Laboratorio di Algoritmi e Strutture Dati I

Esercitazione 2

martedì 24 marzo

Istruzioni per l'uso

- Durante il laboratorio (ogni martedì, dalle 11 alle 13:30): topic sul forum studenti, in cui potete fare domande o togliervi eventuali dubbi.

- Fuori dall'orario di laboratorio: sono sempre raggiungibile sul forum o via email (andrea.loddo@unica.it), rispondo appena possibile.
  - NB: **dovete** contattare unicamente me, l'altro tutor non è più disponibile.

- Se dovete condividermi del codice, è preferibile:
  - usare dei tool online (esempio: https://ideone.com/)
  - condividermi un file .zip contenente il vostro progetto.

- La presenza non è più obbligatoria (https://corsi.unica.it/informatica/2020/03/20/sospensione-obbligo-di-frequenza).

C: reminder su potenziali errori

- Dichiarazione di variabile intera:

```
int x = 7;
```

- Passaggio di parametro per valore:

```
function(x);
```

- Passaggio di parametro per indirizzo:

```
function(&x);
```

- Dentro la funzione, x passata per valore:

```
function(int x) {x += ...; }
```

- Dentro la funzione, x passata per indirizzo:

```
function (int *x)
{
        *x = ...; //valore contenuto in x
        x = ...; //indirizzo di x
}
```

- Un array di interi statico:

```
int a[DIM];
```

- Un array di interi dinamico:

```
int *a = (int*)malloc(DIM*sizeof(int));
```

- Utilizzo di un array statico:

```
a[i] = ...
```

- Utilizzo di un array dinamico:

```
a[i] = ...
```

Pseudo codice: tutorial

Alcune dritte necessarie per la "traduzione" dello pseudocodice in codice.

- Firma di un algoritmo/funzione:

```
algoritmo something(type X) → return_type
```

- "to" sta per <=

```
for i ← 0 to n do
```

- Un semplice assegnamento (=)

```
x ← y
```

Lezione 2: dove eravamo rimasti… (1/2)

```
algoritmo selectionSortIter(array A, int n)

for i ← 0 to n-2 do
   min ← i
   for j ← i+1 to n-1 do
        if  (A[j] < A[min]) then  min ← j
   scambia A[min] con A[i]
```

Avete:

- creato una funzione per la creazione dinamica di array di dimensione N, popolati secondo uno schema S = {*ORDINATO, QUASI_ORDINATO, INV_ORDINATO, CASUALE*}.
- implementato il Selection Sort Iterativo che ordina un array di interi.
- implementato il Selection Sort Iterativo che ordina un array di ricette.

Esercizio 2_1: Selection Sort ricorsivo

- Implementare il **Selection Sort** nella sua versione **ricorsiva**.

- Dichiarare un array di N elementi seguendo le seguenti direttive:
  1. Popolare l'array con elementi ordinati (1, 2, 3, 4, 5…);
  2. Con elementi inversamente ordinati (…5, 4, 3, 2, 1);
  3. Con elementi parzialmente ordinati (1, 2, 3, 4, 5, 43, 7, 123, 12, 0, 97…)
  4. Con elementi random (… 79, 43, 99, 1, 67…)

- Testare l'algoritmo con N = {100, 1.000, 10.000, 100.000, 200.000 e 500.000}

- Misurare i tempi di esecuzione per ogni dimensione N e per ogni tipologia di array indicata e stampare il tempo richiesto.

Esercizio 2_1: Selection Sort ricorsivo

- Implementare il **Selection Sort** nella sua versione **ricorsiva**.
- Eseguire i test descritti nella slide precedente.

Pseudocodice:

```
algoritmo selectionSortRec(array A, int dimA, int start)

if (start >= dimA-1) then
        return

minIndex ← findmin(array A, start, start+1, dimA)
scambia A[minIndex] con A[start]
selectionSortRec(A, dimA, start+1)
```

```
algoritmo findmin (array A, int minpos, int start, int dim) → intero

if (start == dim) then
    return minpos
if (A[start] < A[minpos]) then
    minpos ← start
return findmin(array, minpos, start+1, dim)
```

Esercizio 2_2 : Ordinamenti a confronto

- Inserire nel Section Sort Iterativo e nel Selection Sort Ricorsivo il conteggio del numero di confronti e del numero di scambi effettuati per ordinare l'array:
  - Variabili **cont_confr_iter** e **cont_scambi_iter**
  - Variabili **cont_confr_ric** e **cont_scambi_ric**

  NB: contare solo confronti e scambi tra elementi dell'array (non tra indici)

- Osservare i risultati ottenuti, in termini di tempo (o altre anomalie) e riflettere su:
  - Quale delle due versioni del Selection Sort è più efficiente in termini di memoria occupata?
  - Quale delle due versioni del Selection Sort è più veloce?
  - Che differenze riscontrate nell'esecuzione del Selection Sort con i vari schemi di array in input?

- Osservare i risultati ottenuti, in termini di confronti e scambi e riflettere su:
  - Le differenze, in termini di confronti e scambi, tra i diversi schemi in input
  - Differenze tra la versione iterativa e quella ricorsiva

Lezione 2: dove eravamo rimasti… (2/2)

```
algoritmo ricercaBinariaIter(array A, intero numric, intero
     dim) → intero

primo ← 0
ultimo ← dim - 1
while (primo ≤ ultimo) do
     mezzo ← (primo + ultimo)/2
     if (numric < A[mezzo])
             then ultimo ← mezzo-1
     else if (numric == A[mezzo])
             then return mezzo
     else
             primo ← mezzo+1
return -1
```

```
algoritmo ricBinRic(array A, intero numric, intero primo,
     intero ultimo) → intero

if (primo > ultimo)
     then return -1

mezzo ← (primo + ultimo)/2
if (A[mezzo] == numric)
     then return mezzo
else if (A[mezzo] < numric)
     then return ricBinRic(A, numric, mezzo+1, ultimo)
else
     return ricBinRic(A, numric, primo, mezzo-1)
```

Avete:

- Implementato la ricerca binaria iterativa e ricorsiva.
- Contato il numero di elementi dell'array analizzati per trovare la posizione di un numero cercato.
- Misurato il tempo di esecuzione.

Esercizio 2_3 : ancora Ricerca Binaria

Booking.com

- Creare un array ordinato di hotel:

- Gli hotel sono ordinati alfabeticamente per nome
- Accor Hotel viene prima di T-Hotel

```
typedef struct
{
      char nome[DIM_NOME];
      double prezzo_notte;
      int recensione;
} Hotel;
```

- Modificare la ricerca binaria per permettere la ricerca di un hotel in una lista (array statico) ordinata di hotel

- Come si confrontano due stringhe?

- Modificare la versione Iterativa della Ricerca Binaria
- Non sovrascrivere la versione originale (nuovo progetto)

Lezione 2 : Ricapitolando

Implementare:

- E2_1: Selection Sort ricorsivo con interi
- E2_2: Modificare le due versioni del Selection Sort per monitorare:
  - Tempi di esecuzione
  - Numero di confronti e scambi
- E2_3: Ricerca Binaria su hotel
- Ragionare su quanto osservato!

Lezione 2

startForumQuestions();