TRUBA
Turkish Science e-Infrastructure

EURO

COMPUTHON 2021/1

TÜBİTAK ULAKBİM

ORTA DOĞU TEKNİK ÜNİVERSİTESİ
MIDDLE EAST TECHNICAL UNIVERSITY

Sabancı Üniversitesi

Computhon Schedule

EURO

July 8th (Thursday, Midnight):

The Last Day of Registration

July 10th (Saturday):

Training Day & Problem Announcement

July 15th (Thursday, Midnight):

The Last Day of Solution Submission

July 17th (Saturday):

Presentations & Final Evaluation

Announcement of Winners

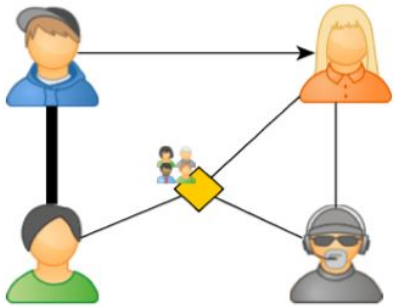
Today's (Tentative) Program

EURO

09:00 - 10:00: EuroCC project & TRUBA
10:00 - 10:30: Problem Definition
10:30 - 12:00: OpenMP Programming
12:00 - 13:00: Lunch Break
13:00 - 15:00: MPI Programming
15:00 - 15:15: Coffee Break
15:15 - 17:00: CUDA Programming
17:00 - 17:30: Questions

Graphs and Networks

- Graphs are used to model the relationships of elements with each other.
- Different vertex and edge types can be used to model different types of relationships.



Examples of Networks

EURO

NETWORK	VERTICES	VERTEX ATTRIBUTES	EDGES	EDGE ATTRIBUTES
Airlines Network	**Airports**	Footfall, Terminals, Staff, City population, International/Domestic, Freight, Hangar capacity	**Airplanes / Routes**	Frequency, # Passengers, Plane Type, Fuel Usage, Distance covered, Empty seats
Banking Network	**Account Holders**	Name, demographics, KYC Document, Products, Account status, balance and other details	**Transactions**	Type, Amount, Authentication (pass/OTP), Time, Location, Device
Social Network	**Users**	Name, demographics, # connections, likes, circles belong to, subscriptions	**Interactions**	Medium (like/comment/direct message), time, duration, type of content, topic
Physician Network	**Doctors**	Demographics, speciality, experience, affiliation (type and size), Weekly patient intake	**Patients**	Demographics, Diagnosis history, visit frequency, purpose, referred to, insurance
Supply Chain Network	**Warehouses**	Location, size, capacity, storage type, connectivity, manual/automated	**Trucks**	Load capacity, # wheels, year of make, geographical permit, miles travelled. Maintenance cost, driver experience

Graph Analytics

EURO

Marketing Analytics: Graphs can be used to find the most influential people on a social network. By directing their messages through the most influential people on a Social Network, advertising and marketing departments can predict the biggest boom in revenue.

Banking: Graphs can be used to find unusual patterns that help reduce fraud. There have been instances where illegal activity has been detected by analyzing the flow of money through interconnected banking networks.

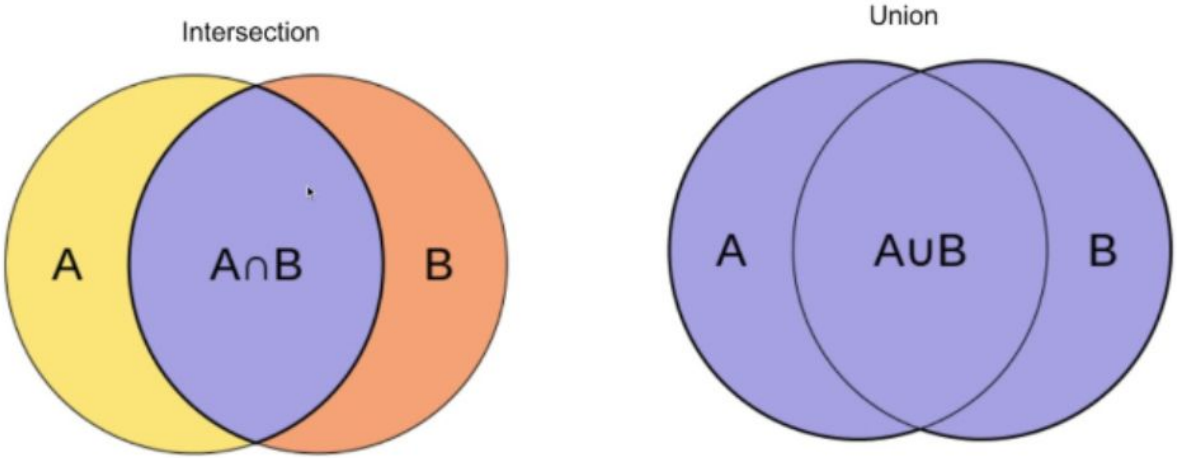
Graph Analytics

EURO

Telecom: Telecom companies often use graphs (Voronoi diagrams) to find the optimal amount and locations of Cell towers to provide maximum coverage.

Supply Chain: Graphs help you determine optimal routes for your delivery trucks and locate locations for warehouses and delivery centers.

Problem Definition: Jaccard Similarity



$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

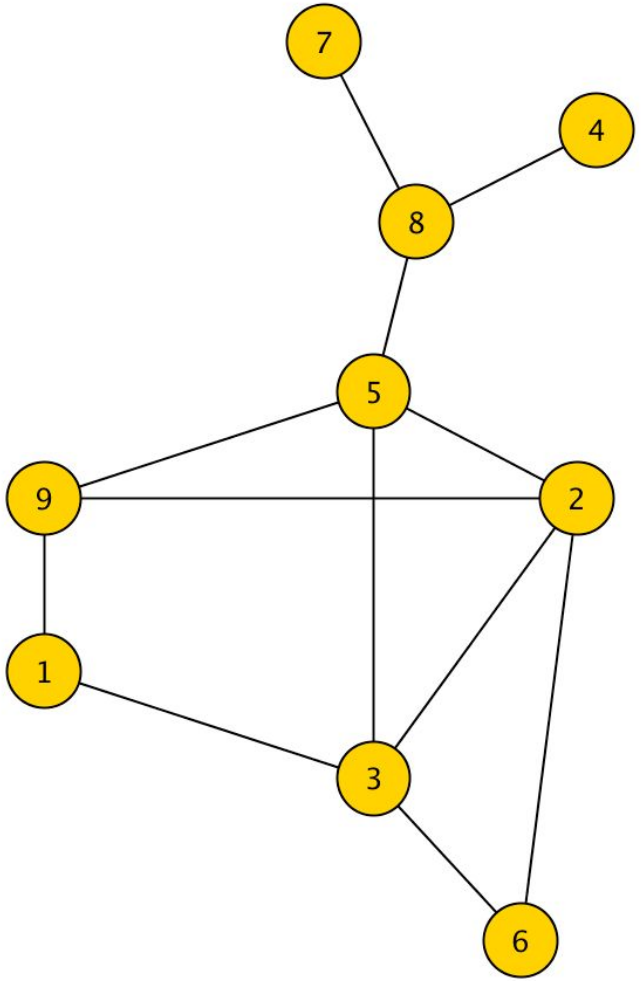
Problem Definition: Jaccard Similarity

On graphs and networks, sets are neighborhoods of the vertices. Hence, each edge is related with two sets.

$$N(2) = \{3,5,6,9\}$$

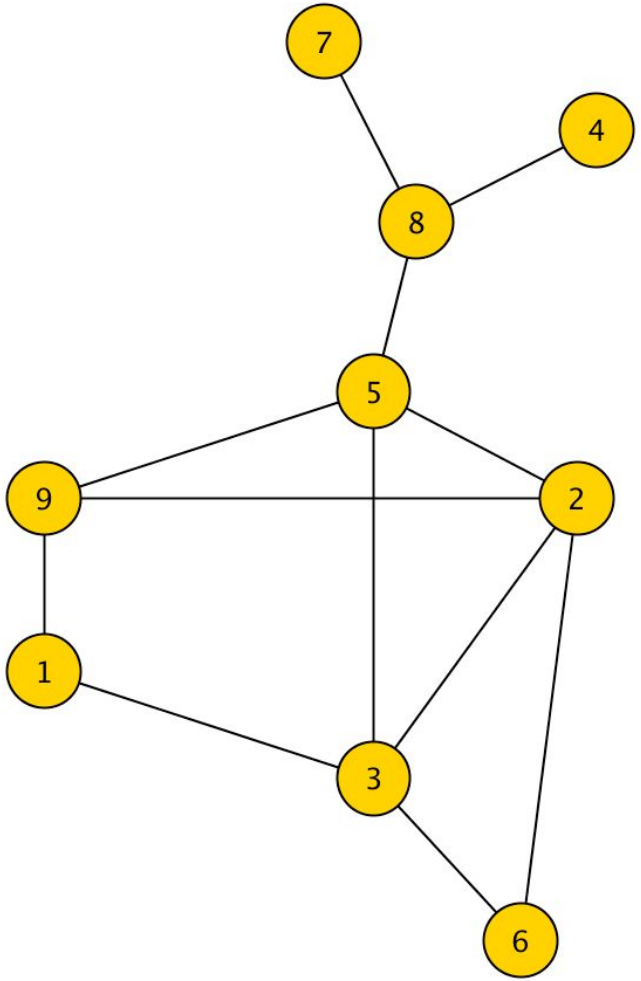
$$N(5) = \{2,3,8,9\}$$

So we can have Jaccard values for edges.

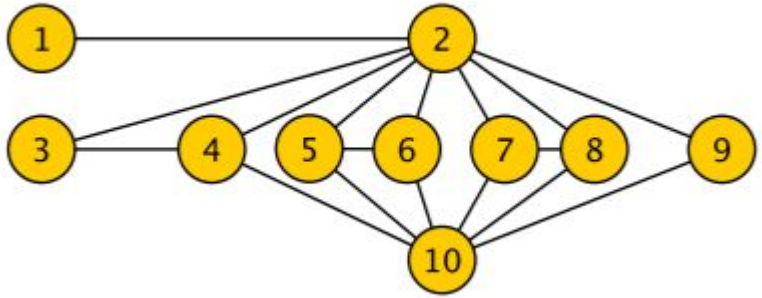


Sample Graph

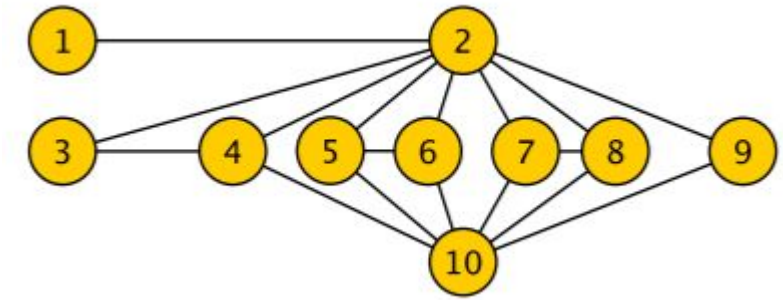
```
J(1,3) = 0.000

J(2,3) = 0.333

J(2,5) = 0.400
```



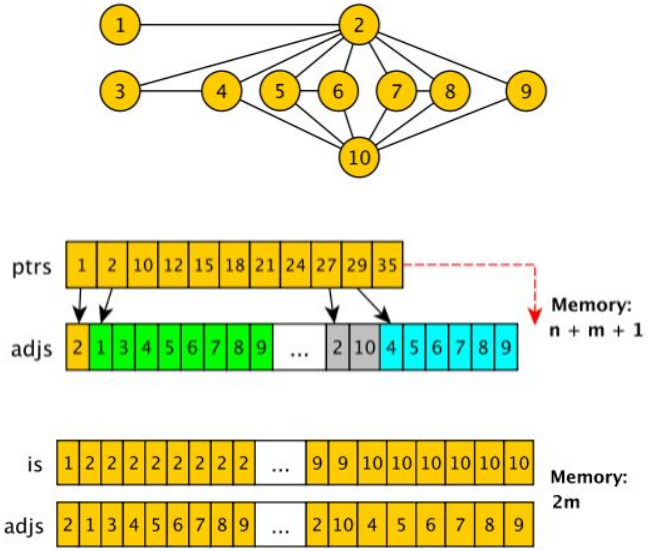
Graph Structures



Graph Data Structures



Graph Data Structures




```
struct CSR {
  int _n;
  int _m;
  int *_xadj;
  int *_adj;
  int *_is;
  CSR(int n, int m, int* xadj, int* adj, int* is):
    _n(n), _m(m), _xadj(xadj), _adj(adj), _is(is){}
};
```

Computing Jaccard Similarity

EURO

```
/////// BEGIN CALCULATION CODE
for (int u = 0; u < n; u++){
  for (int v_ptr = xadj[u]; v_ptr < xadj[u+1]; v_ptr++){
    int v = adj[v_ptr]; // v is a neighbor of u
    int num_intersections = 0;
    unordered_set<int> uv_union; // A mathematical set (with no duplicates). Union of neighborhoods of u and v
    for (int u_nbr_ptr = xadj[u]; u_nbr_ptr < xadj[u+1]; u_nbr_ptr++){ // Go over all neighbors of u
      int u_nbr = adj[u_nbr_ptr];
      uv_union.insert(u_nbr); // Add this neighbor of u to the union of neighborhoods
      for (int v_nbr_ptr = xadj[v]; v_nbr_ptr < xadj[v+1]; v_nbr_ptr++){ // Go over all neighbors of v
        int v_nbr = adj[v_nbr_ptr];
        uv_union.insert(v_nbr); // Add this neighbor of v to the union of neighborhoods
        if (u_nbr == v_nbr){ // Neighbors of u and v match. Increment the intersections
          num_intersections++;
        }
      }
    }
    int num_union = uv_union.size(); // The set contains all the non-repeated neighbors of u and v
    jaccard_values[v_ptr] = float(num_intersections)/float(num_union);
  }
}
/////// END CALCULATION CODE
```

Computing Jaccard Similarity
