q(k-1)$ True.

show Q(k) is true.



iteration
k-1
k

min_price$_k$ = min(P[k], $\boxed{\text{min\_price}_{k-1}}$)

by line 5-6 in code

(IH) min of ▨

max_profit$_k$

= max($\boxed{\text{max\_profit}_{k-1}}$, P[k] - min_price$_{k-1}$)

(IH)

Recurrence Relation

M2 Q3. Compute a tight *upper bound* for the closed form of the following recurrence relation:
$T(n) = c$ for $n \leq 10$ for constant c and otherwise

$$T(n) = T\left(\left\lfloor\frac{n}{2}\right\rfloor + 1\right) + cn$$

exercise question: change 1 into something else does it make a difference?

Guess: $T(n) = O(n)$

$P(n): \exists c' : \boxed{T(n) \leq c'n}$. (choose $c' > c$)

Base case: $1 \leq n \leq 10$ $\quad T(n) = c \leq c'n$

Inductive Step: fix $k \geq 9$ and let $\boxed{P(1) \wedge \cdots \wedge P(k-1)}$ be true

show $P(k)$ is true.

$$\begin{aligned}
T(k) &= T(\lfloor k/2 \rfloor + 1) + k & \text{(def)} \\
&\leq c'(\lfloor k/2 \rfloor + 1) + k & \text{(IH)} \\
&\leq c'(k/2 + 1) + k = c'k - c'k/2 + c' + k \\
&\leq c'k
\end{aligned}$$

≤ 0 choose $c' \geq 10$

want $-\frac{c'k}{2}+c'+k<0$

$$k<c'\left(\frac{k}{2}+1\right)$$

$$\frac{k}{\frac{k}{2}+1}<c'$$

Recurrence Relation

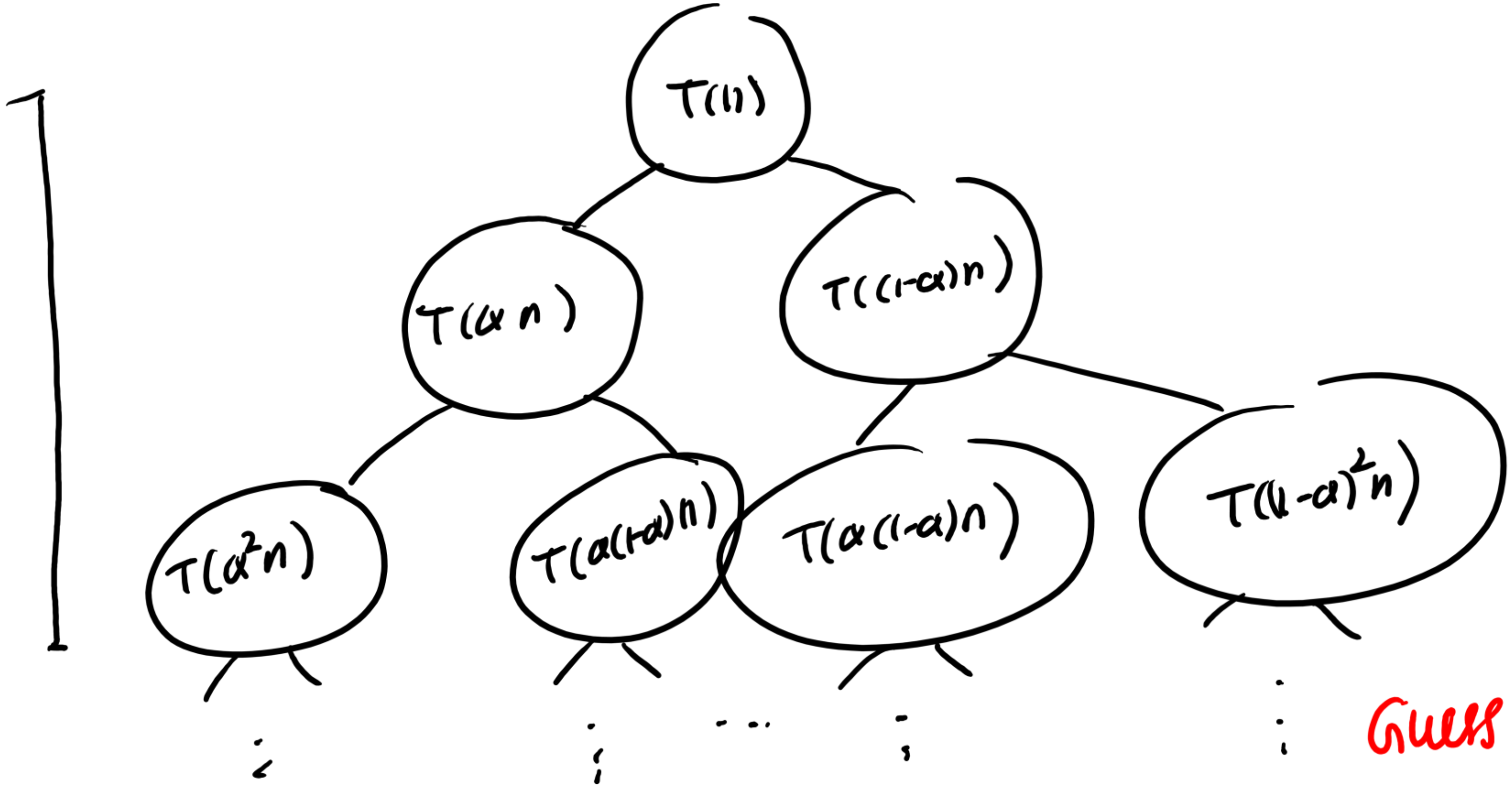
A5 Q1 (a) Find and prove the closed form expression for the recurrence relation

$$T(n) = T(\alpha n) + T\big((1-\alpha)n\big) + cn$$

WLOG $\alpha \in [0, \frac{1}{2})$

height h.

$\log_{\frac{1}{1-\alpha}} n \le h \le \log_{\frac{1}{\alpha}} n$



WORK DONE

cn

$c\alpha n + c(1-\alpha)n$
$= cn$

$c\alpha^2 n + 2c\alpha(1-\alpha)n$
$+ c(1-\alpha)^2 n$
$= cn.$

Guess: $T(n) = \Theta(n \log n)$

$\boxed{\text{logs base } \frac{1}{1-\alpha}}$

$P(n): \exists c_2 : T(n) \leq c_2 n \log n$

Base case: $T(0), \ldots, T(\lceil 1/1-\alpha \rceil)$ are all constants, can choose $c_2 \geq \max(T(0), \ldots, T(\lceil 1/1-\alpha \rceil))$.

Inductive step: $T(k) = T(\alpha k) + T((1-\alpha)k) + ck$ (def)

$$\leq c_2 k \alpha \log \alpha k + c_2 k (1-\alpha) \log (1-\alpha) k + ck \quad (IH)$$

$$= c_2 k \left(\log (\alpha k)^{\alpha} + \log ((1-\alpha)k)^{(1-\alpha)} \right) + ck$$

$$= c_2 k \left(\log k - \log \left(\tfrac{1}{\alpha}\right)^{\alpha} \left(\tfrac{1}{1-\alpha}\right)^{1-\alpha} \right) + ck$$

$$= c_2 k \log k - c_2 k \log \left(\tfrac{1}{\alpha}\right)^{\alpha} \left(\tfrac{1}{1-\alpha}\right)^{1-\alpha} + ck$$

$$\leq c_2 k \log k \quad \text{if } c_2 \geq c \Big/ \log \left(\tfrac{1}{\alpha}\right)^{\alpha} \left(\tfrac{1}{1-\alpha}\right)^{1-\alpha}$$

Recursive Algorithm – Divide and Conquer

Given a *zero-indexed* array P of length n, find the maximum sum of any subsequence of P.

$$P = [\ 1 \quad 2 \quad -9 \quad 3 \ \vdots \ 3 \quad 1 \quad -2 \quad 1\ ]$$

-4   4

3   3   4   sum 0

suffix   prefix

P_L 7 P_R OUTPUT this

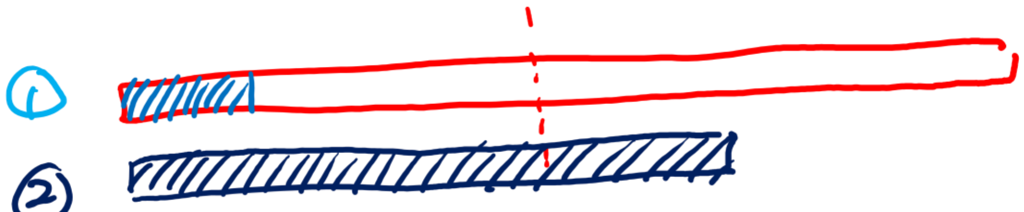
3 possible max sum subsequences in P

① max sum subseq. on P_L

② max sum subseq. on P_R

③ max sum subseq crosses the middle.

- max sum suffix of P_L + max sum prefix of P_R



```
def max_subseq(A : array)
    # Base case
    if len(A) = 1, then OPT = max{0, A[0]}
        return OPT, OPT, OPT, A[0]
    m = len(A)/2
    # divide
    Lmax, Lpref, Lsuf, Lsum = max_subseq(A[:m])
    Rmax, Rpref, Rsuf, Rsum = max_subseq(A[m:])
    # conquer
    umax = max(Lmax, Rmax, Lsuf + Rpref)
    upref = max(Lpref, Lsum + Rpref)
    usuf = max(Rsuf, Rsum + Lsuf)
    usum = Lsum + Rsum
    return umax, upref, usuf, usum.
```

PROOF of correctness of recursive algol. (structural induction)

recursive set S : $A[i] \in S$ all entries in A.

for arrays $A_1, A_2 \in S$ of the same length, $A_1 A_2 \in S$

PRECOND : A non-empty

POST COND : values satisfy definition

umax : val max sum subseq in A

upref : . . . . prefix subseq of A

usuf : . . . suffix subseq of A

usum : sum of all entries in A.

$P(A)$: for $A \in S$, max_subseq(A) satisfies post conditions.

inductive step. $A_L, A_R \in S$ of same length

consider max_subseq($A_L A_R$)

SUPPOSE $P(A_1) \wedge P(A_2)$ are true show $P(A_1 A_2)$

consider umax returned.

By IH, Lmax, Rmax, Lsuf, Rpref are what they say they are. so all 3 ways of making max sum subsequence covered.

show uprefi, usuf, usum as well.

Divide and Conquer Example

Compute x^n for integer x and natural number n.

What we learned this semester

- Combinatorics
  - Permutations, combinations, stars and bars
  - Binomial Coefficient, Fibonacci numbers
  - Pigeonhole Principle
  - Graph theory: trees, cycles, paths, etc.
- Proof of correctness
  - Iterative algorithm: simple multiplication algorithms, Prim’s algorithm, etc.
  - Recursive algorithm: Karatsuba’s algorithm, quicksort, divide-and-conquer, etc.
- Finite Automaton
  - **Languages are just sets of strings**
  - Regular languages
  - Definition of DFA, NFA, and regex
  - Proof of equivalence of the three
  - Limitations of regular languages: **Regular languages cannot count**