Data Structures and Algorithms (1)

Instructor: Ming Zhang

Textbook Authors: Ming Zhang, Tengjiao Wang and Haiyan Zhao

Higher Education Press, 2008.6 (the "Eleventh Five-Year" national planning textbook)

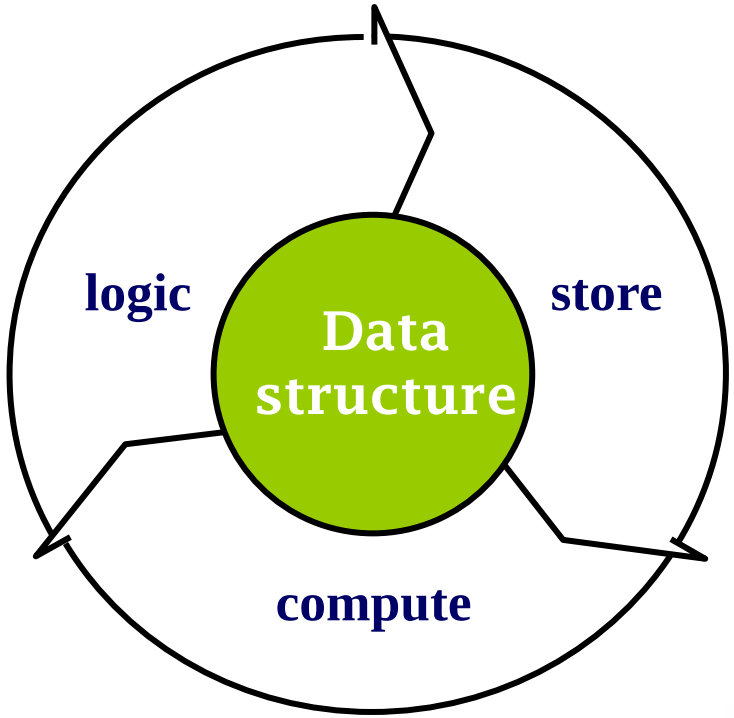
https://courses.edx.org/courses/PekingX/04830050x/2T2014/

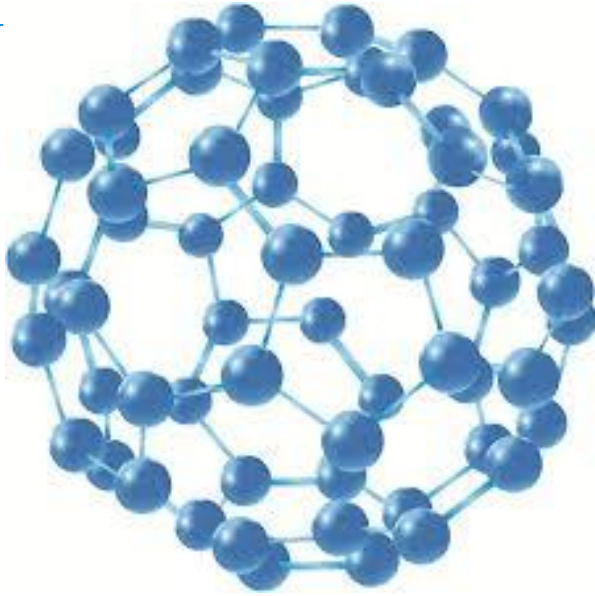
Chapter 1 Overview

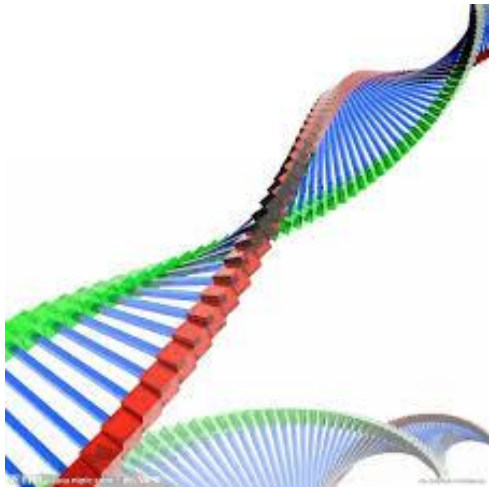
- Problem solving
- Data structures and abstract data types
- The properties and categories of algorithms
- Evaluating the efficiency of the algorithms

1.2 What is data structure

- **Structure: entity + relation**
- **Data structure** :
  - Data organized according to logical relationship
  - Stored in computer according to a certain storage method
  - A set of operations are defined on these data



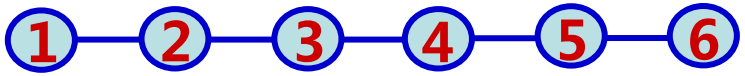




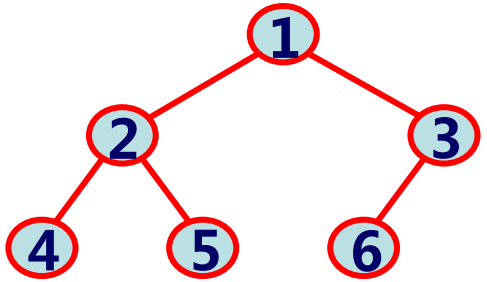
1.2 What is data structure

Logical organization of data structure

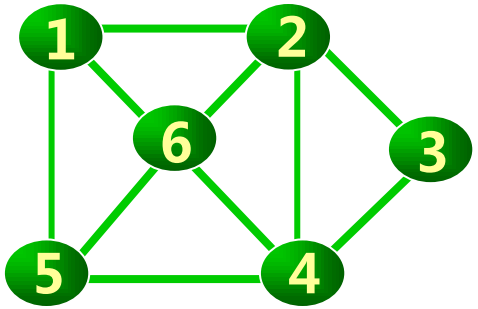
- **Linear Structure**



  - Linear lists (list , stack , queue , string, etc.)
- **Nonlinear Structure**
  - Trees (binary tree , Huffman tree , binary search tree etc)



  - Graphs (directed graph , undirected graph etc)



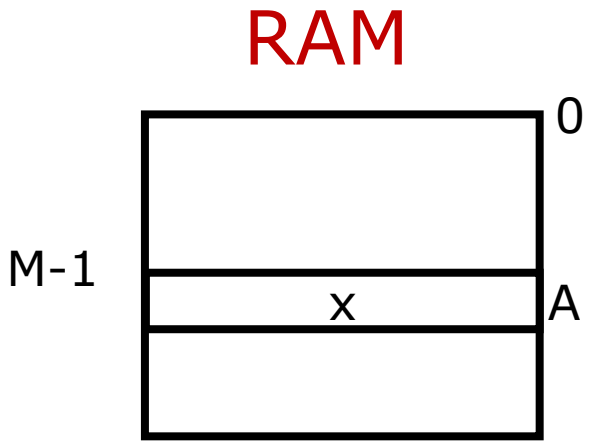
- **Graph ⊇ tree ⊇ binary tree ⊇ linear list**

Storage structure of data

- **Mapping from logical structure to the physical storage space**

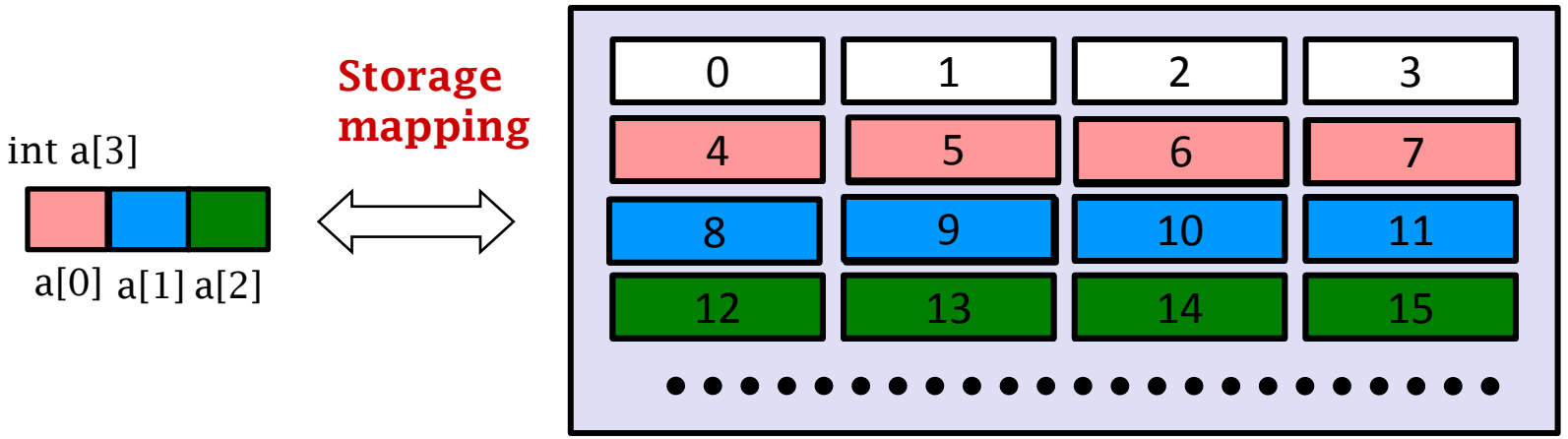
Main memory (RAM)

- Coded in non negative integer address , set of adjacent unit
- The basic unit is the byte
- The time required to access different addresses are basically the same (random access)



Storage structure of data

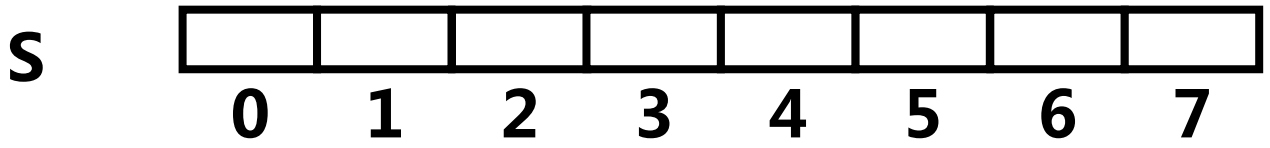
- For logical structure (K , r) , in which r∈R
  - For the node set K, establish a mapping from K to M memory unit : K→M , for every node j∈K , it corresponds to a unique continuous storage area C in M



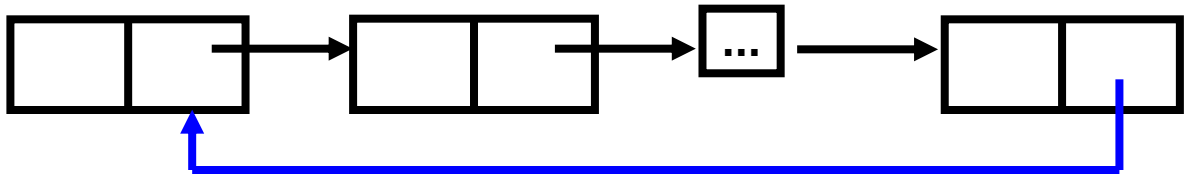
Main memory

Storage structure of data

- Relation tuple $(j_1 , j_2) \in r$
  $(j_1 , j_2 \in K$ are nodes)
  - Sequence : storage units of data are adjacent



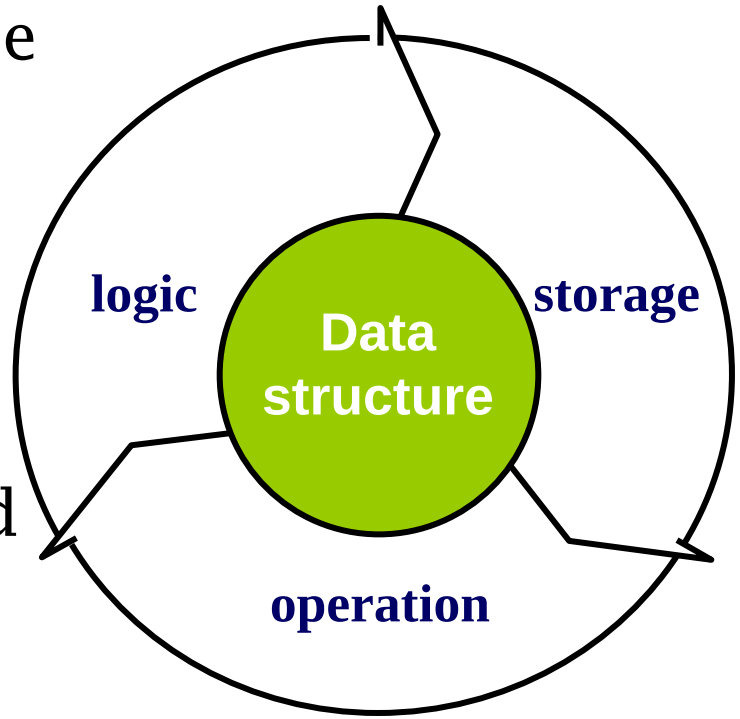
  - Link: a pointer points to the storage address, referring to a certain connection



- Four kinds : Sequence, link, index, hash

Abstract Data Type

- Abbreviated as **ADT** (Abstract Data Type)
  - A set of operations built upon a mathematical model
  - Has nothing to do with the physical storage structure
  - The software system is built upon the data model (object oriented)
- The development of **Modularization**
  - Hide the details of the implementation and operations of the internal data structures
  - Software reuse



1.2 What is data structure

ADT do not care about storage details

——for example , brackets matching algorithm of C++ version

```
void BracketMatch(char *str) {
  Stack<char> S; int i; char ch;
// The stack can be sequential
// or linked, both are referenced
// in the same way
  for(i=0; str[i]!='\0'; i++)  {
      switch(str[i])  {
        case '(':   case '[':   case '{':
              S.Push(str[i]); break;
        case ')':  case ']':   case '}':
          if (S.IsEmpty( )) {
           cout<<"Right brackets
excess!";
              return;
          }
          else {
          ch = S.GetTop( );
          if (Match(ch,str[i]))
              ch = S.Pop( );
          else {
              cout << " Brackets do not match!";
              return;
           }
        } /*else*/
      }/*switch*/
    }/*for*/
    if (S.IsEmpty( ))
        cout<<" Brackets match!";
    else cout<<“Left brackets
excess!";
    }
```

1.2 What is data structure

Sequential stack brackets matching algorithm of C version (different from the linked stack)

```
void BracketMatch(char *str) {
  SeqStack S; int i; char ch;
  InitStack(&S);
  for(i=0; str[i]!='\0'; i++)  {
      switch(str[i])  {
       case '(':    case '[':    case '{':
            Push(&S,str[i]); break;
       case ')':  case ']':    case '}':
            if (IsEmpty(&S)) {
                printf("\nRight brackets
excess!");
                return;
            }
            else {
                GetTop (&S,&ch);
                if (Match(ch,str[i]))
                    Pop(&S,&ch);
                else {
                    printf("\nBrackets don't
match!");
                    return;
                }
               } /*else*/
             }/*switch*/
           }/*for*/
           if (IsEmpty(&S))
               printf("\nBrackets match!")
           else printf("\nLeft brackets
excess" );}
```

1.2 What is data structure

Linked stack brackets matching algorithm of C version (different from the sequential stack)

```
void BracketMatch(char *str) {
  LinkStack S; int i; char ch;
  InitStack(/*&*/S);
  for(i=0; str[i]!='\0'; i++)  {
      switch(str[i])  {
       case '(':    case '[':    case '{':
            Push(/*&*/S, str[i]);
            break;
       case ')':  case ']':    case '}':
            if (IsEmpty(S)) {
                printf("\nRight brackets
excess!");
                return;
            }
            else {
                GetTop (/*&*/S,&ch);
                if (Match(ch,str[i]))
                    Pop(/*&*/S,&ch);
                else {
                    printf("\nBrackets don't
match!");
                    return;
                    }
                } /*else*/
            }/*switch*/
        }/*for*/
        if (IsEmpty(/*&*/S))
            printf("\nBrackets match!")
        else printf("\nLeft brackets excess");}
```

Abstract Data Type

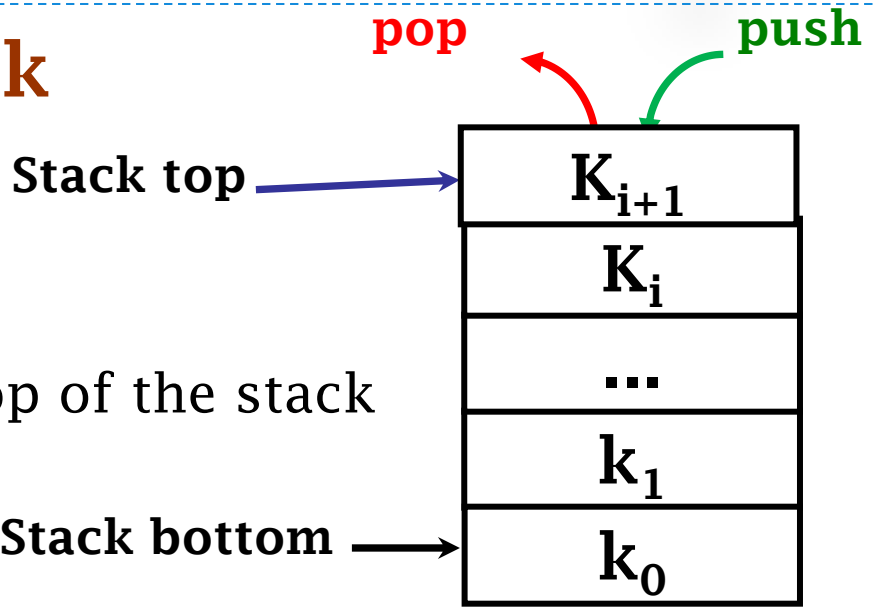
- Two-tuples of abstract data structure

  <Data object D , data operation P>

- Firstly, defines logical structure; then data operations
  - **Logical structure** : relationship between data objects
  - **Operations** : algorithms running on the data

1.2 What is data structure

Example : abstract data type of stack



- Logical structure : linear list
- Operation : Restricted access
  - Only allow for the insert, delete operation at the top of the stack
  - push、pop、top、isEmpty

```
template <class T>          // Element type  of stack is T
class Stack {
public:                     // Stack operation set
   void clear();            // Turned into an empty stack
   bool push(const T item);// Push item into the stack , return true if succeed, otherwise false
   bool pop(T & item);// Pop item out of the stack ,  return true if succeed, otherwise false
   bool top(T& item); // Read item at the top of the stack, return true if succeed, otherwise false
   bool isEmpty(;      // If the stack is empty return true
   bool isFull();      // If the stack is full return true
};
```

Questions about abstract data type

- How to present a logical structure in an ADT ?
- Is abstract data type equivalent to the class definition ?
- Can you define a ADT without templates ?

Data Structures and Algorithms

Thanks

the National Elaborate Course (Only available for IPs in China)
http://www.jpk.pku.edu.cn/pkujpk/course/sjjg/

Ming Zhang, Tengjiao Wang and Haiyan Zhao
Higher Education Press, 2008.6 (awarded as the "Eleventh Five-Year" national planning textbook)