UNIT – 1

INTRODUCTION

1. Some Representative Problems
   - A First Problem: Stable Matching:
     - The Problem
     - Designing the Algorithm
     - Analyzing the Algorithm
     - Extensions
2. Five Representative Problems:
   - Interval Scheduling
   - Weighted Interval Scheduling
   - Bipartite Matching
   - Independent Set
   - Competitive Facility Location
3. Computational Tractability:
   - Some Initial Attempts at Defining Efficiency
   - Worst-Case Running Times and Brute-Force Search
   - Polynomial Time as a Definition of Efficiency
4. Asymptotic Order of Growth:
   - Properties of Asymptotic Growth Rates
   - Asymptotic Bounds for Some Common Functions
5. Implementing the Stable Matching Algorithm
   - Using Lists and Arrays: Arrays and Lists,
   - Implementing the Stable Matching Algorithm
6. A Survey of Common Running Times:
   - Linear Time
   - $O(n \log n)$ Time
   - Quadratic Time
   - Cubic Time
   - $O(nk)$ Time
   - Beyond Polynomial Time
   - Sub linear Time.

UNIT 1

INTRODUCTION

1. A first problem: Stable Matching

1.1 The problem:

- Designing a college admission process or job recruiting process that is self-enforcing.
- All juniors in college majoring in computer science begin applying to companies for summer internships.
- Application process is the interplay between two different types of parties.
  1. Companies (the employers)
  2. Students (the applicants)
- Each applicant has a preference ordering on companies and each company forms a preference ordering on its applicants.
- Based on these preferences, companies extend offers to some of their applicants, applicants choose which of their offers to accept.
- Gale and Shapely considered the sorts of things that could start going wrong with the process.
  1. ”**Raj**” accepted job at company “**CluNet**”.
  2. **”WebExodus”** offers job to **“Raj”**.
  3. **”Raj”** now prefers **“WebExodus”** and rejects **“CluNet”.**
  4. **”Kiran”** gets an offer from **“CluNet”**.
  5. **”Kiran”** already had accepted offer from **“BabelSoft”**.
  6. **”Kiran**” accepts offer from **“CluNet”** and rejects **“BabelSoft”**.
  7. **”Deepa”** who has accepted the offer from **“BabelSoft”**calls up**“WebExodus”** to join them (i.e. she preferred WebExodus over BabelSoft.
  8. **”WebExodus”** rejects **“Raj”** and accept **“Deepa”** (i.e. WebExodus preferred Deepa over Raj).
- Situation like this creates chaos and both applicants and employers endup unhappy with the process as well as outcome as the process is not self-enforcing and people are not allowed to act in their self-interest.

- According to Gale and Shapley: Given a set of preferences among employers and applicants, we can assign applicants to employers so that for every employer E, and every applicant A who is not scheduled to work for E, at least one of the following two things should hold:
  1. “E” prefers every one of its accepted applicants to “A”.
  2. “A” prefers her current situation over working for employer “E”.

  If this holds, the outcome is stable.
  - Individual self-interest will prevent any applicant/employer deal from being made behind the scene.

1.2. Formulating the problem:

- Each applicant is looking for a single company. Each company is looking for many applicants. Each applicant does not typically apply to every company.
- Each of **napplicants** applies to each of **ncompanies** and each company wants to accept a single applicant.

  (OR)
- **n**men and **n** women can end up getting married, in this case everyone is seeking to be paired with exactly one individual of opposite gender.
  - M is a set of n men, M={m_1,m_2,……,m_n}
  - W is a set of n women, W={w_1,w_2,……w_n}
  - M*W, is the set of all possible ordered pairs of form (m,w), where m∈M and w∈W

- **Matching:** A matching “**S**” is a set of ordered pairs, each from M*W, with the property that each member of M and each member of W appears in at most one pair in **S**.

Perfect matching

- A perfect matching S^1 is a matching with the property that each member of M and each member of W appears in exactly one pair in S^1.
- A perfect match is a way of pairing men with the women in such a way that everyone ends up married to somebody and nobody is married to more than one person. (i.e. neither singlehood nor polygamy).

Instability:

- Say there are 2 pairs (m,w) and (m^1,w^1) in S with property that
  - m prefers w^1 to w
  - w^1 prefers m to m1

The pair (m,w^1) is an instability with respect to S: (m,w^1) does not belong to S.

- Our goal is a set of marriages with no instabilities. A matching S is stable if:

  1. **It is perfect.**
  2. **There is no instability with respect to S.**

- **Example 1:**

  We have a set of two men, $\{m, m^1\}$ and a set of two women $\{w, w^1\}$. The preference lists are:

  m prefers w to w^1

  m^1 prefers w to w^1

  w prefers m to m^1

  w^1 prefers m to m^1

  - There is a unique stable matching, consisting of pairs (m,w) and (m^1,w^1).
  - (m^1,w) and (m,w^1) would not be a stable match, because the pair (m,w) would form an instability with respect to this matching.

- **Example 2:**
  m prefers w to w^1
  m^1 prefers w^1 to w
  w prefers m^1 to m
  w^1 prefers m to m^1

  - (m,w) and (m^1,w^1) is stable, because both men are happy as neither would leave their matched partners.
  - (m^1,w) and (m,w^1) is stable as both women are happy.
  - So its possible for an instance to have more than one stable matching.

1.3 <u>Designing the Algorithm:</u>

<u>Basic steps:</u>

1. Initially, everyone is unmarried.
   - If an unmarried man **m** chooses woman **w** who ranks highest on his preference list and proposes her.
   - A mam **m^1** whom **w** prefers, **w** may or may not receive a proposal from **m^1**.