Unit 11

Organizing and Illustrating Data

DATA REPRESENTATION AND DATA STRUCTURES

Pre-reading Activities

In this unit, you will

- improve your understanding of the target technical words.
- learn about controlling your ideas in writing a paragraph.
- learn how to preview a reading comprehension passage through pre-reading questions to improve comprehension.
- be familiar with the organizing and illustrating data.

I. Target Academic Vocabulary

Check out the meanings and functions of the target academic words in a monolingual and bilingual dictionary.

Executable (adj)

Ascend (v)

Array (n)

Shrink (v)

Modify (v)

Fetch (v)

Inferior (adj)

Account for (v)

II. Writing development

Supporting topic sentences with Contrast

The definition and four common structures used with *contrast* in a contrastive paragraph were discussed. In this unit, three other common structures, such as *verbal structures, sentence connectors*, and *conjunctions* are introduced.

A. Verbal structures (differ from; contrast with; be [am, is, are,...] different from)

- Nonnumeric data differ from numeric data in regard to storing data in a computer.

- Nonnumeric data contrast with numeric data in respect to storing data in a computer.
- Nonnumeric data are different from numeric data in respect to storing data in a computer.

B. Sentence connectors (however; on the other hand; in contrast)

- Nonnumeric data are converted into numbers before storing in a computer; *however*, numeric data are complete and can straightly be stored in a computer.
- Nonnumeric data are converted into numbers before storing in a computer; *on the other hand*, numeric data are complete and can straightly be stored in a computer.
- Nonnumeric data are converted into numbers before storing in a computer; *in contrast*, numeric data are complete and can straightly be stored in a computer.

Punctuation note: *There should be a semicolon (;) before sentence connectors and a comma (,) after them, when they appear at the beginning of a next sentence.*

C. Conjunction (but)

Nonnumeric data are converted into numbers before storing in a computer, *but* numeric data are complete and can straightly be stored in a computer.

Punctuation note: *There* should be a comma (,) before a conjunction between the two sentences.

III. Pre-reading questions:

Read and respond to the questions below, and then discuss them in pair/group.

1. Does anyone have any ideas in regard to numeric and nonnumeric data?

---

2. What are the existing data structures? Describe one or two of them briefly.

---

3. What are the basic principles for choosing a good data structure?

IV.Reading comprehension passage

This article discusses data representation and date structures. It also describes different data structures and basic principles for choosing the best data structures.

DATA REPRESENTATION AND DATA STRUCTURES

The computer is a powerful tool for data processing, which needs to be stored in computer memory for any process. Computer memory is constructed in binary cells. Hence, all data in digital computers is stored in a sequence of 0s and 1s. This includes numeric data, text, executable files, images, audio, and video.

Data representation refers to the methods used internally to represent information stored in a computer. Data are either numeric or nonnumeric. Numeric data can be in integer or in floating point type. Two different standards exist in a computer to hold each of the numeric data. Processing the former data type is more straightforward for a computer compared to the other data type. However, in a defined memory location, floating point type number can be broader with more precision compared to integer data type.

Nonnumeric data, e.g., a text, is stored in the computer memory as a sequence of numbers. There are standards, such as ASCII code (American Standard Code for Information Interchange) standard to convert text characters into numbers by assigning a unique numeric value for each symbol used in the text. These numbers are unsigned, hence the unsigned data type is defined to store nonnumeric data.

1. Data structure

Representing information is fundamental to computer science. The primary purpose of most computer programs is not to perform calculations, but to store and retrieve information — usually as fast as possible. To efficiently use and process stored data in a computer, a data

structure is defined in computer science. There are different kinds of data structures, each of which is suited for specific applications. Usually, efficient data structures are a key to designing efficient algorithms. Some formal design methods and programming languages emphasize data structures, rather than algorithms, as the key organizing factor in software design. In addition, in terms of memory space and time limit, some operations can only be carried out on data with a well-designed structure. For example, an array data structure is suitable for sorting data in ascending or descending order, and B-trees data structure are useful for implementation of databases.

Moreover, data structures provide a means to manage large amounts of data efficiently, such as large databases and Internet indexing services. The power of a computer for searching in a large database, and developing object-oriented concepts in programming languages are due to the power of data structures. Here we briefly review some of the existing data structures:

- An array data structure stores a number of elements (for example, integer numbers) in a specific order. They are accessed using an index to specify which element is required.
- A record is the simplest data structure, which contains a fixed number of values or data types typically indexed by names. The elements of records are usually called fields or members. Each field can be accessed separately, and a record can be passed to a procedure at once for any process.

• A set is a data structure that can store specific values, without any particular order, and no repeated values. Values themselves are not retrieved from sets, rather one tests a value for membership to obtain a Boolean "in" or "not in".

• A linked list is a data structure with successive elements connected via a pointer. It can grow or shrink during the execution of a program. In other words, each element can be created whenever needed. This feature of linked list helps the programmer to control the usage of memory space.

• The stack is one of the most important data structures in computer science. Indeed, a stack is a means of storing information in a computer. The stored data in a stack can be accessed in last-in-first-out order. In other words, when a new object is entered in a stack, it is placed on top of all the previously entered objects. Stack data structures are suitable for implementing recursive algorithms, and defining local variables.

• Hashing is a data structure used to implement an associative array, a structure that can map keys to values. In hashing, a hash function is used to compute an index into an array, from which the correct value can be found.

• An object contains a number of data fields, like a record, and also a number of program code fragments for accessing or modifying them.

2. Basic principle of data structures

Data structures are generally based on the ability of a computer to randomly fetch and store data at any place in its memory, specified by an address—a bit string that can itself be stored in memory and

manipulated by the program. The record and array data structures, for example, are based on computing the addresses of data items with arithmetic operations, while the linked data structures are based on storing addresses of data items within the structure itself. The implementation of some other data structures, for example stack data structure, usually requires writing a set of procedures that create and manipulate instances of that structure.

3. *Choosing a data structure*

Each data structure is associated with costs and benefits. In practice, it is hardly ever true that one data structure is better than another for use in all situations. If one data structure or algorithm is superior to another in all respects, the inferior one will usually have long been forgotten. For nearly every existing data structure and algorithm, there are examples of where it is the best choice. A data structure requires a certain amount of space for each data item, time to perform a single basic operation, and programming effort. Each problem has constraints on available space and time.

Each solution to a problem makes use of the basic operations in some relative proportion, and the data structure selection process must account for this. Only a careful analysis of a problem's characteristics can determine the best data structure for the task. As an example, a bank must support many types of transactions with its customers, but we will examine a simple model where customers wish to open accounts, close accounts, add money to or withdraw money from their accounts. We can consider this problem at two distinct levels: (1) the requirements

for the physical infrastructure and workflow process that the bank uses in its interactions with its customers, and (2) the requirements for the database system that manages the accounts.

The typical customer opens and closes accounts far less often than he or she accesses to the account. Customers are willing to wait many minutes while accounts are created or deleted but are typically not willing to wait more than a brief time for individual account transactions such as a deposit or withdrawal. These observations can be considered as informal specifications for the time constraints on the problem.

It is a common practice for banks to provide two tiers of service. Human tellers or Automated Teller Machines (ATMs) support customer access to account balances and updates such as deposits and withdrawals. Special service representatives are typically provided (during restricted hours) to handle opening and closing accounts. Teller and ATM transactions are expected to take little time. Opening or closing an account can take much longer (perhaps up to an hour from the customer's perspective).

From a database perspective, we see that ATM transactions do not modify the database significantly. For simplicity, assume that if money is added or removed, this transaction simply changes the value stored in an account record. Adding a new account to the database is allowed to take several minutes. Deleting an account does not have time constraint, because from the customer's point of view, all the money would be returned (equivalent to a withdrawal). From the bank's point

of view, the account record might be removed from the database system after business hours, or at the end of the monthly account cycle.

When considering the choice of data structure to use in the database system that manages customer accounts, we see that a data structure that has little concern for the cost of deletion, but is highly efficient for search and moderately efficient for insertion, should meet the resource constraints imposed by this problem. Records are accessible by unique account numbers (sometimes called an exact-match query). One data structure that meets these requirements is the hashing function described above. Hashing function allows for extremely fast exact-match search. A record can be modified quickly when the modification does not affect its space requirements. Hashing function also supports efficient insertion of new records. While deletions can also be supported efficiently, too many deletions lead to some degradation in performance for the remaining operations. However, the hashing function can be reorganized periodically to restore the system to peak efficiency. Such reorganization can occur offline so as not to affect ATM transactions.

Suppose a company wants to develop a database system containing information about cities and towns in a country with thousands of cities and towns, and the database program should allow users to find information about a particular place by name (another example of an exact-match query). Users should also be able to find all places that match a particular value or range of values for attributes such as location or population size. This is known as a range query.

A reasonable database system must answer queries quickly enough to satisfy the patience of a typical user. For an exact-match query, a few

seconds are satisfactory. If the database is meant to support range queries that can return many cities that match the query specification, the entire operation may be allowed to take longer, perhaps on the order of a minute. To meet this requirement, it will be necessary to support operations that process range queries efficiently by processing all cities in the range as a batch, rather than as a series of operations on individual cities.

The hash table suggested in the previous example is inappropriate for implementing our city database, because it cannot perform efficient range queries. A simple linear index, similar to array data structure, would be more appropriate if the database was created once, and then never changed, such as an atlas distributed on a CD-ROM.

Post-reading Activities

I. Reading comprehension

Directions: Mark each statement as T (True), F (False), or NG (Not Given) to the information in the reading comprehension passage.

----1. No difference is stated in the passage in regard to numeric and nonnumeric data.

----2. Processing data in floating point format is more straightforward than the integer format.

----3. The main purpose of most computers is to store and retrieve data quickly.

----4. B-trees data structure is suitable for sorting data in ascending and descending order.

---5. There are many issues to consider for selecting a data structure in an application program.

---6. The aim of linked list is to help programmers analyze the use of memory space.

---7. There is always one data structure, which fits all different situations.

---8. Banks provide two kinds of ATM and digital services for their customers.

Questions *9-15*: Choose the appropriate letter *A-C.*

9. Which one of the following ideas is NOT true according to the reading comprehension passage?

A. All digital data are stored in the computer 0s and 1s, sequentially.

B. All data stored in a computer are either numeric or nonnumeric.

C. The main purpose of most computer programs is to perform calculations.

10. One of the following options is the most important data structure.

A. A set

B. A hashing

C. A stack

11. All of the following ideas are true in regard to hashing data structure EXCEPT…..

A. an associative array.

B. storing information.

C. code fragments.

12. The difference between superior and inferior algorithm is that ...

A. The former one is always active and live.

B. The latter one is already remembered and used.

C. There is usually no difference between the two.

13. Which one of the following banking data structures takes customers' time more?

A. Withdrawing money during a day.

B. Making a deposit in person.

C. Opening and closing an account.

14. One of the following is NOT a main concern for considering the choice of data structures.

A. Cost of deletion

B. Efficient search

C. Resource constraints

15. The reason stated in the passage in regard to appropriateness of our city database is .....

A. lack of inefficient range queries.

B. lack of efficient range queries.

C. lack of exact-match queries.

II. Vocabulary activities

Directions: Read each sentence on inputs and outputs of a computer stated below. Circle the one word or phrase in parentheses ( ) that has the same meaning as the underlined word in the sentence. Compare your answers with a partner.

1. Data are either numeric or nonnumeric. Numeric data can be in integer (*insufficient/complete/related*) or in floating point type.
2. A linked list is a data structure with successive (*in a row/ associative/ irrelevant*) elements connected via a pointer.
3. Stack data structures are suitable for implementing recursive (*repetitive/backward/enough*) algorithms, and defining local variables.
4. Data structures are generally based on the ability of a computer to randomly fetch and store data at any place in its memory, specified by an address—a bit string that can be itself stored in memory and manipulated (*written/brought/controlled*) by the program.
5. If one data structure or algorithm is superior (*better/ worse/improved*) to another in all respects, the inferior one will usually have long been forgotten.
6. Each problem has constraints (*availabilities/limits/chances*) on available space and time.
7. Each solution to a problem makes use of the basic operations in some relative proportion, and the data structure selection process must account for (*describe/intend/increase*) this.
8. Deleting an account does not have time constraint, because from the customer's point of view that all the money be returned - equivalent to a withdrawal (*open an account/ close an account/ taking out money*).

9. A reasonable (*logical/enhanced/expanded*) database system must answer queries quickly enough to satisfy the patience of a typical user.
10. A simple linear index, similar to array data structure, would be more appropriate if the database is created once, and then never changed, such as an atlas distributed (*spread/limited/taken*) on a CD-ROM.

III. Writing development activities

Directions: List the differences between human teller and ATM and write down three points of differences you can think of. From this list, select a point, which you think is the most important. Then, write a topic sentence showing that you will describe these differences through sentences, and use verbal structures, sentence connectors, and conjunctions to support ideas. Work with a partner or in groups and show your topic sentence to your classmates to see if it is suitable for the differences which you plan to write about.

List of three points:

1.----------

2.----------

3.----------

The most important point is ----------

Topic sentence is ----------
