Data Structures

Jan Michael C. Yap
Core Facility for Bioinformatics
jcyap@up.edu.ph

All concepts in this lesson were taken from the book Data Structures by Evangel P. Quiwa

Outline

- Data Structures
- Linear Lists
- Tables
- Graphs

Outline

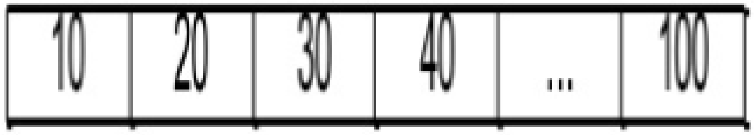
- **Data Structures**
- Linear Lists
- Tables
- Graphs

Data Structures

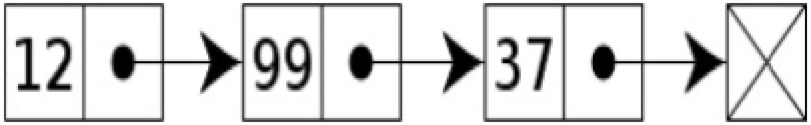
- A **data type** is a **kind of data** that variables may hold in a programming language **plus the operations automatically provided**
  - E.g. character, integer, and floating point
- An **abstract data type (ADT)** is a **set of data elements** and the **set of operations defined on the data elements**
  - E.g. Stacks and queues
- A **data structure** is an **implementation or realization of an ADT** in terms of language data types or other data structures such that operations on the data structure are expressible in terms of directly executable procedures

Data Structure Design

- **Sequential or Contiguous Design**
  - A **pointer** references the **address of the first memory block** (representing the first element) in the data structure, and the subsequent elements are adjacent to the (address of the) first memory block.



- **Linked design**
  - Each element in the data structure has a **link pointing to the address of the next element**, which **may not be adjacent to the address of the element next to it**
  - A pointer still references the address of the first element in the structure



Outline

- Data Structures
- **Linear Lists**
- Tables
- Graphs

Linear List

- A list is a **finite, ordered set of zero or more elements**
  – Elements may be **atoms or lists**

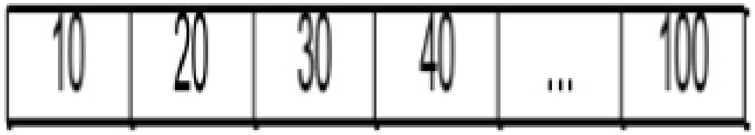
$$L = ((\mathrm{a},\mathrm{b}),\mathrm{c},(()),((\mathrm{d},\mathrm{e}),\mathrm{f}))$$

- A **linear list** is a list where all elements are **atoms**
  – A generalized list (or list structure) is a list where an element may either be a list or an element

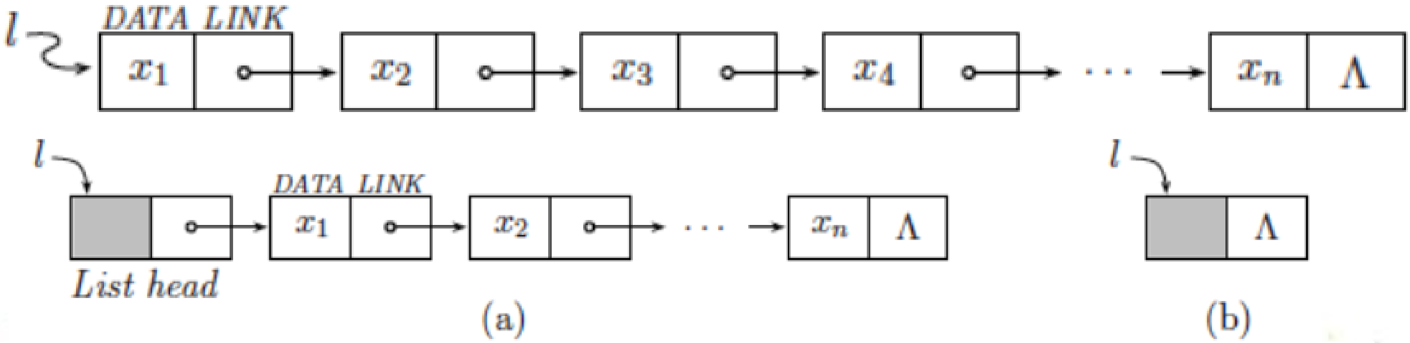
$$L = (x_1, x_2, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_{n-1}, x_n)$$

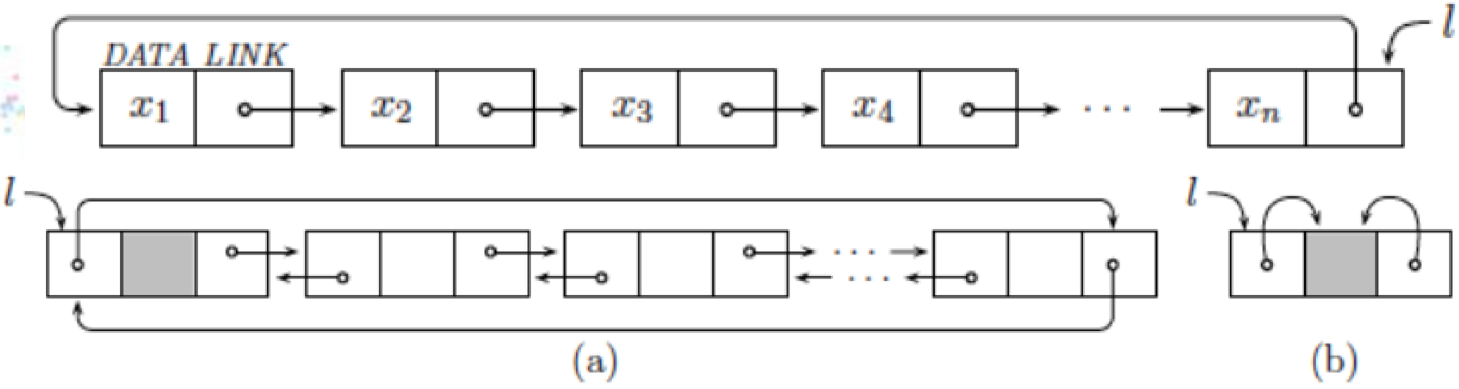
Linear List Implementations

- Contiguous design



- Linked design



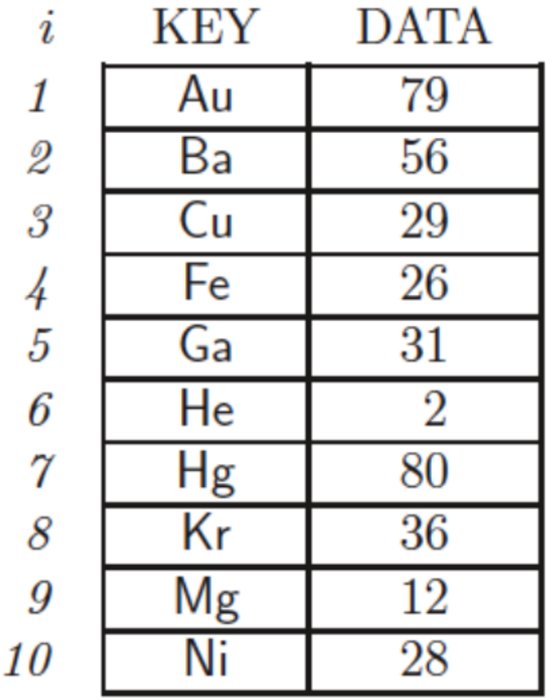


Outline

- Data Structures
- Linear Lists
- **Tables**
- Graphs

Sequential Tables

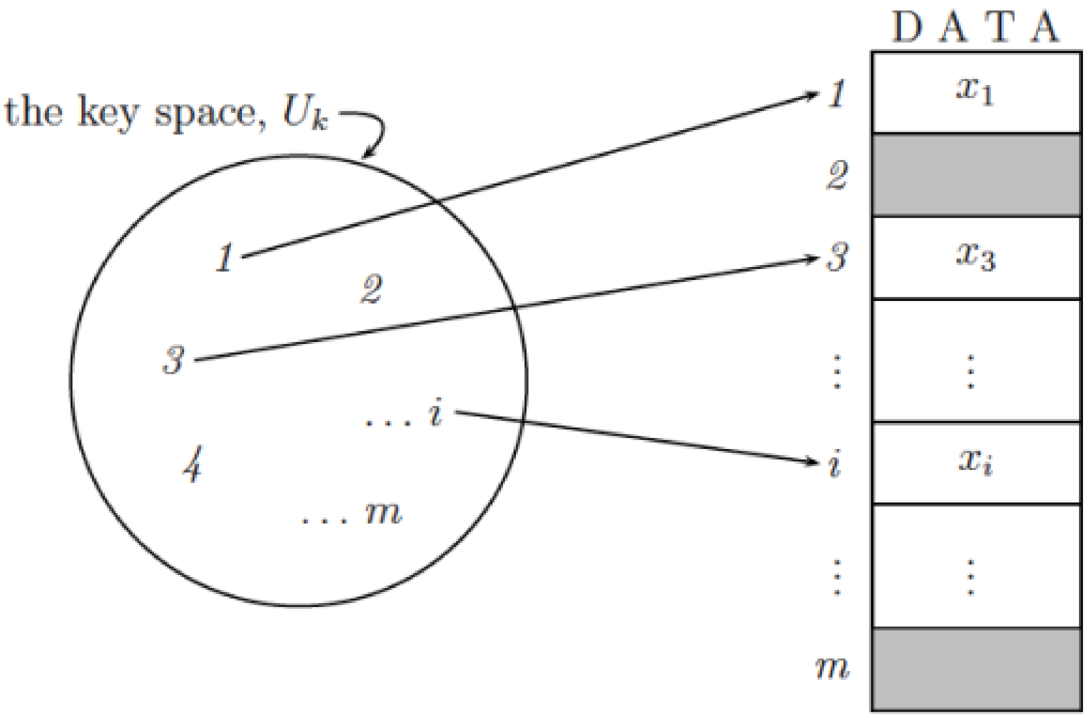
- A table where a **key/identifier** is assigned to each element in the table that will be **used for searching** the structure
- Usually implemented using **sequential/contiguous design**



i	KEY	DATA
1	Au	79
2	Ba	56
3	Cu	29
4	Fe	26
5	Ga	31
6	He	2
7	Hg	80
8	Kr	36
9	Mg	12
10	Ni	28

Direct Access Tables

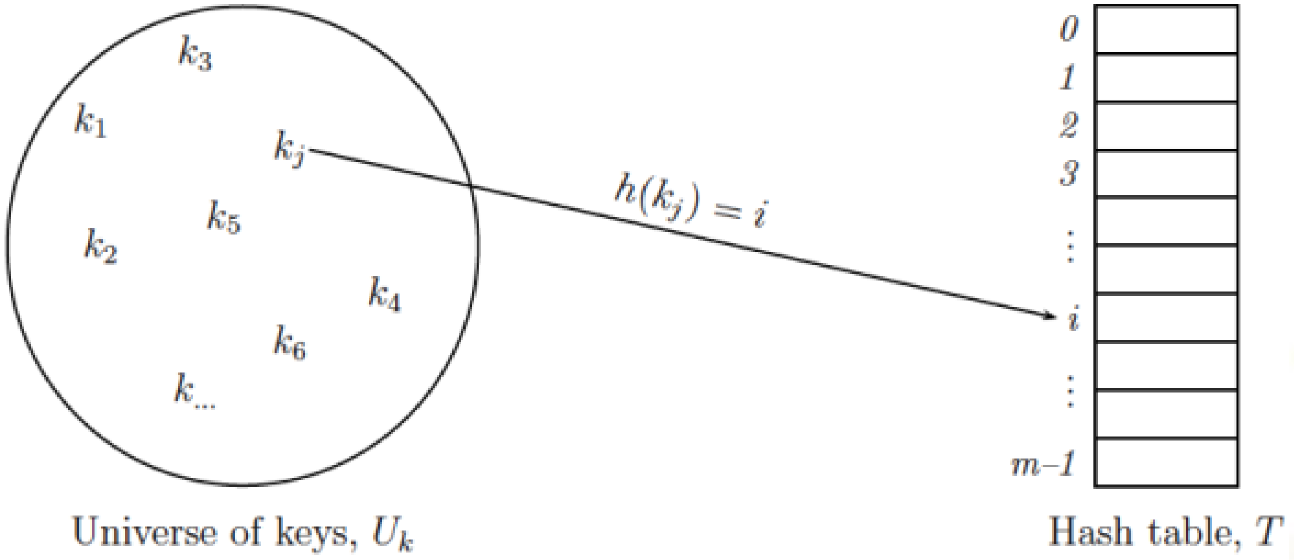
- A table where a key/identifier assigned to each element is the **index/position where it is stored in the table**



Hash Tables

- A table that uses a **hash function** to **convert a (user-defined) key/identifier to an index/position** where an element will be stored in the table

$$h : U_k \rightarrow \{0, 1, 2, \ldots, m-1\}$$



Hash Tables: Implementation Considerations

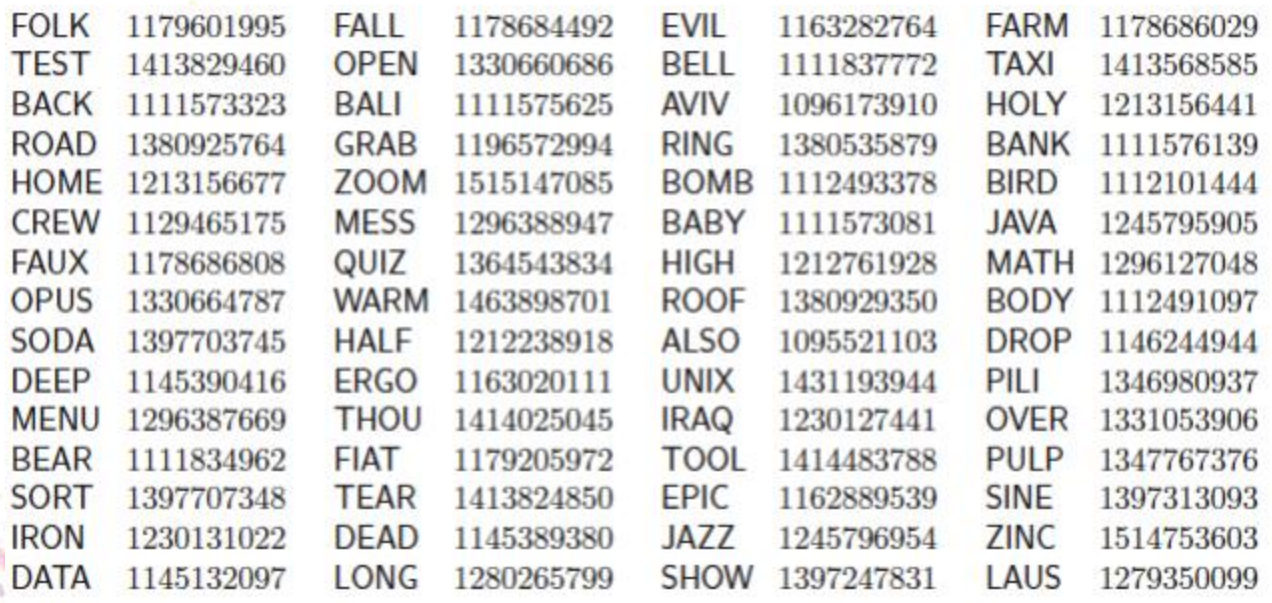
- Convert **non-numeric keys to numbers**
- Choose a **good hash function**
- Choose a **collision resolution** policy

Converting Non-numeric Keys to Numbers

$$\mathsf{KEYS} = 11052519$$

$$\mathsf{KEYS} = 11 \times 26^3 + 05 \times 26^2 + 25 \times 26^1 + 19 \times 26^0 = 197385$$

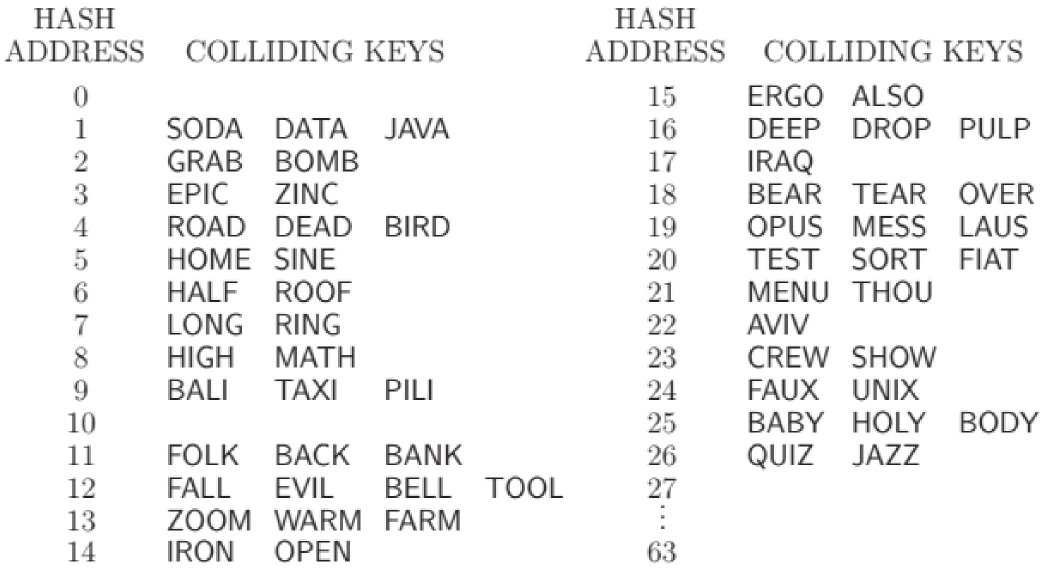
$$\mathsf{KEYS} = 01001011\,01000101\,01011001\,01010011_2$$
$$= 75 \times 256^3 + 69 \times 256^2 + 89 \times 256^1 + 83 \times 256^0 = 1262836051_{10}$$



FOLK	1179601995	FALL	1178684492	EVIL	1163282764	FARM	1178686029
TEST	1413829460	OPEN	1330660686	BELL	1111837772	TAXI	1413568585
BACK	1111573323	BALI	1111575625	AVIV	1096173910	HOLY	1213156441
ROAD	1380925764	GRAB	1196572994	RING	1380535879	BANK	1111576139
HOME	1213156677	ZOOM	1515147085	BOMB	1112493378	BIRD	1112101444
CREW	1129465175	MESS	1296388947	BABY	1111573081	JAVA	1245795905
FAUX	1178686808	QUIZ	1364543834	HIGH	1212761928	MATH	1296127048
OPUS	1330664787	WARM	1463898701	ROOF	1380929350	BODY	1112491097
SODA	1397703745	HALF	1212238918	ALSO	1095521103	DROP	1146244944
DEEP	1145390416	ERGO	1163020111	UNIX	1431193944	PILI	1346980937
MENU	1296387669	THOU	1414025045	IRAQ	1230127441	OVER	1331053906
BEAR	1111834962	FIAT	1179205972	TOOL	1414483788	PULP	1347767376
SORT	1397707348	TEAR	1413824850	EPIC	1162889539	SINE	1397313093
IRON	1230131022	DEAD	1145389380	JAZZ	1245796954	ZINC	1514753603
DATA	1145132097	LONG	1280265799	SHOW	1397247831	LAUS	1279350099

Hash Function: Example

- **Division method**: h(k) = k mod m
  - m is the **size of the table**
  - Hash table is assumed to be **indexed between 0 to m - 1**
  - mod operation takes the **remainder when k is divided by m**
    - Guarantees that the **index where the element will be hashed would be between 0 and m – 1 inclusive**



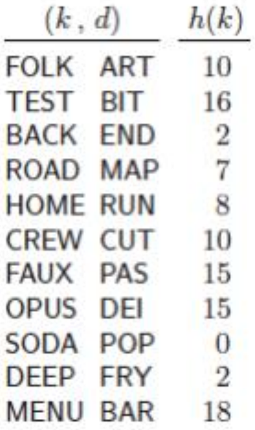
HASH ADDRESS	COLLIDING KEYS				HASH ADDRESS	COLLIDING KEYS		
0					15	ERGO	ALSO	
1	SODA	DATA	JAVA		16	DEEP	DROP	PULP
2	GRAB	BOMB			17	IRAQ		
3	EPIC	ZINC			18	BEAR	TEAR	OVER
4	ROAD	DEAD	BIRD		19	OPUS	MESS	LAUS
5	HOME	SINE			20	TEST	SORT	FIAT
6	HALF	ROOF			21	MENU	THOU	
7	LONG	RING			22	AVIV		
8	HIGH	MATH			23	CREW	SHOW	
9	BALI	TAXI	PILI		24	FAUX	UNIX	
10					25	BABY	HOLY	BODY
11	FOLK	BACK	BANK		26	QUIZ	JAZZ	
12	FALL	EVIL	BELL	TOOL	27			
13	ZOOM	WARM	FARM		⋮			
14	IRON	OPEN			63			

Collision Resolution

- **Birthday Paradox**: How many persons you have to put in a room so you could have a good 50% chance that two of those persons have
  the same birthday (not necessarily the same year)?
- This is a basis for the **collision problem**, and is in fact common even in sparsely occupied hash tables

$$h(K), h(K)-1, h(K)-2, \ldots, 1, 0, m-1, m-2, \ldots, h(K)+1$$

$$p(K) = [h(K) - i] \bmod m \qquad i = 0, 1, 2, \ldots, m-1$$



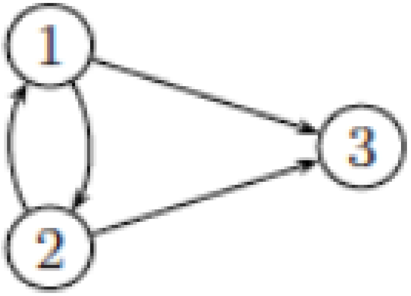
(k, d)		$h(k)$
FOLK	ART	10
TEST	BIT	16
BACK	END	2
ROAD	MAP	7
HOME	RUN	8
CREW	CUT	10
FAUX	PAS	15
OPUS	DEI	15
SODA	POP	0
DEEP	FRY	2
MENU	BAR	18

Outline

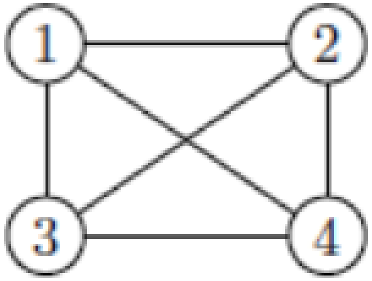
- Data Structures
- Linear Lists
- Tables
- **Graphs**

Graphs

- A **graph** consists of a finite nonempty set of **vertices**, and a finite possibly empty set of **edges**
  - Usually denoted by **G = (V,E)**
  - Edges **may or may not be weighted**
- **Directed graph (or digraph)**

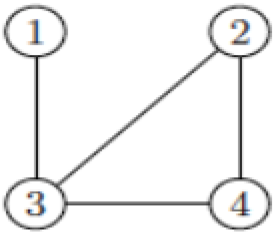


- **Undirected graph**

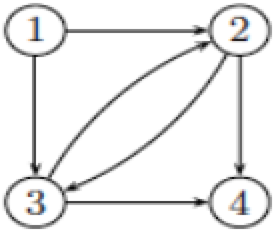


Graphs: Sequential Design

- Adjacency Matrix

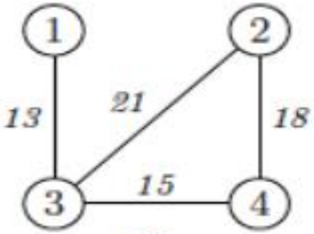


$$\begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 & 1 \\ 3 & 1 & 1 & 0 & 1 \\ 4 & 0 & 1 & 1 & 0 \end{array}$$



$$\begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 1 & 1 & 0 \\ 2 & 0 & 0 & 1 & 1 \\ 3 & 0 & 1 & 0 & 1 \\ 4 & 0 & 0 & 0 & 0 \end{array}$$

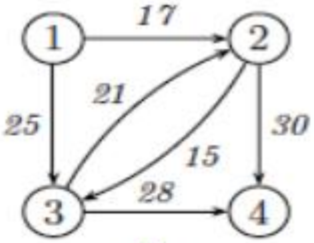
- Cost Adjacency Matrix



G_1

$$\begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & \infty & 13 & \infty \\ 2 & \infty & 0 & 21 & 18 \\ 3 & 13 & 21 & 0 & 15 \\ 4 & \infty & 18 & 15 & 0 \end{array}$$

C_1



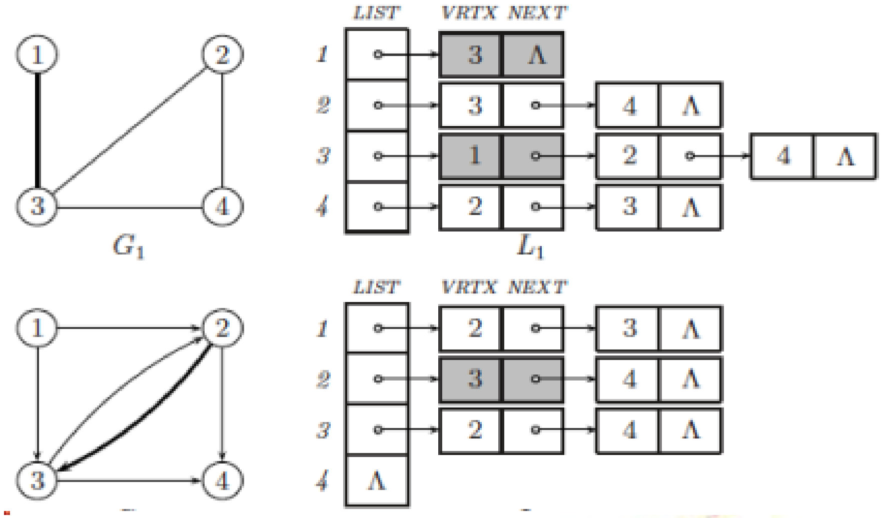
G_2

$$\begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 17 & 25 & \infty \\ 2 & \infty & 0 & 15 & 30 \\ 3 & \infty & 21 & 0 & 28 \\ 4 & \infty & \infty & \infty & 0 \end{array}$$

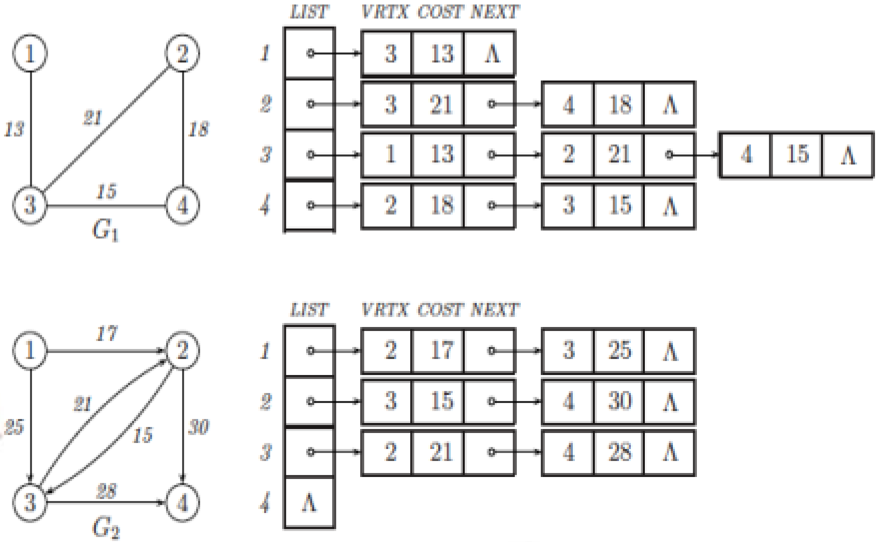
C_2

Graphs: Linked Design

- Adjacency List



- Cost Adjacency List



THANK YOU VERY MUCH! ☺