Algorithms, Design & Analysis

Lecture 07: **Secretary Problem And Integer Multiplication**

Mohid & Wasif

Information Technology University

11th February,2025

About Your Fellows

- Hi there! We are **Mohid Faisal** and **Muhammad Wasif**.
- We are Associate Students at ITU.

Recap: High School Multiplication Algorithm

Time Complexity Analysis:

- Let n be the number of digits.
- There are n^2 digit multiplications.
- Addition and carry operations take at most $O(n)$.
- Overall, the time complexity is $O(n^2)$.

Can we do better?

Karatsuba Multiplication Algorithm

Karatsuba Algorithm $\rightarrow$ the divide and conquer algorithm for Integer Multiplication

Approach:

- Given two binary numbers: $X = 0011101010,\ Y = 1010000101.$
- Split them into two halves:
  - $X_h = \underline{00111},\ X_l = \underline{01010}$
  - $Y_h = \underline{10100},\ Y_l = \underline{00101}$
- Reconstruct:
  - $X = 2^{n/2} \times \underline{X_h} + \underline{X_l}$
  - $Y = 2^{n/2} \times \underline{Y_h} + \underline{Y_l}$
- Multiply using:

$$X \cdot Y = (2^{n/2}\underline{X_h} + \underline{X_l}) \cdot (2^{n/2}\underline{Y_h} + \underline{Y_l})$$

$$= 2^n \underline{X_h Y_h} + 2^{n/2}\underline{X_h Y_l} + 2^{n/2}\underline{Y_h X_l} + \underline{X_l Y_l}$$

- Next, we compute the smaller multiplications recursively and then add up the results along with bit shifts to get the final product.

Karatsuba Multiplication Algorithm - Explanation

Algorithm Overview

Given two numbers X and Y, the algorithm follows these steps:

1. If the number of bits is small (base case), multiply directly.
2. Split X into two halves: higher bits X_h and lower bits X_l.
3. Split Y into two halves: higher bits Y_h and lower bits Y_l.
4. Recursively compute:

$$hh = X_h \times Y_h$$
$$hl = X_h \times Y_l$$
$$lh = X_l \times Y_h$$
$$ll = X_l \times Y_l$$

5. Combine the results using bit shifts:

$$X \times Y = (2^n \times hh) + (2^{n/2} \times (hl + lh)) + ll$$

Recurrence Relation:

$$T(n) = 4T(n/2) + O(n)$$

By Master Theorem:

$$T(n) = \Theta(n^2)$$

Gauss's Insight

Gauss observed that the product of two complex numbers $(a+bi)\cdot(c+di)$ can be computed using only three multiplications instead of the naive four. Specifically:

$$(a+bi)\cdot(c+di) = (ac-bd)+(ad+bc)i$$

This can be rewritten using three multiplications:

- Compute ac.
- Compute bd.
- Compute $(a+b)(c+d)$.

Then, the term $ad+bc$ can be derived as:

$$ad+bc = (a+b)(c+d)-ac-bd$$

This reduces the number of multiplications from 4 to 3.

Optimization of Algorithm

We can do some algebra to express the multiplication that will require fewer recursive calls and give us a better big-O bound.

$$\underline{X} \cdot \underline{Y} = (2^{n/2}\underline{X_h} + \underline{X_l}) \cdot (2^{n/2}\underline{Y_h} + \underline{Y_l})$$

$$= 2^n \underline{X_h Y_h} + 2^{n/2}\underline{X_h Y_l} + 2^{n/2}\underline{Y_h X_l} + \underline{X_l Y_l}$$

Note that:

$$(\underline{X_h} + \underline{X_l}) \cdot (\underline{Y_h} + \underline{Y_l}) = \underline{X_h Y_h} + (\underline{X_h Y_l} + \underline{X_l Y_h}) + \underline{X_l Y_l}$$

Thus:

$$\underline{X_h Y_l} + \underline{X_l Y_h} = (\underline{X_h} + \underline{X_l}) \cdot (\underline{Y_h} + \underline{Y_l}) - \underline{X_h Y_h} - \underline{X_l Y_l}$$

Now, instead of four recursive calls, we will make three recursive calls.

Optimized Karatsuba Algorithm

The optimization reduces the number of recursive calls from 4 to 3 by computing:

$$X \cdot Y = (2^{n/2} X_h + X_l) \cdot (2^{n/2} Y_h + Y_l)$$

Expanding and rearranging using Gauss's trick:

$$X \cdot Y = 2^n X_h Y_h + 2^{n/2}(X_h Y_l + X_l Y_h) + X_l Y_l$$

Instead of computing $X_h Y_l$ and $X_l Y_h$ separately, we compute:

$$P = (X_h + X_l) \cdot (Y_h + Y_l)$$

Then, we derive:

$$X_h Y_l + X_l Y_h = P - X_h Y_h - X_l Y_l$$

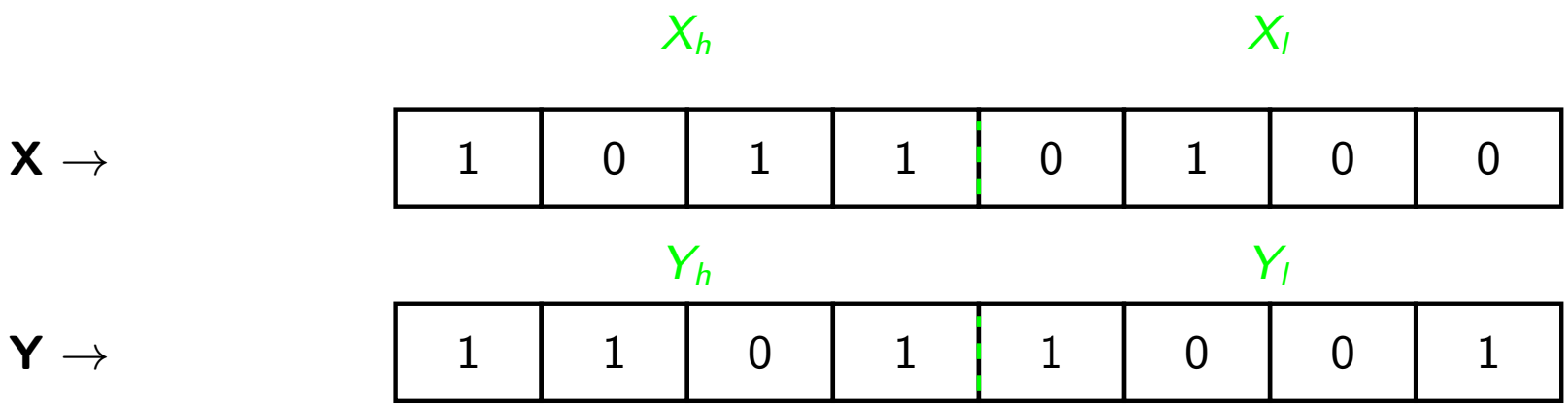
Optimized Recurrence Relation:

$$T(n) = 3T(n/2) + O(n)$$

By Master Theorem:

$$T(n) = \Theta(n^{\log_2 3})$$

Karatsuba Multiplication - Visual Representation



Recursive Multiplications:

$$hh = X_h \times Y_h$$

$$ll = X_l \times Y_l$$

$$(X_h + X_l) \times (Y_h + Y_l) - hh - ll$$

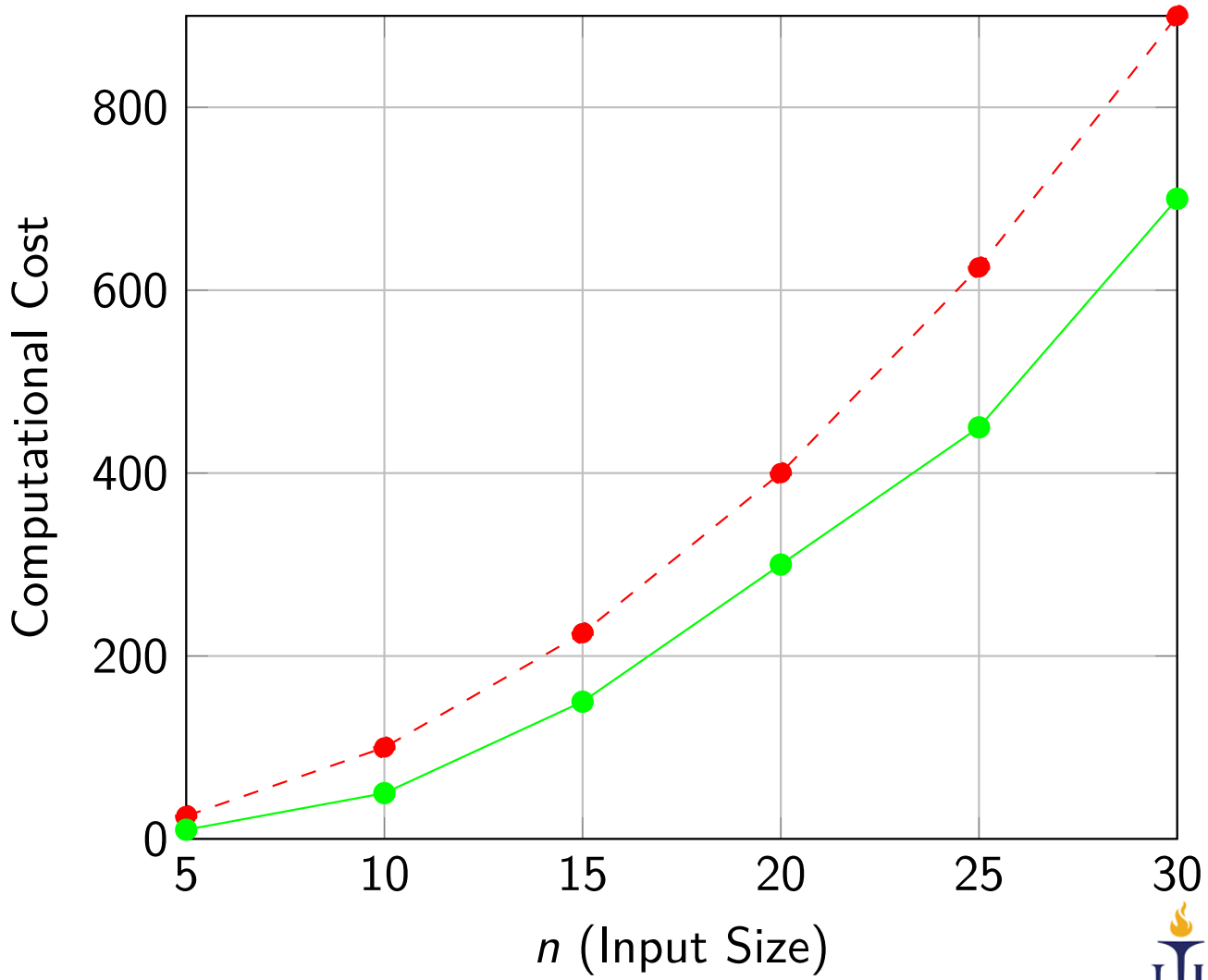
Final Result:

$$2^n \times hh + 2^{n/2} \times (hl + lh) + ll$$

Time Complexity Comparison

Multiplication Algorithms:

- $O(n^2)$ - High School Method
- $O(n^{1.585})$ - Karatsuba Algorithm



Strassen's Matrix Multiplication

Strassens algorithm is a **divide-and-conquer** approach for matrix multiplication that is asymptotically faster than the conventional method. It reduces the number of multiplications required, improving efficiency.

Standard Matrix Multiplication: For two $n \times n$ matrices A and B, the traditional matrix multiplication requires:

- $O(n^3)$ scalar multiplications.
- $O(n^3)$ scalar additions.

Strassens Approach

Strassens algorithm breaks down the multiplication process using **divide and conquer** and reduces the number of scalar multiplications.

Step 1: Divide the Matrices For two $n \times n$ matrices A and B, split each into four $\frac{n}{2} \times \frac{n}{2}$ submatrices:

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}, \quad B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

where each A_{ij} and B_{ij} is of size $\frac{n}{2} \times \frac{n}{2}$.

Strassens Algorithm: Recursive Multiplications

Step 2: Compute 7 Recursive Multiplications Instead of directly computing the 8 matrix products, Strassen defined 7 intermediate matrix products:

$$M_1 = (A_{11} + A_{22})(B_{11} + B_{22})$$

$$M_2 = (A_{21} + A_{22})B_{11}$$

$$M_3 = A_{11}(B_{12} - B_{22})$$

$$M_4 = A_{22}(B_{21} - B_{11})$$

$$M_5 = (A_{11} + A_{12})B_{22}$$

$$M_6 = (A_{21} - A_{11})(B_{11} + B_{12})$$

$$M_7 = (A_{12} - A_{22})(B_{21} + B_{22})$$

Strassens Algorithm: Compute Result Matrix

Step 3: Compute the Result Matrix Using the intermediate results, we compute the submatrices of C:

$$C_{11} = M_1 + M_4 - M_5 + M_7$$

$$C_{12} = M_3 + M_5$$

$$C_{21} = M_2 + M_4$$

$$C_{22} = M_1 - M_2 + M_3 + M_6$$

Time Complexity Analysis

Strassen's method reduces the number of multiplications from 8 to 7, but it increases the number of additions and subtractions.

Recurrence Relation:

$$T(n) = 7T(n/2) + O(n^2)$$

Solving using the Master Theorem:

$$T(n) = O(n^{\log_2 7}) \approx O(n^{2.81})$$

This is faster than the standard matrix multiplication complexity of $O(n^3)$.

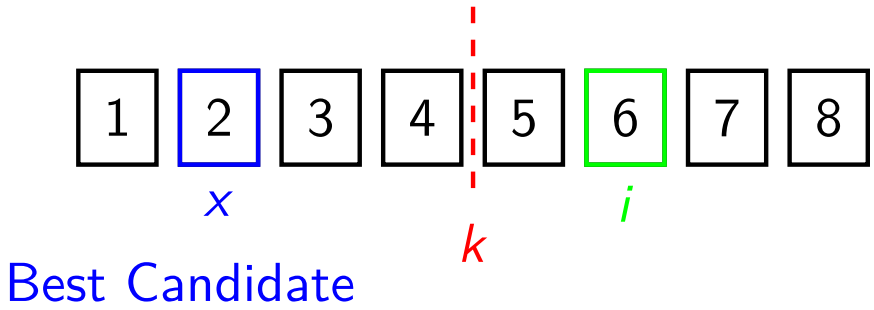
The Secretary Problem: Background and Rules

During WWII, new treatments were often tested on human subjects to determine their effectiveness. Researchers grappled with a critical question: when was the optimal point to stop testing and adopt a treatment? This dilemma inspired the Secretary Problem (or Optimal Stopping Problem) in computer sciencedeciding the best moment to stop searching among sequentially presented options to maximize the chance of selecting the best one.

Rules

- Candidates are interviewed one at a time in a random order.
- You must decide immediately after each interview.
- Once a candidate is rejected, you cannot go back and hire them.
- Only one candidate can be selected.
- The objective is to choose the best candidate overall.

The Secretary Problem: Algorithm



Algorithm

1. Determine the total number of candidates (n).
2. Interview the first k candidates and record the best candidate seen (x).
3. For each candidate from $k+1$ to n, if a candidate is better than x, hire them immediately and stop.
4. If no candidate after the observation phase is better, select the last candidate.

Let's analyze! But before that Some Background of Probability.

Probability Background

Uniform Probability: Definition

In a uniform probability modelwhere every outcome is equally likelythe probability of an event E is given by:

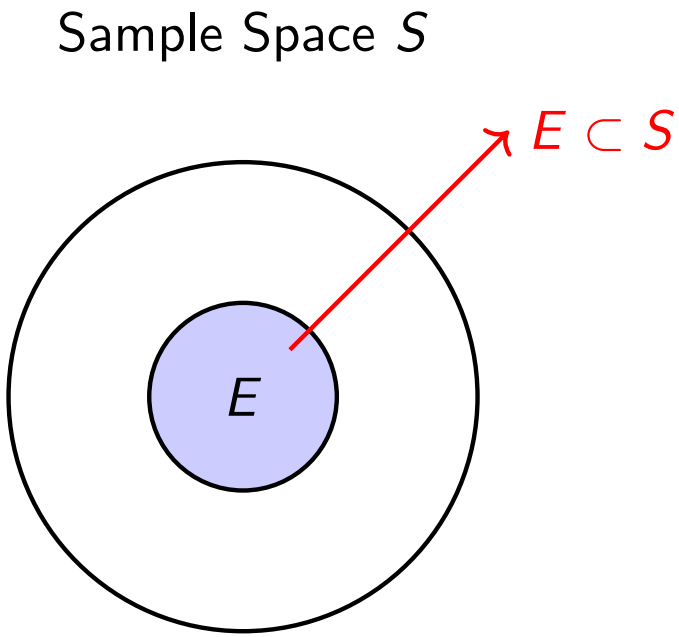
$$P(E) = \frac{|E|}{|S|},$$

where:

- $|S|$ is the total number of outcomes (sample space).
- $|E|$ is the number of favorable outcomes.

Consequently, for any outcome x in S,

$$P(x) = \frac{1}{|S|}.$$



Probability Background

Conditional Probability

The probability of an event E given that event F occurs [P(F)=1] is

$$P(E \mid F) = \frac{P(E \cap F)}{P(F)}, \quad P(F) > 0.$$

Independent Events

Events E and F are independent if the occurrence of one does not depend on the other.

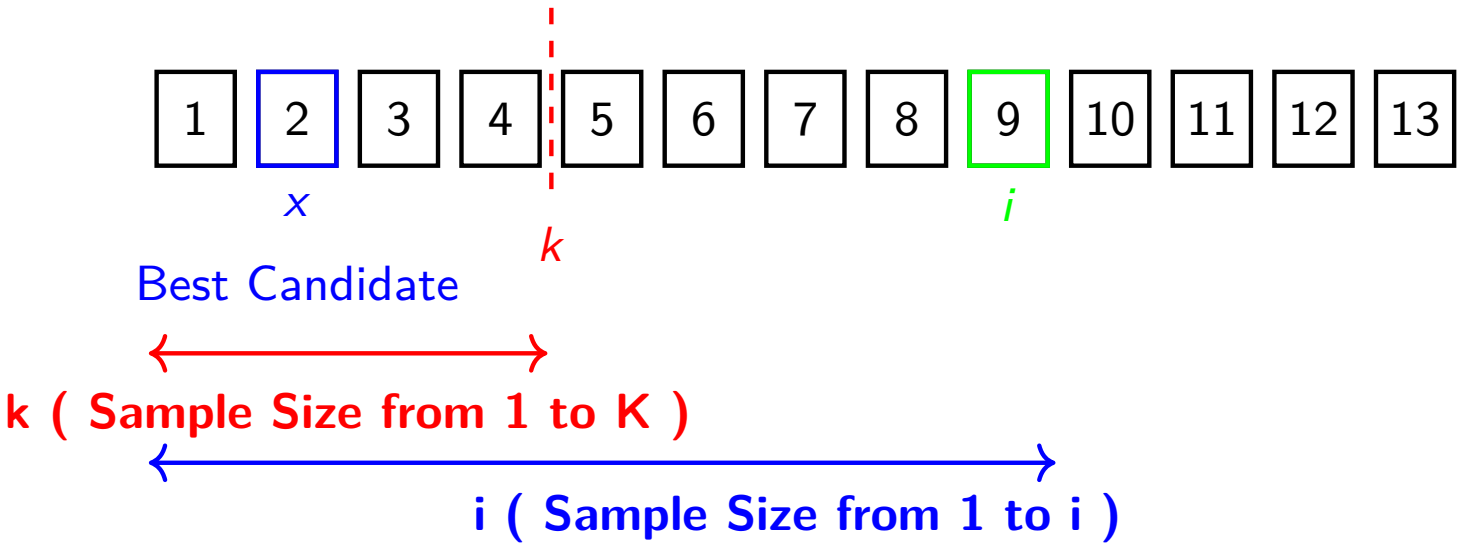
Since E and F are independent: $P(E) = P(E \mid F)$

and similarly

$$P(F) = P(F \mid E)$$

Probability of Optimal Hire

$$P(\text{OH}) = \text{we hire the best candidate}$$



$$P(OH) = \sum_{i=k+1}^{n} (P(i \text{ is best and we hire } i))$$

- Why use k+1 instead of 1?

  $P(\text{OH}) = 0$, Since we never hire the first k candidates

Probability of Optimal Hire

$$P(OH_i) = i \text{ is best and we hire } i$$

- B_i: i-th candidate is the best
- O_i: none of the applicants in $\{k+1 \ldots i-1\}$ are picked

$$OH_i = B_i \ \wedge \ O_i$$

$$P(\mathrm{OH}_i) = P\big(B_i \wedge O_i\big)$$

$$P(\mathrm{OH}_i) = P\big(B_i\big) \times \Pr\big(O_i \mid B_i\big)$$

- Since B_i and O_i are independent:

$$P(\mathrm{OH}_i) = P\big(B_i\big) \times P\big(O_i\big)$$

Probability of Optimal Hire

$$P(B_i) = \frac{1}{n}, \text{ [Uniform Probability]}$$

$$P(O_i) = \frac{k-1}{i-1}$$

$$P(\text{OH}_i) = P(B_i) \times P(O_i)$$

$$P(\text{OH}_i) = \frac{1}{n} \times \frac{k-1}{i-1}$$

$$P(\text{OH}) = \sum_{i=k+1}^{n} \left(\frac{1}{n} \times \frac{k-1}{i-1} \right)$$

Maximum Probability of Optimal Hire

Class Discussion

- **Professor:** What is the best value of k to maximize the Probability of Optimal Hire?
- **Sharib:** $k = \frac{n}{3}$.
- **Other students:** (suggested different values)
- **Professor:** We will find that in the next class. Time's up!