A Familiar yet Vague Term: “Abstract Data Type”

ADT $\triangleq$ data + operation

C++ Object Oriented Programming

Pei-yih Ting

NTOU CS

Abstract Data Type 抽象資料型態

- ✧ **Abstract**?!

  和實際的東西有距離的

  - ✶ Disassociated from any specific instance 抽象的, 不具體的
  - ✶ Expressing a quality apart from an object 抽象化 (理論化)
  - ✶ Having only intrinsic form with little attempt at pictorial representation or narrative content 摘要、重點

- ✧ **Data type**?

  characteristics of a set of data,

  template for instances of data storage

  specifies: { format
ranges
memory resources }

Abstract Data Type (cont’d)

✧ See what people on Internet said

何謂ADT(Abstract data type)
我一直搞不懂ADT是啥?
抽象資料型態(ADT)
我知道是一個自訂的資料型態,
但是卻似懂非懂,
可以幫忙解釋一下嗎?
感謝...

~~簡單的說陣列 (array) 就是一種抽象的觀念,~~
~~但是你做出了 int array[10]; 這樣的實踐, 就是抽象觀念的實作...~~

Any better?!

Abstract Data Type (cont’d)

- http://en.wikipedia.org/wiki/Abstract_data_type

- In computing, an abstract data type (ADT) is a specification of **a set of data** and **the set of operations** that can be performed on the data.
- e.g. container, deque, list, map, multimap, multiset, priority queue, queue, set, stack, string, tree, heap

- Such a data type is ***abstract*** in the sense that it is independent of various concrete implementations.
  - Question: Are they still abstract without specifying the set of operations (only the set of data)??

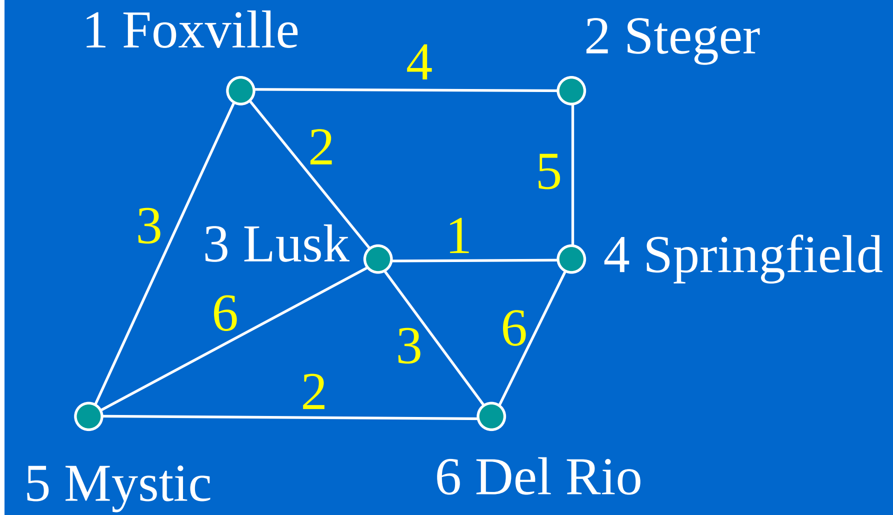
Abstract Data Type (cont’d)

- ✧ Are you really satisfying with this definition???
  - ✶ “Data type” is an easy idea: the attributes
  - ✶ It looks like that “data type” itself could also be independent of various implementations.
  - ✶ Why are the additional “operations” related to the keyword “abstract”???

Minimal Spanning Tree (1/4)

✧ JohnsonBaugh’s *Algorithms*, Section 7.3 (page 284) find Minimal Spanning Tree (MST) with **Prim’s algorithm**:

Six cities

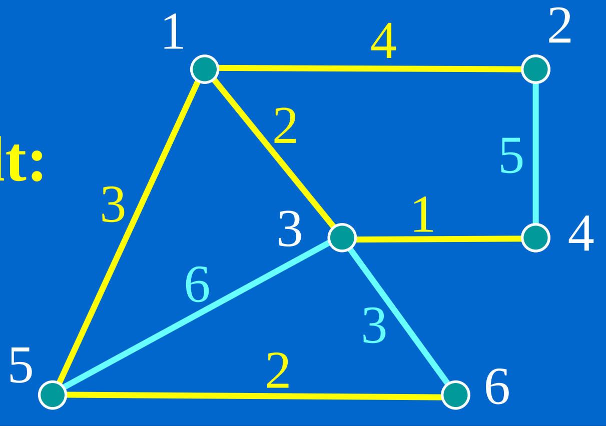


We want to construct a set of interconnecting roads such that one can reach any city from any starting city and the **total construction costs are minimized**.

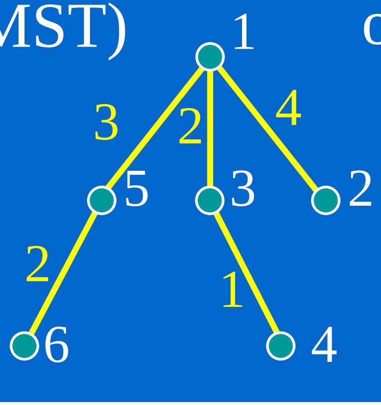
The estimated costs for some pairs of cities are as labeled.

Result:

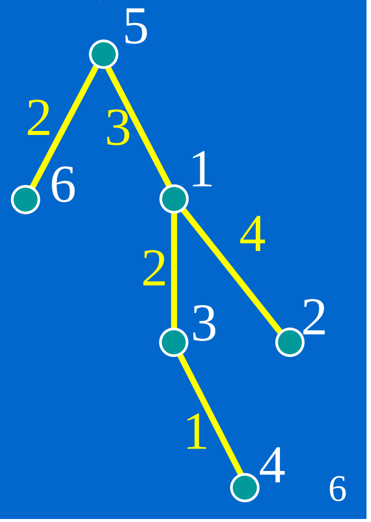
Best



A tree (MST)

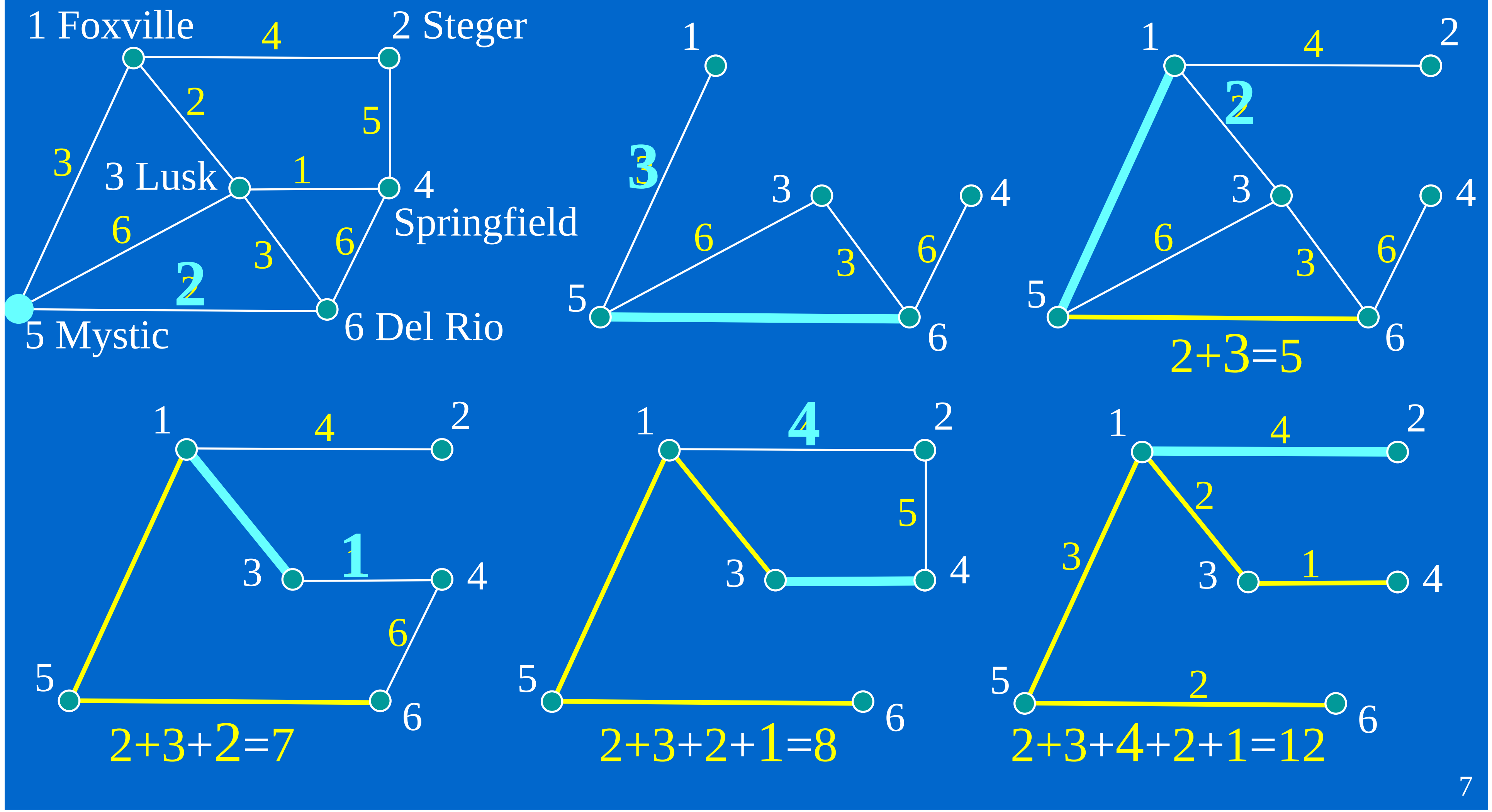


or



Prim’s MST (2/4)

✧ **Prim’s algorithm**: starting with vertex **5** (Mystic)



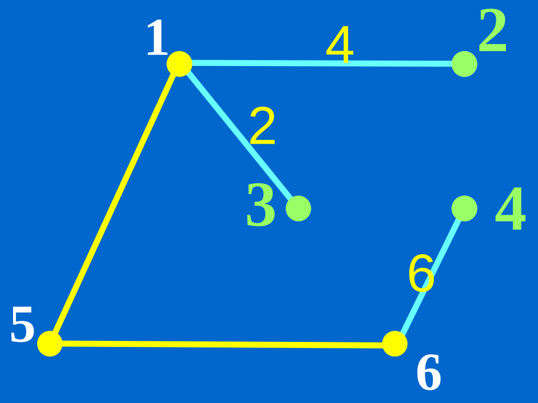
Prim's MST (3/4)

h: a list of vertices ***v*** not in the MST and its minimum weight to MST (weight of the edge from ***v*** to the vertex ***parent*[*v*]**)

***parent*[*v*]**: (*v*, *parent*[*v*]) is the edge with minimum weight

h

v	minimum weight from *v* to **MST**	***parent*[*v*]**
2	**4**	1
3	**2**	1
4	**6**	6



MST={1,5,6}

Prim's MST (4/4)

```
prim(adj, start, parent) {
    n = adj.last
    for i = 1 to n
        key[i] = ∞
    key[start] = 0
    parent[start] = 0
    h.init(key, n)
    for i = 1 to n {
        v = h.del()
        ref = adj[v]
```

v=1

ref={5,3,2}

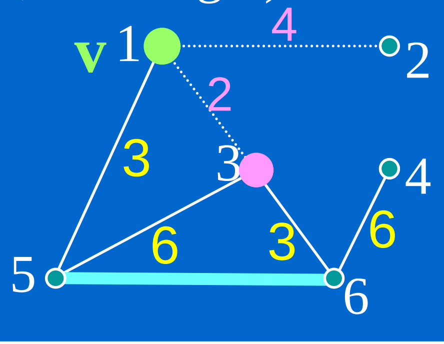
w

```
        while (ref != null) {
            w = ref.ver
            if (h.isin(w) &&
                ref.weight < h.keyval(w)) {
                parent[w] = v
                h.decrease(w, ref.weight)
            }
            ref = ref.next
        }
    }
}
```

w=3, w∉ MST

ref.weight=2

h.keyval(w)=3



h is an **abstract data type** that supports the following operations

h.**init**(key, n): initializes h to the values in key

h.**del**(): deletes the item in h with the smallest weight and returns the vertex

h.**isin**(w): returns true if vertex w is in h

h.**keyval**(w): returns the weight corresponding to vertex w

h.**decrease**(w, new_weight): changes the weight of w to new_weight (smaller)

Abstract Painting

✧ Picasso

Miro - Angel

抽象畫 - **非寫實** 畫風

~~看不懂的畫~~ => 畫家眼中覺得重要的描述

Abstract

- ✧ Mathematic formula: Central Limit Theorem, Stirling formula, Fourier Transform, …
- ✧ Physic formula: Newton’s law, wave equation, …

It is quite likely that you cannot understand the meaning of these formula because they are abstracted out from their original application environments.

Thus, you say that these formula are quite abstract.

Abstraction

- **Abstraction**: the process or result of generalization by reducing the information content of a concept or an observable phenomenon
  - A method to find general form of an idea
  - A method to find a unified explanation
  - A method to simplify the complex exteriors.
  - 抽象化 – 單純化 – 簡化
  - ex. 鳥可以飛, 飛機可以飛, 蚊子可以飛 → 有翅膀的

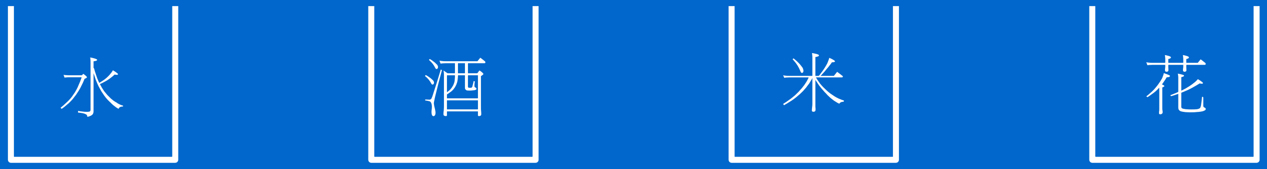
    but 鴕鳥, 肉雞…

  需要描述翅膀怎麼用才能飛 – 需要有**操作型定義**

  一個資料結構真正代表的意義 – 必需用這個資料結構所支援的動作來描述/限定

Data vs. Operation

- 杯子 ..... pure data



水 酒 米 花

- Data storage can be used for any imaginable purpose.
- You want your data storage to be specific. You specify its “operations”
  - How do you use this data?
  - For what do you use it?

Back to ADT

- ✧ abstract data type (ADT):

  is a specification of

  { **a set of data** and

  the set of operations performed on the data.

抽象的資料型態?

or

精確表達由同類型物件抽象化出來的共通特性的資料型態?

- ✧ It is independent of various implementations
- ✧ It provides specific descriptions of the functionalities of a piece of data in terms of operations abstracted from many similar objects.

The C syntax: x.y vs. x.z()

- In C, how do you capture the idea of

  h.key and **h.decrease(w, weight)**
- Are these two syntactically correct in C?
- Yes.
- decrease is called a "function pointer"
- It is a piece of data (attribute), and at the same time, you can invoke a function via this data.
  - e.g.

```
void fun(int x)
{
    …
}
```

```
void (*fp)(int);
 …
fp = fun;
(*fp)(5);  /* calling fun(5) */
```

```
// cl testfp.c
#include <stdio.h>

struct MyStruct
{
    int data;
    int (*fp)(int, struct MyStruct *);
};

int isEqual(int, struct MyStruct *);

void main()
{
    struct MyStruct obj = {123, isEqual};
    int data;
    int (*myfp)(int, struct MyStruct *) = isEqual;

    printf("Please input an integer: ");
    scanf("%d", &data);
    printf("%d\n", obj.fp(data, &obj));
    printf("%d\n", (*obj.fp)(data, &obj));
    printf("%d\n", myfp(data, &obj));
    printf("%d\n", (*myfp)(data, &obj));
    printf("%d\n", isEqual(data, &obj));
}

int isEqual(int data,
            struct MyStruct *self)
{
    printf(" calling isEqual() ");
    if (data == self->data)
        return 1;
    else
        return 0;
}
```