Priority Queues, Maps, Sets

Κ08 Δομές Δεδομένων και Τεχνικές Προγραμματισμού

Κώστας Χατζηκοκολάκης

ADTs for efficient search

- So far we've seen ADTs in which elements have a **predefined order**
  - Depending only on the insertion order, not on the **values**
- And search is inefficient
- We will now discuss ADTs that take the **order** of elements into account
  - **Any** order can be used (many options, even for the same data)

The compare function

- A `compare` function determines the order of elements
  - `compare(a, b) < 0` iff `a` is “smaller” than `b`
  - `compare(a, b) > 0` iff `a` is “greater” than `b`
  - `compare(a, b) == 0` iff `a` and `b` are **equivalent** (**not** equal!)
- Examples for integers:
  - `return *(int*)a - *(int*)b`
  - `return abs(*(int*)a) - abs(*(int*)b)`
  - `return a - b`
  - Which elements are equivalent?
- Examples for strings, structs, Vectors, etc?

Priority Queue

- Similar to a Queue or Stack
  - No iteration, only a specific item can be accessed/removed
- **BUT**: the item that can be accessed/removed is the **maximum** one
  - with respect to a given order (`compare`)
- We can insert elements in any order
- When we remove, we always get the maximum element!

Create, insert

```
// Δημιουργεί και επιστρέφει μια νέα ουρά προτεραιότητας, της οποίας
// Αν destroy_value != NULL, τότε καλείται destroy_value(value) κάθε
// Αν values != NULL, τότε η ουρά αρχικοποιείται με τα στοιχεία του V

PriorityQueue pqueue_create(CompareFunc compare, DestroyFunc destroy_

// Προσθέτει την τιμή value στην ουρά pqueue.

void pqueue_insert(PriorityQueue pqueue, Pointer value);
```

- The `compare` function must be given to `pqueue_create`
  - Cannot be modified.
- We can initialize the queue with a Vector of `values`
  - might be **faster** than insterting them one by one!

Accessing the maximum element

```c
// Επιστρέφει το μεγαλύτερο στοιχείο της ουράς
Pointer pqueue_max(PriorityQueue pqueue);

// Αφαιρεί την μεγαλύτερη τιμή της ουράς
void pqueue_remove_max(PriorityQueue pqueue);
```

- Guaranteed to work independently from the insertion order
- Efficient, even after removing elements!

When to use Priority Queues

- When we need to quickly find the maximum value
  - on data that is updated
- Many applications
  - Shortest Path in a graph (Dijkstra)
  - Minimum Spanning Tree (Prim)
  - Search in Artificial Intelligence (A*)
  - Sorting elements (Heapsort)
  - Load balancing algorithms
  - Compression (Huffman codes)
  - …

ADTMap

- Associates a `key` to a `value`
- Allows to search for `value` given `key` (or a key **equivalent** to it)
- Can be thought as an “array” with non-integer indexes!

```
// Αυτό είναι συνηθισμένο
names[19] = "Bob";
...
printf("The student with id 19 is %s\n", names[19]);

// Ωραία θα ήταν να μπορούσαμε να κάνουμε το αντίστροφο (αλλά δε γίνε
ids["Bob"] = 19;
...
printf("The id of Bob is %d\n", ids["Bob"]);
```

- We can use a Map `ids` that associates `"Bob"` to `19`

a.k.a. Symbol table, Dictionary, Associative array, Unordered map

Create

```
// Δημιουργεί και επιστρέφει ένα map, στο οποίο τα στοιχεία συγκρίνον
// τη συνάρτηση compare.
// Αν destroy_key ή/και destroy_value != NULL, τότε καλείται destroy_
// ή/και destroy_value(value) κάθε φορά που αφαιρείται ένα στοιχείο.

Map map_create(CompareFunc compare, DestroyFunc destroy_key, DestroyF
```

- We need to pass a `compare` function
  - the Map needs to know which keys are equivalent to `"Bob"`
- Separate destroy functions for keys and values

Insert, find

```c
// Προσθέτει το κλειδί key με τιμή value. Αν υπάρχει κλειδί ισοδύναμο
// παλιά key & value αντικαθίστανται από τα νέα.

void map_insert(Map map, Pointer key, Pointer value);

// Επιστρέφει την τιμή που έχει αντιστοιχιστεί στο συγκεκριμένο key,
// το key δεν υπάρχει στο map.

Pointer map_find(Map map, Pointer key);
```

Example

```
// name => id
ADTMap ids = map_create(compare_strings, NULL, NULL);

// ids["Bob"] = 19
int id = 19;
map_insert(ids, "Bob", &id);

...

// Find ids["Bob"]
printf("The id of Bob is %d\n", *(int*)map_find(ids, "Bob"));
```

Remove

```
// Αφαιρεί το κλειδί που είναι ισοδύναμο με key από το map, αν υπάρχε
// Επιστρέφει true αν βρέθηκε τέτοιο κλειδί, διαφορετικά false.

bool map_remove(Map map, Pointer key);
```

Example:

```
map_remove(ids, "Bob");

...

if(map_find(ids, "Bob") == NULL) {
    printf("No student named Bob exists");
}
```

Iteration

```c
// Μέσω κόμβων

for(MapNode node = map_first(ids);           // ξενικάμε από τον πρώτο
    node != MAP_EOF;                         // μέχρι να φτάσουμε στο
    node = map_next(ids, node)) {            // μετάβαση στον επόμενο

    char* name = map_node_key(ids, node);    // το κλειδί του συγκεκρι
    int* id = map_node_value(ids, node);     // η τιμή του συγκεκριμέν
    printf("%s's id is %d\n", name, *id);
}
```

When to use Map

- When we need equality search
  - One of the most commonly used operations
  - And the data is updated frequently
- Check if some value is already seen
  - eg. memoization
- Provided natively by many programming languages
  - but we need to understand how it works!

ADTSet

- An **ordered** set of **unique** values
- Allows to search for a `value` efficiently
- Allows to iterate the set in the correct **order**

Create

```
// Δημιουργεί και επιστρέφει ένα σύνολο, στο οποίο τα στοιχεία συγκρί
// βάση τη συνάρτηση compare.
// Αν destroy_value != NULL, τότε καλείται destroy_value(value) κάθε
// αφαιρείται ένα στοιχείο.

Set set_create(CompareFunc compare, DestroyFunc destroy_value);
```

We need to pass a `compare` function.

The Set maintains the **order** of the values.

Insert, find

```c
// Προσθέτει την τιμή value στο σύνολο, αντικαθιστώντας τυχόν προηγού
// ισοδύναμη της value.

void set_insert(Set set, Pointer value);

// Επιστρέφει την μοναδική τιμή του set που είναι ισοδύναμη με value,
// δεν υπάρχει

Pointer set_find(Set set, Pointer value);
```

Example

```
ADTSet names = set_create(compare_strings, NULL);

set_insert(names, "Alice");
set_insert(names, "Bob");

...

printf("Is Bob present? %d\n", set_find(names, "Bob") != NULL);
```

Remove

```
// Αφαιρεί τη μοναδική τιμή ισοδύναμη της value από το σύνολο, αν υπά
// Επιστρέφει true αν βρέθηκε η τιμή αυτή, false διαφορετικά.

bool set_remove(Set set, Pointer value);
```

Example:

```
set_remove(names, "Bob");

...

if(set_find(students, "Bob") == NULL) {
    printf("No student named Bob exists");
}
```

Iteration

```
// Μέσω κόμβων, στη σειρά διάταξης!

for(SetNode node = set_first(ids);          // ξενικάμε από τον πρώτο
    node != SET_EOF;                        // μέχρι να φτάσουμε στο
    node = set_next(ids, node)) {           // μετάβαση στον επόμενο

    char* name = set_node_value(ids, node); // Η τιμή του συγκεκριμέν
    printf("%s\n", name, *id);
}
```

When to use Set

- When we need to access values in a certain **order**
  - And the data is updated frequently
- When we need **range** search
  - One of the most commonly used operations
- But also equality search