CoderFarm - Corso base

Lezione 0

Carlo Collodel, Francesco Cerroni

17 ottobre 2022

Presentazione

A chi è rivolto il corso?

Il corso base è per tutti coloro che desiderano iniziare a programmare o vorrebbero prendere parte alla programmazione "competitiva", il nostro obiettivo è in particolare prepararvi per le varie Olimpiadi a carattere Informatico (OIS, OII, ...).
Vi ricordiamo che:

- Per questo corso, non è necessario saper già programmare.
- Le lezioni di questo corso si svolgeranno con cadenza settimanale, il Lunedì alle 16.00. Se ci dovessero essere variazioni vi informeremo per tempo :)

Presentazione

Quali argomenti tratteremo?

Tratteremo:

- Basi di C++.
- Metodi e funzioni più avanzate di C++, la STL.
- Algoritmi noti e tecniche varie.
- Tecniche Greedy.
- Complete search e Backtracking.
- Programmazione dinamica.
- Algoritmi su grafi.
- Strutture dati avanzate.

Presentazione

Chi siamo?

- **Nome:** Carlo Collodel
- **Classe:** 2004
- **Provenienza:** Veneto
- **Risultati alle OII:** Medaglia d'oro alle nazionali 2021
- **Altri risultati olimpionici:** Primo classificato nel 2021 e secondo classificato nel 2022 alle Olimpiadi di Cybersecurity. Medaglia di Bronzo alle Olimpiadi di Matematica 2022. Terzo classificato alle OIS 2022.

Presentazione

Chi siamo?

- **Nome:** Francesco Cerroni
- **Classe:** 2003
- **Provenienza:** Friuli Venezia Giulia
- **Risultati alle OII:** Medaglia d'oro alle nazionali 2021
- **Altri risultati olimpionici:** Vincitore alle Olimpiadi di Astronomia e medaglia di Argento alle Olimpiadi della Fisica

Introduzione

Gare di Informatica?

Come funzionano le gare di Informatica?

- Vengono proposti alcuni problemi e bisogna scrivere dei programmi (quasi sempre in C++) che prendano dei dati in *input*, elaborino questi dati per risolvere uno di questi problemi, e ritornino i risultati in *output*.
- C'è un limite di **tempo** (durata della gara e tempo di esecuzione dei programmi).
- C'è un limite di **memoria**.
- Potrebbero esserci dei *subtask*, ovvero dei "sottoproblemi" o "casi particolari" per ogni problema proposto, questi sono più semplici del problema a cui appartengono e forniscono una quantità di punti parziale quando risolti.

Introduzione

Gare di Informatica!

Che abilità servono per avere un buon risultato in gara?

- Saper riconoscere problemi noti.
- Saper ricavare soluzioni e **algoritmi** efficienti (e funzionanti!) per i problemi proposti.
- Saper *implementare* tali algoritmi in un tempo breve.
- Saper trovare gli errori nelle implementazioni che scriverete :)

Introduzione

Matematica e Informatica

In Matematica, quando si approcciano dei problemi, può servire cercare una formula chiusa, fare osservazioni geometriche oppure prendere delle scelte ottimali "a occhio". In Informatica, invece, molto spesso è necessario fare un grande numero di operazioni per arrivare a un risultato, e queste operazioni devono essere ben definite per poter essere eseguite da un computer.

Esempio:

Data la lista di valori:

$$\mathcal{L} = [6, 10, 3, 2, 42, 69, 7]$$

Ordinare gli elementi in maniera crescente.

Introduzione

Matematica e Informatica

Ordinare una lista è un problema banale per l'uomo.
Ma ora poniamo lo stesso quesito a un computer:
Come potremmo spiegare "come ordinare una lista" a una macchina *stupida* come il computer?

Figura: La catena di montaggio è molto simile a un algoritmo, in quanto si scompone un problema (la produzione di un bene) in passaggi elementari e ben definiti.

Introduzione

Cos'è un algoritmo?

Quando per risolvere un problema ricorriamo ad una sequenza di passaggi diciamo che stiamo utilizzando un **algoritmo**.
Ciò che caratterizza in particolare gli algoritmi è l'essere una sequenza generica di passaggi che non dipendono dai valori sui quali vengono applicati, quindi possono essere utilizzati in una grande varietà di situazioni.

Esempio:

Se trovo un metodo generico per ordinare una lista di valori simile alla lista $\mathcal{L}$ delle slide precedenti, posso applicarlo a qualsiasi lista indipendentemente da quali(!) e quanti(!!) valori contiene.

Introduzione

Cos'è un algoritmo?

Piccolo gioco per voi: provate a suggerire in chat una sequenza di passaggi definiti per ordinare la lista $\mathcal{L}$ precedente:

$$\mathcal{L} = [6, 3, 5, 2, 4, 8, 7]$$

Infine, scrivete se avete un'idea per un algoritmo che funzioni per ogni lista!

Introduzione

Cos'è un algoritmo?

Idea (Bubblesort):

Scorro la lista, se trovo due elementi adiacenti "nell'ordine sbagliato", li scambio.
Osservazione: con questa sequenza di operazioni sono sicuro che l'elemento più grande finirà alla fine.
Ripeto nuovamente questa operazione, non considerando l'elemento della lista appena spostato alla fine.

Quante operazioni eseguo?

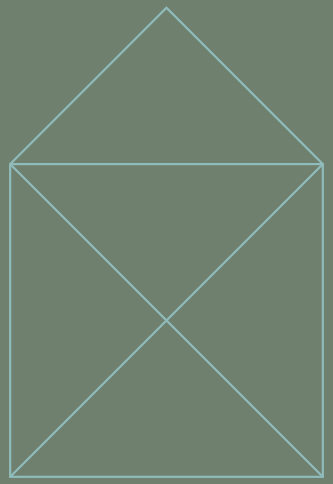
Scorro la lista tante volte quante la sua dimensione, complessità quadratica.
Esistono algoritmi più *efficienti*, parleremo di **Complessità asintotica**.

Problemini

Pensiamo a un algoritmo!

Casetta

Disegnare una casetta senza staccare la matita dal foglio.



Problemini

Pensiamo a un algoritmo!

Casetta

Congratulazioni a tutti coloro che ci sono riusciti! Immagino tutti i vostri algoritmi abbiano due caratteristiche in comune:

1. Il punto scelto di partenza nel disegno è sicuramente uno dei due punti in basso.
2. Il punto di arrivo nel disegno è sicuramente l'altro punto in basso da cui *non* siete partiti.

Per quanto questo problema possa sembrare semplice, le due osservazioni appena fatte sono tutt'altro che banali: esiste infatti un'intera teoria che si occupa di problemi di questo tipo (Teoria dei Grafi, e nel particolare *Grafi Euleriani*) e sarà uno degli argomenti presenti nel corso.

Problemini

Pensiamo a un algoritmo!

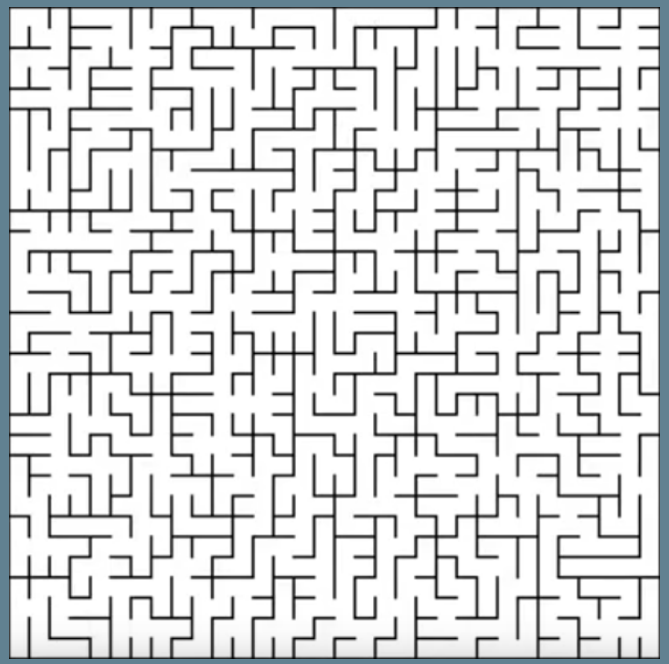
Labirinto

- Ci troviamo all'entrata di un labirinto di cui non conosciamo la topologia.
- Sappiamo che ci sono una sola entrata e una sola uscita.
- Come facciamo a trovare la via d'uscita? (è sufficiente un modo per uscire, non serve trovare il *miglior* modo per uscire)

Problemini

Pensiamo a un algoritmo!

Esempio di labirinto



Problemini

Pensiamo a un algoritmo!

Labirinto

È sufficiente tenere la mano destra (o sinistra) sulla rispettiva parete e continuare a seguirla fino all'uscita!

Parliamo di Numeri!

Interi

I numeri interi vengono salvati nella memoria del computer nella loro rappresentazione *binaria* (base 2).
Ogni cifra di un numero in base 2 può assumere solo il valore 0 o 1. (come per la base 10, nella quale ogni cifra assume un valore tra 0 e 9). Una cifra binaria viene detta **bit**, un gruppo di 8 cifre binarie consecutive è detto **byte**(!).

Parliamo di Numeri!

Interi

Algorithm 1 Come passare dalla base 10 alla base 2?

1: **procedure** BASE10TO2(n)
2: $risultato \leftarrow []$ ▷ Il risultato sarà una lista di *bit*
3: **while** $n \neq 0$ **do** ▷ Continuiamo a dividere finché possiamo
4: $resto \leftarrow n \bmod 2$ ▷ Controlla il bit meno significativo
5: $risultato = [resto] + risultato$
6: $n \leftarrow \frac{n}{2}$

Parliamo di Numeri!

Interi con segno

Per rappresentare gli interi con segno, si usa un'ulteriore rappresentazione detta **complemento a due**.
Il complemento a due permette di rappresentare gli interi negativi dividendo circa a metà il range rappresentabile con un numero binario da n bit(!).
Per passare da un numero negativo a uno positivo in complemento a due (o viceversa) è sufficiente calcolare il **complemento a uno** (cambiare tutti i bit del numero da 0 in 1 e da 1 in 0) del numero interessato e poi aggiungere 1.

Parliamo di Numeri!

Interi con segno

Esempi con 8 bit

$$15_{10} \rightarrow 00001111_2$$
$$-15_{10} \rightarrow -00001111_2 \rightarrow (11110000 + 1)_{C2} \rightarrow 11110001_{C2}$$

$$1_{10} \rightarrow 00000001_2$$
$$-1_{10} \rightarrow -00000001_2 \rightarrow (11111110 + 1)_{C2} \rightarrow 11111111_{C2}$$

$$0_{10} \rightarrow 00000000_2$$
$$-0_{10} \rightarrow -00000000_2 \rightarrow (11111111 + 1)_{C2} \rightarrow 00000000_{C2}$$

I numeri interi senza segno in C++ assumono diverse dimensioni in *bit*

Keyword	Numero di bit	Intervallo
short int	16	$[0, 2^{16} - 1]$
int	32	$[0, 2^{32} - 1]$
long long int	64	$[0, 2^{64} - 1]$
__int128_t	128	$[0, 2^{128} - 1]$

Per riferimento:

- $2^{16} = 65536$
- $2^{32} \approx 4.2 \cdot 10^{9}$
- $2^{63} \approx 1.8 \cdot 10^{19}$
- $2^{127} \approx 3.4 \cdot 10^{38}$

I numeri interi con segno in C++ assumono diverse dimensioni in *bit*

Keyword	Numero di bit	Intervallo
short int	16	$[-2^{15}, 2^{15}-1]$
int	32	$[-2^{31}, 2^{31}-1]$
long long int	64	$[-2^{63}, 2^{63}-1]$
__int128_t	128	$[-2^{127}, 2^{127}-1]$

Per riferimento:

- $2^{15} = 32768$
- $2^{31} \approx 2.1 \cdot 10^{9}$
- $2^{63} \approx 9.2 \cdot 10^{18}$
- $2^{127} \approx 1.7 \cdot 10^{38}$

Parliamo di numeri!

Numeri decimali

Come per gli interi, esistono diversi tipi, per rappresentare i numeri decimali:

Keyword	**Numero di bit**
float	32
double	64
long double	64/80/128

A differenza degli interi, il formato in cui questi numeri vengono salvati in memoria è basato sulla *notazione scientifica* (in base 2 però!). I numeri floating point sono costituiti da tre componenti fondamentali: il bit di segno, i bit di esponente e la mantissa.
Un grande vantaggio è che si possono rappresentare numeri enormi con relativamente "pochi" bit in memoria, in contrasto, però, c'è una grande perdita di precisione.

Hello World!

Il vostro (forse) primo programma in C++

```cpp
#include <iostream>

using namespace std;

int main()
{
  cout << "Hello World!" << endl;
}
```

https://www.onlinegdb.com/online_c++_compiler

Hello World!

Eseguiamo il programma!

```
❯ g++ -o helloworld helloworld.cpp
❯ ls
helloworld  helloworld.cpp
❯ ./helloworld
Hello World!
```

Fine

~~Compiti per casa :)~~

- Seguite la guida che abbiamo scritto su come compilare!
- Provate a compilare ed eseguire l'Hello World se non l'avete già fatto!
- Se volete fare qualche altro esercizio di logica per curiosità: `https://discrete-math-puzzles.github.io/everything.html`

È tutto, grazie per averci seguito :) Se vi serve qualcosa, avete dubbi oppure avete qualche feedback o richiesta particolare, scriveteci!
Ci vediamo alla prossima lezione!

- **E-Mail**: `base@coderfarm.it`
- **Telegram**: T.B.D.