Welcome to COMP 3711H!

Honors Design and Analysis of Algorithms

Instructor: Amir Goharshady goharshady@cse.ust.hk

TAs: Chun Kit Lam and Kerim Kochekov {cklamaq, kkochekov}@connect.ust.hk

Guest Lectures by Dr. Harshit Motwani hjmotwani@connect.ust.hk

Grading:

Homeworks: 4 x 10% -- not only designing algorithms but also coding

Midterm Exam: 30% -- pen and paper

Final Exam: 30% -- pen and paper

Required Coding Practice: 0% -- will fail the course if not done

This course expects a lot of self-study and coding practice.

Ample resources will be put on Canvas.

We use C++ for coding in classes/tutorials and also expect you to write C++ code in homeworks.

It is NOT allowed to use ChatGPT or other AI tools in exams and homeworks.

We will also make sure they would be useless.

Runtime Analysis of algorithm

Given a set A of atomic ops and an algorithm $\mathcal{A}$

$T(\mathcal{A}, I)$ = runtime of $\mathcal{A}$ on I = # atomic ops

$|I| = n$

WCET

$$T(\mathcal{A}, n) = \max_{|I|=n} T(\mathcal{A}, I)$$

Asymptotic Analysis

Two algorithms with runtimes $f(n)$ and $g(n)$.

$\cancel{5}n + \cancel{5}$ $\quad\quad$ $\cancel{100}n$

Formal Def: $f \in O(g)$ if

$$\exists n_0 \quad \exists c > 0 \quad \forall n > n_0 \quad f(n) \leq c \cdot g(n)$$

Def 2. $f \in \Omega(g)$ if

$$\exists n_0 \quad \exists c > 0 \quad \forall n > n_0 \quad c \cdot f(n) \geq g(n)$$

Def 3. $f \in \Theta(g) \iff f \in O(g) \wedge g \in O(f)$

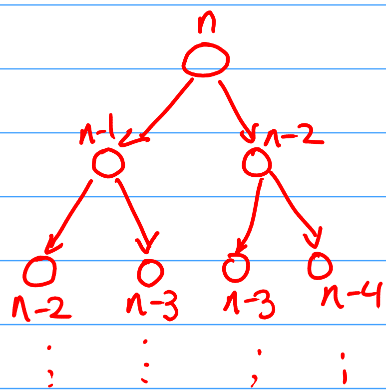
Def 4. $f \in o(g) \iff f \in O(g) \wedge g \notin O(f)$

$f \in \omega(g) \iff f \in \Omega(g) \wedge g \notin \Omega(f)$

Fibonacci numbers

$$f_0 = 1 \qquad f_1 = 1 \qquad f_i = f_{i-1} + f_{i-2} \quad i \geq 2$$

```
function fib(n):
    if n=0 or n=1:
        return 1
    return fib(n-1) + fib(n-2)
```



$$T(n) = \begin{cases} O(1) & n=0 \text{ or } n=1 \\ T(n-1) + T(n-2) + O(1) & \text{o.w.} \end{cases}$$

$$T(n) > f_n$$

A[n+1]

A[0] = A[1] = 1

for $2 \leq i \leq n$:
    A[i] = A[i-1] + A[i-2]

Runtime $\in O(n)$

Maximum Contiguous Subsequence

Input: An array A of integers

10 5 −2 −3 10 12 0

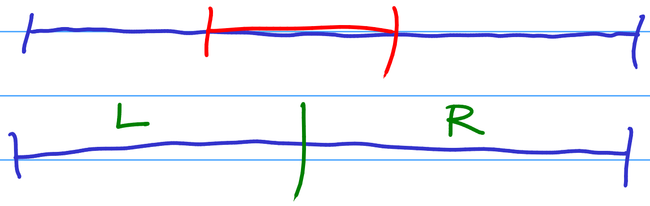
Output: Find $i \leq j$ such that $\sum_{k=i}^{j} A[k]$ is maximized.

$\parallel$

$B[j] - B[i-1]$

Naïve algorithm $O(n^3)$

First improvement $O(n^2)$



Case 1: answer entirely in L

Case 2: answer entirely in R

Case 3: Intersects both

mcs (A, start, end):

Base cases

ans = mcs (A, start, $\underbrace{\frac{start+end}{2}}$) // mid

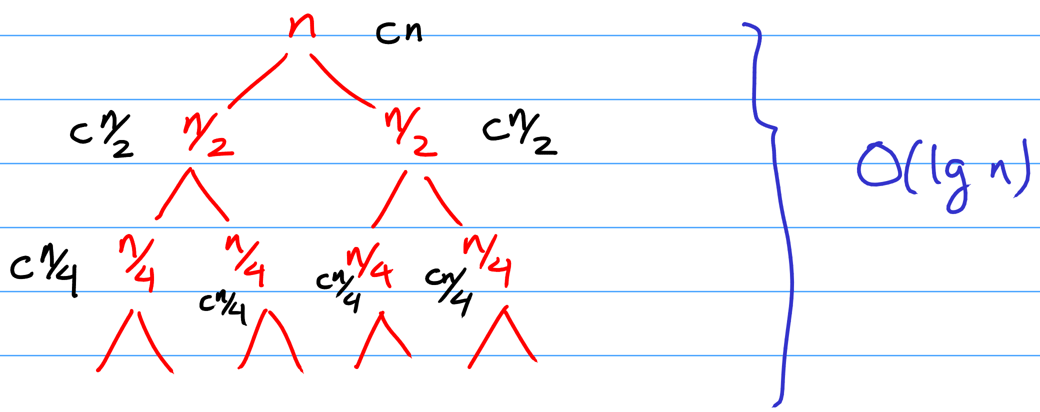
ans = max (ans, mcs (A, $\frac{start+end}{2}+1$, end))

sumL=0, sumR=0

bestL=0, bestR=0

```
for i = mid downto start
    sumL += A[i]
    bestL = max(bestL, sumL)
        ⋮
if bestL + bestR > ans:
    ans = bestL + bestR
```

$T(n) = 2T(n/2) + O(n)$ $\qquad$ $T(n) \le 2T(n/2) + cn$



Total runtime $\in O(n \lg n)$

$B[i] = \sum_{k=1}^{i} A[k]$

Best runtime $\in O(n)$