Programmazione [SCQ0093758] - LT Informatica [SC1167] - A.A. 2023-2024

Descrizione | Vista consegna

Secondo Compitino -Es.1 (programmazione)

Disponibile dal: Tuesday, 25 June 2024, 10:38
Data di scadenza: Tuesday, 25 June 2024, 14:13
File richiesti: stesso_ordine.c (Scarica)
Grandezza massima file caricato: 56 KiB
Tipo di lavoro: Lavoro individuale

Punti: 9

Per ciascuna delle seguenti funzioni:
- Scrivere PRE e POST condizioni
- Scrivere la misura di complessita' del problema (il parametro che decresce con le chiamate ricorsive)
- Non e' necessario scrivere ipotesi induttiva etc...ma e' richiesto commentare adeguatamente il codice, spiegando caso base e chiamate ricorsive (in particolare perche' la misura di complessita' decresce in una chiamata ricorsive).

Parte 1

Implementare una funzione RICORSIVA con firma
void swap_eq(int* arr, int dim, int i, int j, int n)
che, dati:
- un array **arr** di interi di dimensione **dim**
- interi non-negativi **i**, **j**, **n** tali che **n>0** e **0 ≤ i+n ≤ j** ed **j+n ≤ dim**

modifica l'array **arr** scambiando la porzione **arr[i] .. arr[i+n-1]** con **arr[j] .. arr[j+n-1].**
Notare che le clausole 0 ≤ i+n ≤ j ed j+n ≤ dim assicurano che le porzioni esistano e siano disgiunte

Parte 2

UTILIZZARE la funzione **swap_eq** per implementare una funzione RICORSIVA con firma
void swap(int* arr, int dim, int m, int n, int p)
che, dati:
- un array **arr** di interi di dimensione **dim**
- interi non-negativi **m**, **n**, **p** tali che **0 ≤ m < n < p ≤ dim**

modifica l'array **arr** scambiando la porzione **arr[m] .. arr[n-1]** con **arr[n] .. arr[p-1].**
Notare che:
1) essendo ***m<n<p<dim***, le porzioni in questione esistono e sono necessariamente disgiunte
2) le porzioni possono avere dimensioni differenti (il cuore dell'esercizio e' capire come decomporre il caso "complesso" di porzioni di dimensioni differente nel caso piu' semplice di porzioni di dimensione uguale (suggerimento: la funzione swap e' una combinazione di condizionali, swap_eq, e chiamate ricorsive soltanto).

N.B. Soluzioni iterative che richiamano la funzione stessa al termine senza calcolare niente ricorsivamente non saranno accettate.
Punti:
- 2 punti per corretta implementazione swap_eq
- 3 punti per corretta implementazione swap
- 2 punti PRE/POST e commenti
- 2 punti misura complessita' swap

File richiesti

stesso_ordine.c

```c
#include <stdio.h>



//
//PRE:
//POST:
void swap_eq(int* a, int dim, int i, int j, int n) {
// codice qui
}


//PRE:
//POST:
void swap(int* a, int dim, int m, int n, int p) {
// codice qui
}



int main(void) {
    int dim;
    int m;
    int n;
    int p;

    scanf("%d", &dim);
    int arr[dim];
    for(int i = 0; i < dim; i++){
        scanf("%d", arr+i);
    }

    void print_array(int *a, int d) {
    for (int i = 0; i < d; i++) {
        printf("%d ", a[i]);
    }
    printf("\n");
}

scanf("%d", &m);
scanf("%d", &n);
scanf("%d", &p);

swap(arr, dim, m, n, p);
print_array(arr, dim);

}

```

VPL

◄ Secondo Compitino - Quiz

Vai a...

Secondo Compitino - Es.2 (programmazione) ►