**Soham Chatterjee**
Email: soham.chatterjee@tifr.res.in
Course: Algorithms

**Assignment - 1**
Dept: STCS
Date: September 9, 2024

**Problem 1** P3 (15 marks)

Solve the recurrences:

(i) $T(n) = 2T(n/2) + n\log n$,

(ii) $T(n) = 7T(n/3) + n^2$,

(iii) $T(n) = \sqrt{n}T(\sqrt{n}) + n$.

***Solution:***

(i) We have the recurrence relation $T(n) = 2T\left(\frac{n}{2}\right) + n\log n$. So

$$\begin{aligned}
T(n) &= 2T\left(\frac{n}{2}\right) + n\log n \\
&= 4T\left(\frac{n}{4}\right) + \frac{n}{2}\log\frac{n}{2} + n\log n \leq 2^2 T\left(\frac{n}{2^2}\right) + 2n\log n \\
&= 2^3 T\left(\frac{n}{2^3}\right) + \frac{n}{2^2}\log\frac{n}{2^2} + 2n\log n \leq 2^3 T\left(\frac{n}{2^3}\right) + 3n\log n \\
&\cdots \\
&= 2^k T\left(\frac{n}{2^k}\right) + \frac{n}{2^k}\log\frac{n}{2^k} + (k-1)n\log n \leq 2^k T\left(\frac{n}{2^k}\right) + kn\log n \\
&\cdots \\
&\leq 2^{\log n} T(1) + \log n(n\log n) \leq nT(n) + n\log^2 n = n(T(1) + \log^2 n) = O(n\log^2 n)
\end{aligned}$$

So we claim $T(n) \leq cn(T(1)+\log^2 n)$ for all $n \geq n_0$ for some $c$ which we will choose accordingly. Now $n_0 = 2$. So for $n = 2$ we have $T(2) = 2T(1) + 2\log 2 = 2T(1) + 2 = 2(T(1) + 1) \leq c2(T(1) + \log^2 2)$. Hence the base case follows. Now let $T(n) = cn\log^2 n$ is true for all $n = 2, \ldots, k-1$. Now

$$T(k) = 2T\left(\frac{k}{2}\right) + k\log k \leq 2c\frac{k}{2}\left(T(1) + \log^2\frac{k}{2}\right) + k\log k = ck\left(T(1) + \log^2\frac{k}{2}\right) + k\log k$$

Now $\log^2\frac{k}{2} = (\log k - 1)^2 = \log^2 k - 2\log k + 1$. So we have

$$ck\left(T(1) + \log^2\frac{k}{2}\right) + k\log k = ck(T(1)+\log^2 k) - 2ck\log k + ck + k\log k = ck(T(1)+\log^2 k) + (1-2c)k\log k + ck$$

If $c \geq 1$ we have $1 - 2c \leq -1$. So we have

$$(1-2c)k\log k + ck \leq ck - k\log k \leq 0$$

Here the last inequality follows if $c \leq \log k$. Since $k \geq 2$ we have $\log k \geq 1$. So take $c = 1$. Then $(1 - 2c)k\log k + ck \leq 0$. Therefore

$$T(k) = k(T(1) + \log^2 k) + (1-2)k\log k + k \leq k(T(1) + \log^2 k)$$

Hence by mathematical induction we have for all $n \geq 2$, $n \in \mathbb{N}$ we have $T(n) \leq n(T(1) + \log^2 n)$. Now

$$n(T(1) + \log^2 n) = n(T(1)\log^2 n + \log^2 n) = (1 + T(1))n\log^2 n = O(n\log^2 n)$$

Hence we have $T(n) = O(n\log^2 n)$.

(ii) We have the recurrence relation $T(n) = 7T\left(\frac{n}{3}\right) + n^2$. So

$$\begin{aligned}
T(n) &= 7T\left(\frac{n}{3}\right) + n^2 \\
&= 7^2 T\left(\frac{n}{3^2}\right) + \frac{n^2}{9} + n^2 \\
&= 7^3 T\left(\frac{n}{3^3}\right) + \frac{n^2}{3^4} + \frac{n^2}{3^2} + n^2 = 7^3 T\left(\frac{n}{3^3}\right) + n^2 \sum_{i=1}^{3} \frac{1}{3^{2i}} \\
&\cdots \\
&= 7^n T\left(\frac{n}{3^k}\right) + n^2 \sum_{i=1}^{k} \frac{1}{9^i} \\
&\cdots \\
&= 7^{\log_3 n} T(1) + n^2 \sum_{i=1}^{\log_3 n} \frac{1}{9^i} \leq n^{\log 7_3} T(1) + \frac{9}{8} n^2 \leq T(1) n^2 + \frac{9}{8} n^2 = \left(T(1) + \frac{9}{8}\right) n^2
\end{aligned}$$

So we claim $T(n) = (T(1) + c)n^2$ for some $c \geq 2$ and $n \geq n_0$ where $n_0 \in \mathbb{N}$. So take $n_0 = 3$. Then $T(3) = 7T(1) + 9 \leq 9T(1) + 18 \times 9 = (T(1) + c)9$. Hence this follows for the base case. Now suppose $T(n) = (T(1) + c)n^2$ for all $n = 3, \ldots, k-1$. Then for $n = k$

$$T(k) = 7T\left(\frac{k}{3}\right) + k^2 \leq 7(T(1) + c)\frac{k^2}{3^2} + k^2 = k^2\left(\frac{7(T(1) + c)}{9} + 1\right)$$

We want

$$\frac{7(T(1) + c)}{9} + 1 \leq T(1) + c \iff 7(T(1) + c) + 1 \leq 9(T(1) + c) \iff 1 \leq 2(c + T(1))$$

this is indeed true since $c \geq 2$. Hence we have $T(k) \leq (c + T(1))k^2$. Hence by mathematical induction we have $T(n) \leq (c + T(1))n^2$ for all $n \geq 4$ with $n \in \mathbb{N}$. Now $(c + T(1))n^2 = O(n^2)$. Hence $T(n) = O(n^2)$.

(iii) We have the recurrence relation

$$T(n) = \sqrt{n} T(\sqrt{n}) + n \iff \frac{T(n)}{n} = \frac{T(\sqrt{n})}{\sqrt{n}} + 1$$

Now denote $F(n) = \frac{T(n)}{n}$. Then we have the new recurrence relation

$$f(n) = f(\sqrt{n}) + 1$$

Now suppose $n = 2^{2^k}$. Then

$$\begin{aligned}
f\left(2^{2^k}\right) &= f\left(\sqrt{2^{2^k}}\right) + 1 = f\left(2^{2^{k-1}}\right) + 1 \\
&= f\left(2^{2^{k-2}}\right) + 2 \\
&\cdots \\
&= f\left(2^{2^0}\right) + k \\
&= f(2) + k
\end{aligned}$$

Now $f(2) = \frac{T(2)}{2}$ which is a constant. So there exists $n_0 \in \mathbb{N}$ such that $f(2) \leq \log\log n$ for all $n \geq n_0$. So for large $k$ we have

$$f\left(2^{2^k}\right) = f(2) + k \leq 2k$$

Hence we claim $f(n) = O(\log_2 \log_2 n)$. For $n = n_0$ we already have $f(n_0) \leq 2\log_2 \log_2 n$. So let for $n = n_0, \ldots, t-1$ we have $f(n) \leq c \log_2 \log_2 n$ for some $c \in \mathbb{N}$. Certainly seeing the $n = n_0$ we have $c \geq 2$ but we will choose $c$ appropriately later. Now for $n = t$

$$\begin{aligned} f(t) &= f(\sqrt{t}) + 1 \\ &\leq c \log_2 \log_2 (\sqrt{t}) + 1 \\ &= c \log_2 \left( \frac{1}{2} \log_2 t \right) + 1 \\ &= c \log_2 \frac{1}{2} + c \log_2 \log_2 t + 1 \\ &= c \log_2 \log_2 t - c + 1 \leq 2 \log_2 \log_2 t \end{aligned}$$

So if we choose $c = 2$ then we are done. Hence by mathematical induction $f(n) = O(\log_2 \log_2 n)$ for all $n \geq n_0$. Now we have $f(n) = \frac{T(n)}{n}$ and $f(n) = O(\log_2 \log_2 n)$. Hence we have

$$T(n) = O(n \log_2 \log_2 n)$$

■

**Problem 2** P4 (5 marks)

Give the best upper bounds you can on the $n$th Fibonacci number $F_n$, where $F_n = F_{n-1} + F_{n-2}$ and $F_1 = F_2 = 1$

***Solution:*** We have the recurrence relation $F(n) = F_{n-1} + F_{n-2}$ with $F_1 = F_2 = 1$. So we can represent this with matrices like following:

$$\begin{bmatrix} F_n \\ F_{n-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} F_{n-1} \\ F_{n-2} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^2 \begin{bmatrix} F_{n-2} \\ F_{n-3} \end{bmatrix} = \cdots \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^{n-2} \begin{bmatrix} F_2 \\ F_1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^{n-2} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^{n-1} \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

Denote $\overline{F}_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and $\overline{F}_k = \begin{bmatrix} F_{k+1} \\ F_k \end{bmatrix}$ and $A = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$. Therefore we have $\overline{F}_n = A^n \overline{F}_0$.

Now clearly $A$ has full rank and $\forall\ k \in \mathbb{N}$, $\overline{F}_k \in \mathbb{R}^2$. So we will find the eigenvalues of $A$ to find an eigenbasis.

$$\det(A - tI) = \det \begin{bmatrix} 1-t & 1 \\ 1 & -t \end{bmatrix} = -t(1-t) - 1 = t^2 - t - 1$$

So if $t^2 - t - 1 = 0$ then

$$t = \frac{1 \pm \sqrt{1+4}}{2} = \frac{1 \pm \sqrt{5}}{2}$$

So denote $\varphi = \frac{1+\sqrt{5}}{2}$ and $\psi = \frac{1-\sqrt{5}}{2}$. Now let $X = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ be an eigenvector corresponding to $\varphi$. Then

$$AX = \begin{bmatrix} x_1 + x_2 \\ x_1 \end{bmatrix} = \varphi \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Therefore $x_1 = \varphi x_2$. Therefore take $x_2 = 1$ then we have $x_1 = \varphi$. So $X = \begin{bmatrix} \varphi \\ 1 \end{bmatrix}$. Similarly we have $Y = \begin{bmatrix} \psi \\ 1 \end{bmatrix}$ is an eigenvector of $A$ corresponding to $\psi$.

Now we want to express $\overline{F}_0$ as a linear combination of $X, Y$. Notice

$$\frac{1}{\sqrt{5}}(X - Y) = \frac{1}{\sqrt{5}} \begin{bmatrix} \varphi - \psi \\ 0 \end{bmatrix} = \frac{1}{\sqrt{5}} \begin{bmatrix} \frac{1+\sqrt{5}}{2} - \frac{1-\sqrt{5}}{2} \\ 0 \end{bmatrix} = \frac{1}{\sqrt{5}} \begin{bmatrix} \sqrt{5} \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \overline{F}_0$$

Therefore

$$\overline{F}_n = A^n \overline{F}_0 = A^n \left( \frac{1}{\sqrt{5}}(X - Y) \right) = \frac{1}{\sqrt{5}}(AX - AY) = \frac{1}{\sqrt{5}}(\varphi^n X - \psi^n Y) = \frac{1}{\sqrt{5}} \varphi^n \begin{bmatrix} \varphi \\ 1 \end{bmatrix} - \frac{1}{\sqrt{5}} \psi^n \begin{bmatrix} \psi \\ 1 \end{bmatrix}$$

Therefore $F_n = \frac{\phi^n - \psi^n}{\sqrt{5}}$.

[I knew about How to solve Linear Recurrences using Matrices] ■

**Problem 3** P5 (10 marks)

Consider two sets $A$ and $B$, each having $n$ integers in the range from 0 to $10n$. We wish to compute the Cartesian sum of $A$ and $B$, defined by

$$C = \{x + y : x \in A, y \in B\}$$

Note that the integers in $C$ are in the range 0 to $20n$ . We want to find the elements in $C$ and the number of times each element of $C$ is realized as a sum of elements in $A$ and $B$. Give an algorithm that solves the problem in $O(n \log n)$ time, and prove correctness.

***Solution:*** Given $A$, $B$ we create two polynomials $p_A(x) = \sum_{k \in A} x^k$ and $p_B(x) = \sum_{k \in B} x^k$. Since all entries of $A$ and $B$ ranges form 0 to $10n$. We have $\deg p_A \leq 10n$ and $\deg p_B \leq 10n$. Hence now we can use the algorithm for polynomial multiplication to calculate $p = p_A \cdot p_B$. Now $\deg p \leq 20n$. For any term $x^k$ in $p$, $\exists\ a \in A$ and $b \in B$ such that $a + b = c$ since $p$ is the product of $p_A$ and $p_B$. Let $S_k := \{(a, b) \in A \times B \text{ such that } a + b = k\}$. Then the coefficient of $x^k$ in $p$ is $|S_k|$ since

$$\text{Coeff}(x^k) = \sum_{i=0}^{k} \text{Coeff}_A(x^i) \cdot \text{Coeff}_B(x^{k-i})$$

where $\text{Coeff}_A(x^i)$ is the coefficient of $x^k$ in $p_A$ and $\text{Coeff}_B(x^{k-i})$ is the coefficient of $x^{k-i}$ in $p_B$.

So now we will describe the algorithm. We denote the polynomial multiplication algorithm of two polynomials $S, T$ by Polynomial-Multiplication$(S, T)$. So the algorithm will be: ■

---

**Input:** $A = \{a_i \mid i \in [n],\ a_i \in \mathbb{Z},\ 0 \leq a_i \leq 10n\}$, $B = \{b_i \mid i \in [n],\ b_i \in \mathbb{Z},\ 0 \leq b_i \leq 10n\}$

**Output:** $C = \left\{(c, k_c) : \exists\ a \in A,\ b \in B \text{ st } c = a + b,\ k_c = \left|\{(a, b) \in A \times B \mid a + b = c\}\right|\right\}$

```
1 begin
2     Create two arrays S_A and S_B of length 10n + 1 with all elements 0
3     for i = 1, ..., n do
4         S_A[A[i]] ⟵ 1
5         S_B[B[i]] ⟵ 1
6     S ⟵ Polynomial-Multiplication(S_A, S_B)
```

---

**Problem 4** P6 (20 marks)

Define $[n] := \{1, 2, \ldots, n\}$. You are given $n$, and oracle access to a function $f : [n] \times [n] \to [n] \times [n]$ that takes as input two positive integers of value at most $n$, and returns two positive integers of value at most $n$. Let $f_1(x_1, x_2)$ and $f_2(x_1, x_2)$ be the first and second coordinates of $f(x_1, x_2)$, respectively. You are also told that $f_i$ is monotone nondecreasing in coordinate $i$ when coordinate $3 - i$ is kept fixed, and monotone nonincreasing in coordinate $3 - i$ when coordinate $i$ is kept fixed. That is, given $x_1 \leq x_1' \in [n]$ and $x_2 \leq x_2' \in [n]$, $f_1(x_1, x_2) \leq f_1\left(x_1', x_2\right)$, and $f_1(x_1, x_2) \geq f_1\left(x_1, x_2'\right)$. Similarly, $f_2(x_1, x_2) \geq f_2\left(x_1', x_2\right)$, and $f_2(x_1, x_2) \leq f_2\left(x_1, x_2'\right)$.

The problem is to find a fixed point of the function, i.e., values $x_1, x_2 \in [n]$ so that $f(x_1, x_2) = (x_1, x_2)$. Give an algorithm that given $n$ and oracle access to such a function $f$, finds a fixed point of $f$ in time $O(\text{poly}(\log n))$. You must also give a proof of correctness, and running time analysis.

***Solution:***

■

**Problem 5** P7 (15 marks)

A palindrome is a nonempty string over some alphabet that reads the same forward and backward. Examples of palindromes are all strings of length 1, civic, racecar, and aibohphobia. Give an efficient algorithm, with

proof of correctness and run-time analysis, to find the longest palindrome that is a subsequence of a given input string. For example, given the input string character, your algorithm should return carac.

***Solution:***

■

**Problem 6** P8 (25 marks)

The purpose of this question is to extend the closest-points algorithm seen in the first lecture, to give an $O\left(n \log^2 n\right)$ algorithm for finding the closest pair of points in 3 dimensions. All points in this question are in $\mathbb{R}^3$.

(a) (5 marks) Prove that, if all points are at least distance $\delta$ apart, a cube with each dimension of size $2\delta$ contains at most a constant (say $k$ ) number of points.

(b) (10 marks) You are now given 2 sets of points $S_1$ and $S_2$, each containing $n$ points. The distance between any pair of points in $S_1$ is at least $\delta$, and further, each point in $S_1$ has $z$-coordinate in $[0, \delta]$. Similarly, the distance between any pair of points in $S_2$ is at least $\delta$, and each point in $S_2$ has $z$-coordinate in $[-\delta, 0]$.

Extend the algorithm discussed in class to give an $O(n \log n)$-time algorithm for finding the closest pair of points in $S_1 \cup S_2$. Note that, by the first part of the question, any cube with each dimension at most $2\delta$, contains at most $2k$ points from $S_1 \cup S_2$.

(c) (10 marks) Given a set $S$ of $n$ points in $\mathbb{R}^3$, now give an $O\left(n \log^2 n\right)$-time algorithm to find the closest pair of points.

***Solution:***

■

**Problem 7** P9 (10 marks)

This problem relates to one of the questions asked in class. For any $p, q \geq 1$, and any points $x, y$, and $z \in \mathbb{R}^2$, prove or disprove the following:

$$\|x - y\|_p \leq \|x - z\|_p \Leftrightarrow \|x - y\|_q \leq \|x - z\|_q$$

That is, prove or disprove that $y$ is closer to $x$ than $z$ in the $L_p$ distance metric if and only if it is closer to $x$ in the $L_q$ distance metric As usual, $\|x - y\|_p = ((x_1 - y_1)^p + (x_2 - y_2)^p)^{1/p}$.

***Solution:***

■