Δομές Δεδομένων και Αλγόριθμοι (Γ' εξάμηνο)

Τμήμα Μηχανικών Πληροφορικής ΤΕ, ΤΕΙ Ηπείρου

Γκόγκος Χρήστος

25/11/2014

Άσκηση εργαστηρίου #4 (Σωροί – Heaps και Heapsort)

Στην εργασία αυτή υλοποιείται η δομή του Σωρού (Heap) και η ταξινόμηση HeapSort.

```
/*
 *  exercise04.cpp
 *  Christos Gogos
 *  Technolological Educational Institute of Epirus (2014)
 */

#include <iostream>

using namespace std;

const int static MAX_HEAP_SIZE = 1000;
int heap_size = 0;

template<class T>
void heapify(T* heap, int k) {
	int v = heap[k];
	bool flag = false;
	while (!flag && 2 * k <= heap_size) {
		int j = 2 * k;
		if (j < heap_size)
			if (heap[j] < heap[j + 1])
				j++;
		if (v >= heap[j])
			flag = true;
		else {
			heap[k] = heap[j];
			k = j;
		}
	}
	heap[k] = v;
}

template<class T>
T* heap_bottom_up(T* a, int N) {
	T* heap = new T[MAX_HEAP_SIZE]();
	heap_size = N;
	heap[0] = NULL;
	for (int i = 0; i < N; i++)
		heap[i + 1] = a[i];
	for (int i = heap_size / 2; i >= 1; i--)
		heapify(heap, i);
	return heap;
}

template<class T>
void insert_key(T* h, T key) {
	heap_size++;
	h[heap_size] = key;
	int pos = heap_size;
	while (pos != 1 && h[pos / 2] < h[pos]) {
```

```
            swap(h[pos / 2], h[pos]);
            pos = pos / 2;
      }
}

template<class T>
void maximum_key_deletion(T* heap) {
      swap(heap[1], heap[heap_size]);
      heap_size--;
      heapify(heap, 1);
}

template<class T>
T* heap_sort(T* a, int N) {
      T* heap = heap_bottom_up(a, N);
      for (int i = 1; i < N; i++)
      maximum_key_deletion(heap);
      return heap;
}

// utility method
template<class T>
void print_heap(T* heap) {
      printf("HEAP(%d) :[", heap_size);
      for (int i = 1; i <= heap_size; i++) {
            if (i == heap_size)
            printf("%d", heap[i]);
            else
            printf("%d ", heap[i]);
      }
      printf("]");
}

// tests
void test_bottom_up_heap_construction() {
      int N = 10;
      int a[N] = {42, 37, 31, 16, 53, 19, 47, 58, 33, 25};
      int* heap = heap_bottom_up(a, N);
      print_heap(heap);
      delete[] heap;
}

void test_maximum_key_deletion() {
      int N = 10;
      int a[N] = {42, 37, 31, 16, 53, 19, 47, 58, 33, 25};
      int* heap = heap_bottom_up(a, N);
      print_heap(heap);
      printf("\n");
      while (heap_size > 0) {
            printf("key %d deleted ==> ", heap[1]);
            maximum_key_deletion(heap);
            print_heap(heap);
            printf("\n");
      }
      delete[] heap;
}

void test_single_key_insertions() {
      int N = 10;
      int a[N] = {42, 37, 31, 16, 53, 19, 47, 58, 33, 25};

      int* heap = new int[MAX_HEAP_SIZE]();
      heap_size = 0;
      heap[0] = NULL;
```

```
	for (int i = 0; i < N; i++) {
		printf("key %d inserted ==> ", a[i]);
		insert_key(heap, a[i]);
		print_heap(heap);
		printf("\n");
	}
	delete[] heap;
}

void test_heap_sort() {
	int N = 10;
	int a[N] = {42, 37, 31, 16, 53, 19, 47, 58, 33, 25};

	int* heap = heap_sort(a, N);
	for (int i = 1; i <= N; i++)
	printf("%d ", heap[i]);
	delete[] heap;
}

int main(int argc, char **argv) {
	int choice;
	cout << "1. Παράδειγμα δημιουργίας σωρού σε γραμμικό χρόνο" << endl;
	cout << "2. Παράδειγμα διαδοχικών διαγραφών μεγαλύτερης τιμής σωρού" << endl;
	cout << "3. Παράδειγμα εισαγωγής νέων τιμών στο σωρό" << endl;
	cout << "4. Ταξινόμηση τιμών με τη Heapsort" << endl;
	cout << "Επιλογή: ";
	cin >> choice;
	if (choice == 1)
	test_bottom_up_heap_construction();
	else if (choice == 2)
	test_maximum_key_deletion();
	else if (choice == 3)
	test_single_key_insertions();
	else if (choice == 4)
	test_heap_sort();
}
```

Κατά την εκτέλεσή του κώδικα εμφανίζεται το ακόλουθο μενού επιλογών:

```
1. Παράδειγμα δημιουργίας σωρού σε γραμμικό χρόνο
2. Παράδειγμα διαδοχικών διαγραφών μεγαλύτερης τιμής σωρού
3. Παράδειγμα εισαγωγής νέων τιμών στο σωρό
4. Ταξινόμηση τιμών με τη Heapsort
Επιλογή:
```

Η επιλογή 1 επιστρέφει:

```
1. Παράδειγμα δημιουργίας σωρού σε γραμμικό χρόνο
2. Παράδειγμα διαδοχικών διαγραφών μεγαλύτερης τιμής σωρού
3. Παράδειγμα εισαγωγής νέων τιμών στο σωρό
4. Ταξινόμηση τιμών με τη Heapsort
Επιλογή: 1
HEAP(10) :[58 53 47 37 42 19 31 16 33 25]
```

Η επιλογή 2 επιστρέφει:

```
1. Παράδειγμα δημιουργίας σωρού σε γραμμικό χρόνο
2. Παράδειγμα διαδοχικών διαγραφών μεγαλύτερης τιμής σωρού
3. Παράδειγμα εισαγωγής νέων τιμών στο σωρό
4. Ταξινόμηση τιμών με τη Heapsort
```

```
Επιλογή: 2
HEAP(10) :[58 53 47 37 42 19 31 16 33 25]
key 58 deleted ==> HEAP(9) :[53 42 47 37 25 19 31 16 33]
key 53 deleted ==> HEAP(8) :[47 42 33 37 25 19 31 16]
key 47 deleted ==> HEAP(7) :[42 37 33 16 25 19 31]
key 42 deleted ==> HEAP(6) :[37 31 33 16 25 19]
key 37 deleted ==> HEAP(5) :[33 31 19 16 25]
key 33 deleted ==> HEAP(4) :[31 25 19 16]
key 31 deleted ==> HEAP(3) :[25 16 19]
key 25 deleted ==> HEAP(2) :[19 16]
key 19 deleted ==> HEAP(1) :[16]
key 16 deleted ==> HEAP(0) :[]
```

Η επιλογή 3 επιστρέφει:

```
1. Παράδειγμα δημιουργίας σωρού σε γραμμικό χρόνο
2. Παράδειγμα διαδοχικών διαγραφών μεγαλύτερης τιμής σωρού
3. Παράδειγμα εισαγωγής νέων τιμών στο σωρό
4. Ταξινόμηση τιμών με τη Heapsort
Επιλογή: 3
key 42 inserted ==> HEAP(1) :[42]
key 37 inserted ==> HEAP(2) :[42 37]
key 31 inserted ==> HEAP(3) :[42 37 31]
key 16 inserted ==> HEAP(4) :[42 37 31 16]
key 53 inserted ==> HEAP(5) :[42 53 31 16 37]
key 19 inserted ==> HEAP(6) :[42 53 31 16 37 19]
key 47 inserted ==> HEAP(7) :[42 53 47 16 37 19 31]
key 58 inserted ==> HEAP(8) :[42 53 47 58 37 19 31 16]
key 33 inserted ==> HEAP(9) :[42 53 47 58 37 19 31 16 33]
key 25 inserted ==> HEAP(10) :[42 53 47 58 37 19 31 16 33 25]
```

Η επιλογή 4 επιστρέφει:

```
1. Παράδειγμα δημιουργίας σωρού σε γραμμικό χρόνο
2. Παράδειγμα διαδοχικών διαγραφών μεγαλύτερης τιμής σωρού
3. Παράδειγμα εισαγωγής νέων τιμών στο σωρό
4. Ταξινόμηση τιμών με τη Heapsort
Επιλογή: 4
16 19 25 31 33 37 42 47 53 58
```

Ερώτημα 1

Ο κώδικας που έχει παρατεθεί υλοποιεί το λεγόμενο MAX-HEAP στο οποίο η τιμή που βρίσκεται στη κορυφή του σωρού είναι η μέγιστη. Να τροποποιηθεί ο κώδικας έτσι ώστε να υλοποιείται σωρός MIN-HEAP και στη κορυφή του σωρού να βρίσκεται η ελάχιστη τιμή (Παρατήρηση: θα χρειαστεί να γίνουν αλλαγές στις συναρτήσεις heapify και insert_key καθώς και μερικές αλλαγές σε ονόματα συναρτήσεων έστι ώστε να ανταποκρίνονται στο νέο σωρό MIN-HEAP).

Ερώτημα 2

Ένας άλλος τροπος για να υλοποιηθει το ερώτημα 1 είναι να διατηρηθεί ο αρχικός κώδικας και να πραγματοποιειται μια προεπεξεργασία στα δεδομένα πολλαπλασιάζοντας κάθε τιμή με -1. Όταν πρόκειται να παρουσιαστούν τα αποτελέσματα θα πρέπει ξανά να γίνεται πολλαπλασιασμός με το -1 έτσι ώστε να εμφανίζονται οι αρχικές τιμες. Υλοποιήστε τη συνάρτηση `test_heap_sort` του αρχικού κώδικα με αυτό το τέχνασμα έτσι ώστε να πραγματοποιείται φθίνουσα ταξινόμηση.