Διάλεξη 21η: Απλά Συνδεδεμένες Λίστες

Τμήμα Επιστήμης Υπολογιστών, Πανεπιστήμιο Κρήτης

Εισαγωγή στην Επιστήμη Υπολογιστών

Δομές δεδομένων

- Ορισμός:
  - Μια συλλογή δεδομένων που είναι οργανωμένα με συγκεκριμένο τρόπο
  - Ένα σύνολο τελεστών που ενεργούν πάνω στη συλλογή
  - Αλγόριθμοι που υλοποιούν τους τελεστές
- Παράδειγμα: Λίστα
  - Οργάνωση: κάθε στοιχείο δείχνει στο επόμενο ή στο `NULL`
  - Τελεστές: προσθήκη στην αρχή/ενδιάμεσα/στο τέλος, αφαίρεση ενός στοιχείου, εύρεση του n-οστού στοιχείου, συνένωση λιστών, κλπ
  - Αλγόριθμοι: `insert`, `remove`, `nth`, κλπ

Δομές δεδομένων (2)

- Άλλα παραδείγματα:
  - Ταξινομημένη λίστα
    - Οργάνωση: κάθε στοιχείο δείχνει στο επόμενο που είναι μικρότερο (φθίνουσα) ή μεγαλύτερο (αύξουσα) ή στο `NULL`
    - Τελεστές: προσθήκη στοιχείου, αφαίρεση στοιχείου, εύρεση στοιχείου, συνένωση ταξινομημένων λιστών, κλπ
    - Αλγόριθμοι: `insert` (στη σωστή θέση), `remove`, `ismember`, κλπ
  - Δέντρο
    - Οργάνωση: κάθε στοιχείο δείχνει στα “παιδιά” του (ή στο `NULL`)
    - Τελεστές: προσθήκη στοιχείου σε κλαδί, εύρεση ενός στοιχείου, αφαίρεση ενός στοιχείου, συνένωση δέντρων, κλπ
    - Αλγόριθμοι: `insert`, `remove`, `find`, κλπ
  - Άλλα: δυαδικό δέντρο, ισορροπημένο δέντρο, διπλά συνδεδεμένη λίστα, κυκλική λίστα, γράφος, υπεργράφος, στοίβα, ουρά, διπλή ουρά, union-find, skip-λίστα, hash table, αραιός πίνακας, κλπ

Απλά συνδεδεμένη λίστα

- Οργάνωση των δεδομένων
  - Θεωρούμε τα δεδομένα διατεταγμένα γραμμικά (το ένα μετά το άλλο)
  - Κάθε στοιχείο δείχνει στο επόμενο στη μνήμη ή στο `NULL` αν είναι το τελευταίο
- Παράδειγμα δήλωσης (λίστα ακεραίων)

intlist.c

```
struct list_node {
  int data;
  struct list_node *next;
};
typedef struct list_node node_t;
node_t *root;
```

- Αναδρομικός ορισμός
  - Ο δείκτης `root` δείχνει σε (άδεια) λίστα αν είναι `NULL`
  - Ο δείκτης `root` δείχνει σε λίστα αν δείχνει σε ένα στοιχείο `struct list_node` και το `root->next` δείχνει σε λίστα

Απλά συνδεδεμένη λίστα (2)

- Τελεστές
  - Δημιουργία κενής λίστας
  - Εισαγωγή στοιχείου πριν/μετά από άλλο στοιχείο
  - Εύρεση συγκεκριμένου στοιχείου
  - Διαγραφή στοιχείου
  - Διάσχιση της λίστας
  - Απελευθέρωση/διαγραφή λίστας

Απλά συνδεδεμένη λίστα (3)

- Αλγόριθμοι
  - Δημιουργία κενής λίστας

    Ο κενός δείκτης `NULL` αναπαριστά την κενή λίστα
  - Εισαγωγή στοιχείου πριν/μετά από άλλο στοιχείο

```
node_t *list_insert_before(node_t *list, int data);
void list_insert_after(node_t *position, int data);
```

  - Εύρεση συγκεκριμένου στοιχείου

```
node_t *list_find(node_t *list, int data);
node_t *list_nth(node_t *list, int position);
```

  - Διαγραφή στοιχείου

```
node_t *list_remove_node(node_t *list, node_t *node);
node_t *list_remove_data(node_t *list, int data);
```

  - Διάσχιση της λίστας

```
void list_traverse(node_t *list, void (*f)(node_t *node));
void list_print(node_t *list);
```

  - Απελευθέρωση/διαγραφή λίστας

```
void list_delete(node_t *list);
```

Απλά συνδεδεμένη λίστα (4)

- Εισαγωγή στοιχείου πριν/μετά από άλλο στοιχείο

intlist.c

```
node_t *list_insert_before(node_t *list, int data)
{
  node_t *node;

  node = (node_t *) malloc(sizeof(node_t));
  node->data = data;
  node->next = list;
  return node;
}

void list_insert_after(node_t *position, int data)
{
  node_t *node;

  node = list_insert_before(position->next);
  position->next = node;
}
```

Απλά συνδεδεμένη λίστα (5)

- Εύρεση συγκεκριμένου στοιχείου

intlist.c

```
node_t *list_find(node_t *list, int data)
{
  while((list != NULL) && (list->data != data)) {
    list  = list->next;
  }
  return list;
}

node_t *list_nth(node_t *list, int position)
{
  int i  = 0;
  while((list != NULL) && (i < position)) {
    list  = list->next;
    i++;
  }
  return list;
}
```

Απλά συνδεδεμένη λίστα (6)

- Διαγραφή στοιχείου

```
node_t *list_remove_node(node_t *list, node_t *node)
{
  if(list == NULL) {
    return NULL;
  }
  if(list == node) {
    list = list->next;
    free(node);
    return list;
  }
  list->next = list_remove_node(list->next, node);
  return list;
}
```

Απλά συνδεδεμένη λίστα (7)

- Διαγραφή στοιχείου (εναλλακτικά)

intlist.c

```
node_t *list_remove_node(node_t *list, node_t *node)
{
  while((list != NULL) && (list->next != node)) {
     list  = list->next;
  }
  if( list  != NULL) {
     list->next = node->next;
    free(node);
  }
  return list;
}
```