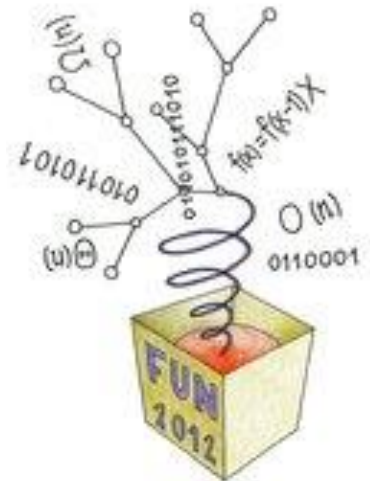
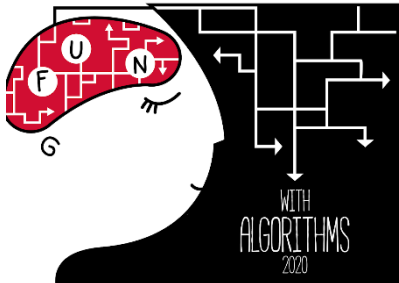


Progettazione di Algoritmi

a.a. 2022/23

Classe 3: matricole congrue 2 modulo 3

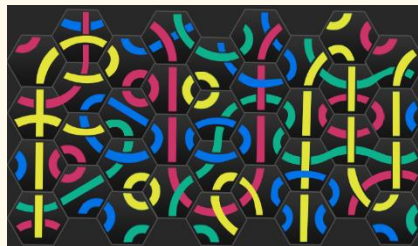
1 marzo 2023



Presentazioni

Marcella Anselmo

- Informazioni: <https://docenti.unisa.it/001367>
- **Orario ricevimento:** giovedì e venerdì 9:30 – 11 nel mio studio o da concordare via email (anche in gruppi)
- Il mio studio è il n° 57 (o meglio 101) al 4° piano della Stecca 7 (fra l'aula F8 e Farmacia)



Informazioni e materiale del corso

- Informazioni pubbliche: <https://docenti.unisa.it/001367>
Didattica: Scheda dell'insegnamento
Materiale e Risorse: Programma, compiti di esame, ...
- Piattaforma <http://elearning.informatica.unisa.it/el-platform/>
compiti da svolgere, slides, avvisi, ...
- Inoltre trovate orario e calendario della vostra classe su
<http://corsi.unisa.it/05121/didattica/orari> (settimana per settimana)
<https://easycourse.unisa.it/AgendaStudenti/>
<https://corsi.unisa.it/informatica/didattica/calendari>

Svolgimento del corso

9 CFU di lezione frontale ed esercitazioni = 72 ore

- Il corso prevede **72 ore** di lezione frontale di carattere teorico o esercitativo, che saranno svolte secondo l'orario previsto, oppure in eventuali ore di recupero, di cui si darà notizia in classe e sulla piattaforma.
- Orario:
 - mercoledì ore 11 – 13 F1
 - giovedì ore 11 – 13 F1
 - venerdì ore 14 – 16, **S6**
- **Vacanze di Pasqua:** 6 - 11 aprile (giovedì – martedì)
- **Ultima lezione** prevista: **giovedì 1 giugno 2023**
- **Altre festività:**

Nessuna! 25 aprile è martedì, 1 maggio è lunedì ☺
15 (me) e 16 (gio) marzo

Prerequisiti

Non vi sono propedeuticità formali, ma dei requisiti **necessari** per affrontare il corso.

Lo studente dovrebbe padroneggiare nozioni di **matematica** acquisite nei precedenti semestri accademici e nei precedenti anni scolastici (**logaritmi, disequazioni, polinomi, sommatorie,...**)

Lo studente dovrebbe avere acquisito la capacità di sviluppare **ragionamenti di tipo logico** (teorema, ipotesi, tesi, dimostrazioni per assurdo, induzione..)

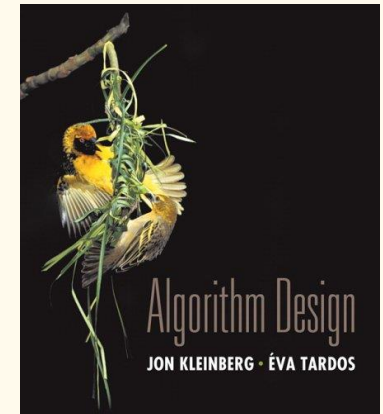
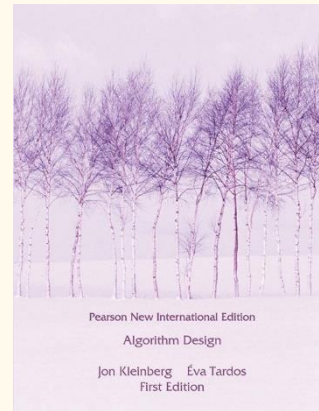
Dovrebbe altresì aver appreso e padroneggiare i concetti di base di **programmazione** (cicli, iterazione, ricorsione,...) e di **strutture dati** elementari (liste, pile, code, alberi,...)

Non **invertite** l'ordine dei corsi: tutto diventa più difficile

Libri di testo

I libri di **testo di riferimento** sono:

[KT] Kleinberg, Tardos.
Algorithm Design.
Pearson Addison Wesley.



[DPV] S. Dasgupta, C. H. Papadimitriou, and U. V. Vazirani.
Algorithms.
McGraw-Hill (cap. 9)

Altri libri di consultazione:

[CLRS] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, *Introduzione agli Algoritmi*, seconda edizione, McGraw Hill.

[DFI] C. Demetrescu, I. Finocchi, G.F. Italiano, *Algoritmi e Strutture Dati*, McGraw Hill, 2004.

Attenti a non perdervi nella ragnatela!

Esami

- L'esame consiste di una **prova scritta** e di una **orale** cui si accede solo dopo il superamento di quella scritta (con circa la metà del punteggio totale). La prova scritta può contenere una parte con domande a risposta multipla. Risolveremo in aula varie prove scritte degli anni scorsi.
- Gli studenti interessati alle prove di esame devono **prenotarsi** su **Esse3** entro il termine utile (5 giorni prima). Ricordo inoltre che è possibile e doveroso **cancellare** la propria prenotazione qualora si decida di non partecipare, per evitare un inutile spreco di risorse.
- DATE ESAMI: su Easycourse e sulla Bacheca di Esse3
- Sono previste **2 prove intercorso**: intorno a **metà aprile** e a **fine corso**. Occorre superare la prima per poter accedere alla seconda. In alcuni casi «non chiari» potrà essere richiesto un orale.

Date esami

Disponibili su Easycourse e sulla Bacheca di Esse3

- 8 giugno ore 15 P3+P4:
 pre-appello e seconda prova intercorso
- 27 giugno ore 15 F8
- 17 luglio ore 15 F8
- 11 settembre ore 15 P4

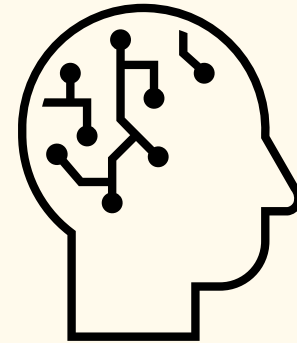
Consigli per lo studio

- **Lezioni** (attive,)
- Slides, appunti, ma... **libri!**
- **Esercizi** (da soli, in gruppi,...)
- **Ricevimento**
- Tutorato
- Organizzare gli esami dei vari corsi
- Non **invertite** l'ordine dei corsi: tutto diventa più difficile
- Non dimenticare obiettivi e **motivazioni**

Un po' di regole

Ad ogni lezione faremo una **pausa** di 5/10 minuti durante la quale potete: **sgranchirvi**, **bere**, mangiare, **telefonare**, **controllare le vostre chat**, **parlare coi vostri colleghi o con me**, etc. etc....

QUINDI



Durante la lezione **NON** farete **niente** di tutto ciò, ma: cercate di concentrarvi per seguire la lezione, capire, porvi e pormi domande, in generale far lavorare il cervello, eventualmente prendere degli appunti e/o seguire sul libro

Contenuto del corso


In breve gli argomenti sono:

- Metodologie di **analisi** di algoritmi
- **Tecniche di progettazione** di algoritmi:
 1. Divide et impera (già in parte a PSD)
 2. Programmazione dinamica
 3. Tecnica *greedy*
 4. Algoritmi esaustivi
- **Strutture dati**: code a priorità e Union-Find
- **Grafi**: proprietà e principali algoritmi su grafi

Contenuto del corso

1. **Analisi asintotica** degli algoritmi (6 ore)
2. Derivazione **ricorrenze** e loro soluzione (4 ore)
3. La tecnica di progetto di algoritmi **divide et impera** e relativi esempi di applicazione: Mergesort, Quicksort, calcolo mediana, moltiplicazione veloce di numeri/ **matrici (Strassen)** , algoritmi ricorsivi su alberi binari (14 ore)
4. La tecnica di progetto di algoritmi **programmazione dinamica** e relativi esempi di applicazione: calcolo di numeri di Fibonacci, combinazioni; problemi di ottimizzazione: Scheduling di risorse, zaino intero, problemi su stringhe, cammini minimi su grafi (14 ore).
5. La tecnica di progetto di algoritmi **greedy** e relativi esempi di applicazione: scheduling di intervalli; scheduling con deadline; compressione dati e codici di Huffman (14 ore).
6. Algoritmi su **grafi**. Connettività e visita di grafi; DAG e ordinamento topologico. Calcolo di cammini minimi (algoritmo di Dijkstra). Calcolo di alberi ricoprenti minimi (algoritmi di Prim e Kruskal) (14 ore).
7. Algoritmi intelligenti di **ricerca esaustiva**: backtracking e branch&bound [DPV, cap. 9] (6 ore)

ALGORITMI IN PRIMA PAGINA

 Corriere Torino

Rasetti: «Un algoritmo ci ruberà il lavoro, ma non la nostra intelligenza»

Mario Rasetti è stato definito un «fisico anomalo», uno scienziato che ha indagato al Politecnico di Torino

1 ora fa

FORMULA 1 FERRARI MERCEDES



F1 2023: l'algoritmo ha deciso che sarà
letto e tra

Comunicato Stampa

Docenti-Riservisti Legge 68/99: la modifica dell'algoritmo lede il loro diritto al lavoro

GPS e algoritmo, ancora caos con il rischio di discriminazione

26 Febbraio 2023



In principio c'erano gli algoritmi....

“Before there were computers, there were algorithms. But now that there are computers, there are even more algorithms, and algorithms lie at the heart of computing.”

T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, Introduction to Algorithms.

- Algoritmi di tipo numerico furono studiati da matematici babilonesi ed indiani più di 3000 anni fa.

Algoritmi in uso fino a tempi recenti furono studiati dai matematici greci 500 anni a.C.

- Esempio: Algoritmo di Euclide per il Massimo Comun Divisore
- Esempio: Algoritmi geometrici (calcolo di tangenti, sezioni di angoli, ...)

Cos'è un algoritmo?

- Un algoritmo è un procedimento per risolvere un problema mediante una sequenza finita di passi elementari
- Il procedimento deve essere descritto in modo preciso allo scopo di poterne automatizzare l'esecuzione
- Il termine deriva dal nome del matematico persiano **Abu Ja'far Muhammad ibn Musa Khwarizmi**
 - Autore di un primo fondamentale trattato di algebra
 - Un cratere lunare porta il suo nome



... lo capiremo pian piano...

Caratteristiche degli algoritmi

- La sequenza di istruzioni deve essere finita (*finitezza*);
- La procedura deve portare ad un risultato (*effettività*);
- Le istruzioni devono essere eseguibili materialmente (*realizzabilità*);
- Le istruzioni devono essere espresse in modo non ambiguo (*non ambiguità*).

Algoritmo vs Programma

- Un **algoritmo** descrive (ad alto livello) una procedura di calcolo che, se seguita, consente di ottenere un certo risultato
- Un **programma** è l'implementazione di un algoritmo mediante un opportuno linguaggio di programmazione.
 - Un programma può essere direttamente eseguito da un calcolatore (processo in esecuzione); un algoritmo solitamente no.

Descriveremo gli algoritmi in **pseudo-codice**

Obiettivi

Abbiamo bisogno di algoritmi ogni qual volta vogliamo scrivere un programma.

Obiettivo finale: programmare in maniera **consapevole** e **creativa**

Dal problema reale al programma che lo risolve:

1. **Formalizzazione** del problema
2. Scelta della **tecnica** per progettare algoritmo
3. Scelta della **struttura dati** per organizzare i dati
4. Dimostrazione della **correttezza**
5. **Analisi** risorse usate / efficienza

Capacità acquisite

Sono tante...

1. Diventare un miglior **programmatore**
2. Affinare le **capacità analitiche**
3. **Pensare** algoritmicamente
4. Conoscere i maggiori **successi** in Informatica
5. Affrontare **colloqui di lavoro**

Siamo sicuri vi servirà?



- **Colloqui di lavoro**

Es. Google: una fra le 6 diverse tipologie di quesiti nella selezione del personale è quella delle **domande algoritmiche** che presentano un compito da svolgere (come fareste a ... ?) ponendo un vincolo nello svolgimento e richiedendo di economizzare qualcosa (tempo, fatica o denaro). Il corso di PA vi «allena» a fare questo.

- **Lavoro in azienda**

Es.: **modificare** programma creato da uno sviluppatore vari anni prima. Occorre capire cosa fa e dove intervenire. Se segue una certa tecnica di progettazione (che voi conoscete), sarà più semplice metterci mano.

Es.: avendo sviluppato un nuovo programma, avrete più «armi» a disposizione per convincere (voi stessi e) il capo della **bontà** della vostra soluzione

- **Futuro**: i linguaggi di programmazione, le macchine, l'organizzazione dei dati, ... cambiano nel tempo, ma una mente aperta, abituata a pensare «in generale», «in linea teorica», affronterà meglio i **cambiamenti**
- **Passato**: non siete i primi informatici; perché non accettare o almeno conoscere le **tecniche affinate** nel tempo da chi è venuto prima di voi?

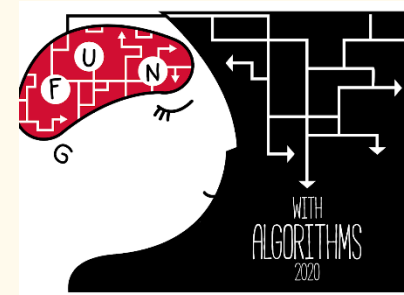
Perché studiare algoritmi?

- Importanti per tutte le altre branche dell'**Informatica**
- Giocano un ruolo chiave nell'innovazione **tecnologica**

– “Everyone knows Moore’s Law – a prediction made in 1965 by Intel co-founder Gordon Moore that the density of transistors in integrated circuits would continue to double every 1 to 2 years....in many areas, performance gains due to improvements in algorithms have vastly exceeded even the dramatic performance gains due to increased processor speed.”

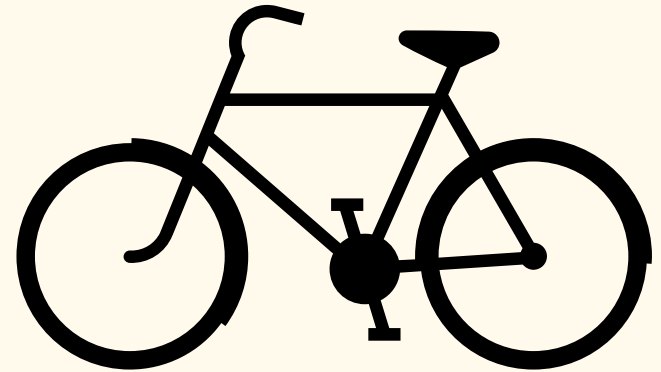
- Excerpt from *Report to the President and Congress: Designing a Digital Future*, December 2010 (page 71).

- Forniscono un **impulso** alla tecnologia
- **Sfida intellettuale**
- Per **divertimento**! «FUN with Algorithms» è una conferenza



Bici o moto?

«Sì, ma io voglio studiare
Intelligenza Artificiale, Machine
Learning, Quantum Computing, ...»



Certo, tutti (o quasi) sono più attratti dal guidare la moto che la bicicletta, ma È possibile guidare la moto se prima non si impara a guidare la bicicletta? [cit.]

Cominciamo a pedalare.....

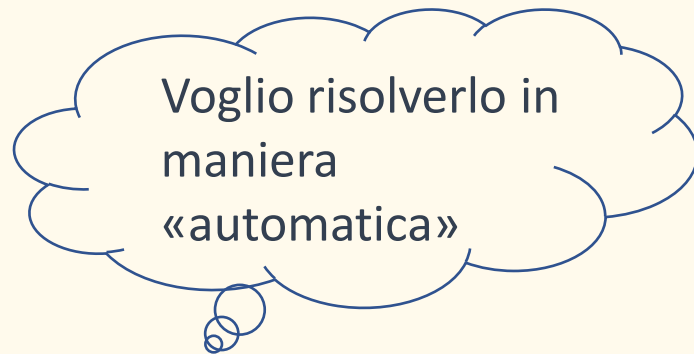
Introduzione alla progettazione e analisi algoritmi

- Sono le **BASI** necessarie conoscere per affrontare ambiti più avanzati
- **Ordinamento**: problema obsoleto? Basi per motori di ricerca
- Divide et Impera più che antico? FFT....
- **Programmazione dinamica**?: Bioinformatica usa massivamente
- **Greedy**? Lo usate sempre senza saperlo: è importante usarlo bene
con consapevolezza

Ho un problema....

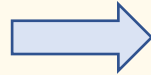
Un problema reale....

1. gestire richieste di affitto campo di calcio
2. calcolare/prevedere numeri di amici di Facebook che avrò fra tot anni
3. Scegliere un regalo per San Valentino...



Approccio 1: lo «smanettone»

Ho un computer + conosco un linguaggio di programmazione



Scrivo del codice, compilo, correggo,

FATTO!



Funziona: l'ho
provato su vari
dati e ci ha
messo pochi
secondi.

Sei sicuro che
funzioni? O finisce
come con i Docenti-
Riservisti?

Ci mette
molto
tempo?

Si poteva
fare
meglio?

Approccio 2: l'informatico laureato!

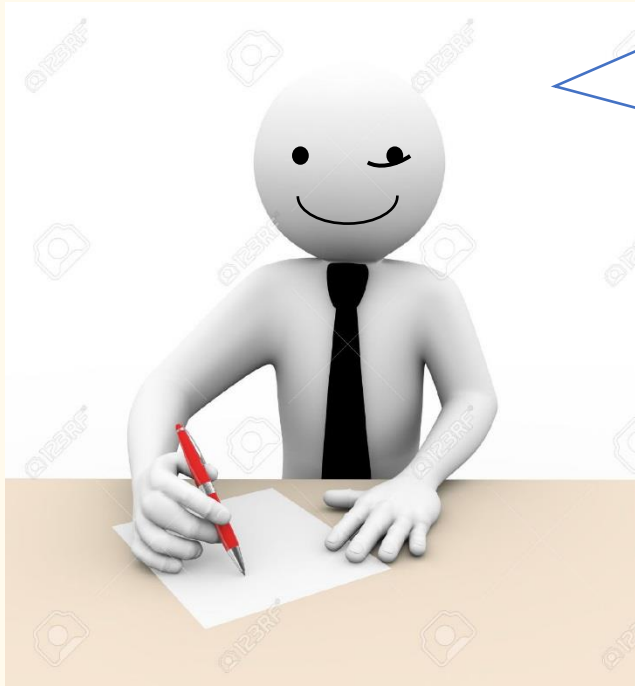
Ho un computer + conosco un linguaggio di programmazione, ma



Prendo carta e penna:

- **Formalizzo** il problema: quali i dati che ho? quali voglio?
- Cerco tutte le **soluzioni** che riesco a trovare applicando le **tecniche** studiate, di cui dimostro il corretto funzionamento su ogni input
- Quale scelgo? Valuto la **complessità** di tempo/spazio di ognuna relativamente ad una organizzazione dei dati. **Scelgo!**
- Vado al computer ad **implementarla**

Alla fine....



Ti posso
dimostrare che
funziona su ogni
input e che fra
quelle
considerate è la
più efficiente.

Sei sicuro
che
funzioni?

Ci mette
molto
tempo?

Si poteva
fare
meglio?

Problemi...

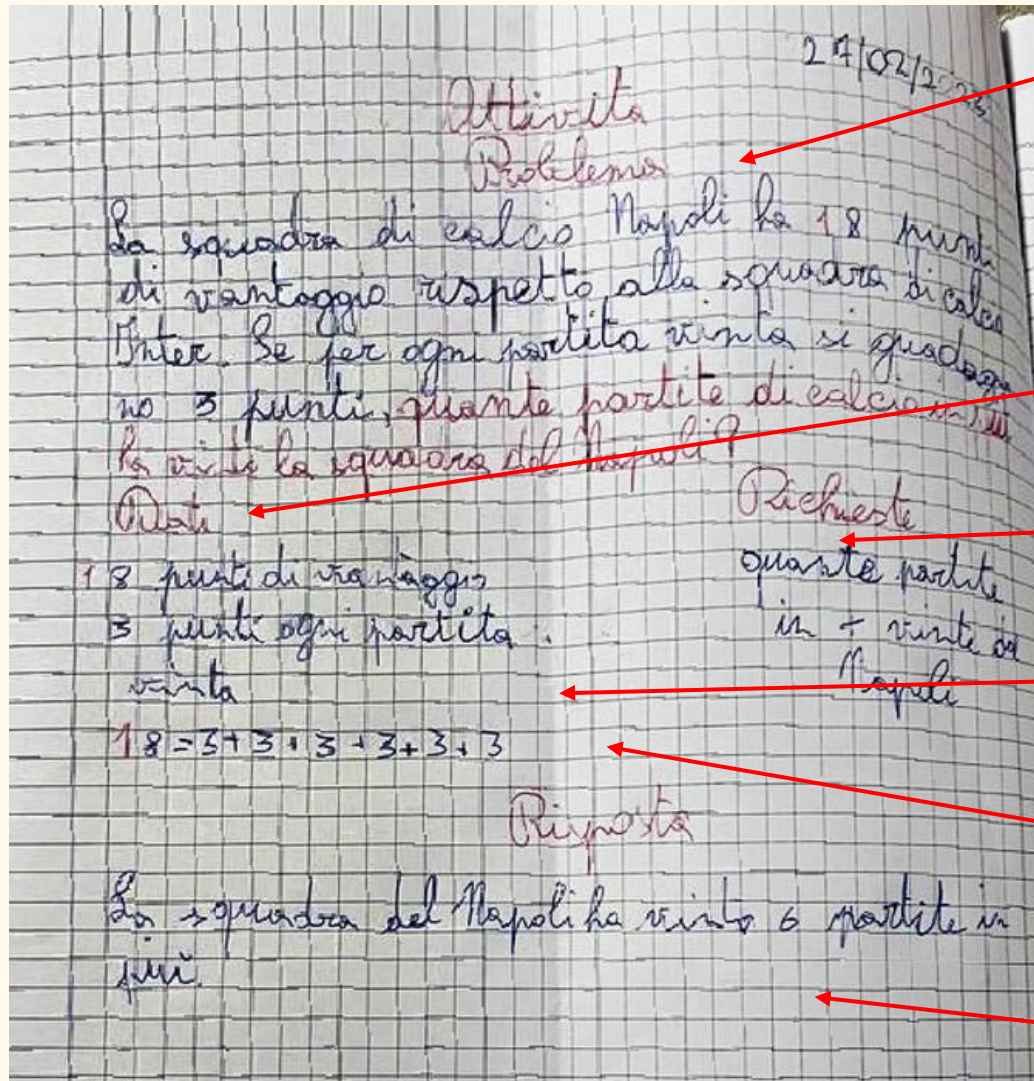
Un algoritmo risolve un problema, ma...

... cos'è un problema?

Corrispondenza fra spazio delle **istanze** (dati in ingresso) e delle **soluzioni**



Torniamo a scuola



Problema
«reale»

Dati in ingresso
(input)

Dati in uscita
(output)

Esecuzione
algoritmo

algoritmo

Soluzione

Un esempio

Problema reale / concreto:

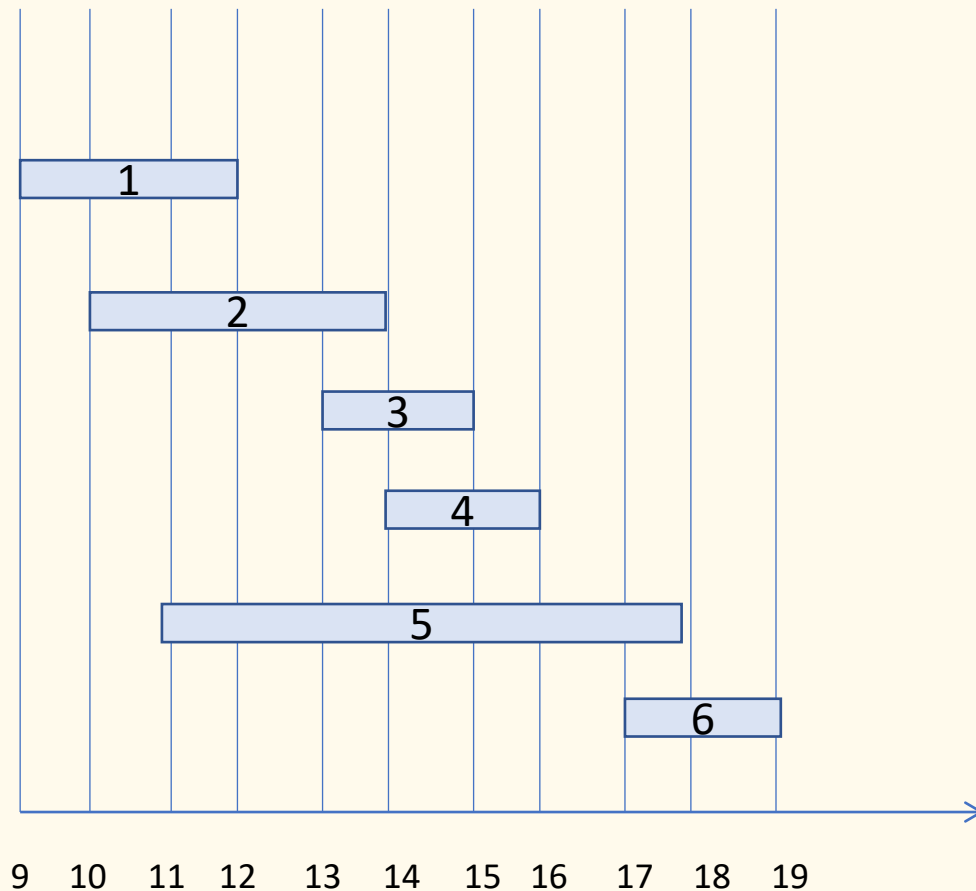
Scheduling di attività:

In una scuola c'è **1** sala computer e più classi che vogliono accedervi ognuna in certi orari.

Come accontentare il maggior numero di classi?

Oggi, come lo risolvereste?

Esempio



$$S = \{1, 2, 3, 4, 5, 6\}$$

$$s_1 = 9, f_1 = 12$$

$$s_2 = 10, f_2 = 14$$

$$s_3 = 13, f_3 = 15$$

$$s_4 = 14, f_4 = 16$$

$$s_5 = 11, f_5 = 18$$

$$s_6 = 17, f_6 = 19$$

Soluzioni **ammissibili** :

$$S_1 = \{1, 3, 6\}$$

$$S_2 = \{1, 6\}$$

$$S_3 = \{2, 4, 6\}$$

$$S_4 = \{5\}$$

Soluzioni ottimali:

$$S_1 = \{1, 3, 6\}$$

$$S_3 = \{2, 4, 6\}$$

Valore ottimo = 3

Formalizzazione del problema

Problema computazionale: definito da input e output

Esempio (continua)

- **Input / Dati:**

$S = \{ 1, 2, \dots, n \}$ insieme delle classi

Per ogni classe i :

- s_i tempo di inizio
- f_i tempo di fine

- **Output / Soluzione:**

S' sottoinsieme di S di **attività compatibili** (= orari non si accavallano) tale che **Card(S') massima**

Nota: S' = soluzione ottimale, Card(S') = valore ottimo

Varianti

- Nell'**input** : Scheduling di attività pesato
Ad ogni classe è associato un valore
Cerco **S'** il cui valore totale sia massimo (non necessariamente la cardinalità)
- Nell'**output**
 - Cerco soltanto **valore ottimo**
 - Cerco **tutti S'**

Scelta della tecnica per Scheduling di attività

1) Ricerca esaustiva/ Forza bruta / naif

- Considero **tutti** i sottoinsiemi di S
- Per ognuno **verifico** compatibilità
- Fra i sottoinsiemi di attività compatibili restituisco quello di cardinalità **massima**

Q: Quanti sono tutti i sottoinsiemi di $\{1, 2, \dots, n\}$?

Sono 2^n

Analisi: il tempo di esecuzione di tale algoritmo è **esponenziale!**

Esponenziale = non accettabile

Altra soluzione?

Tecniche di progettazione

- 2) Divide et impera lo risolve in tempo **esponenziale**
- 3) **Programmazione dinamica** lo risolve in $O(n \log n)$
- 4) Tecnica **greedy** lo risolve in $O(n \log n)$

Tempo $O(n \log n)$ è accettabile, esponenziale no

C'è davvero differenza?

$n \log n$ vs 2^n

Table 2.1 The running times (rounded up) of different algorithms on inputs of increasing size, for a processor performing a million high-level instructions per second. In cases where the running time exceeds 10^{25} years, we simply record the algorithm as taking a very long time.

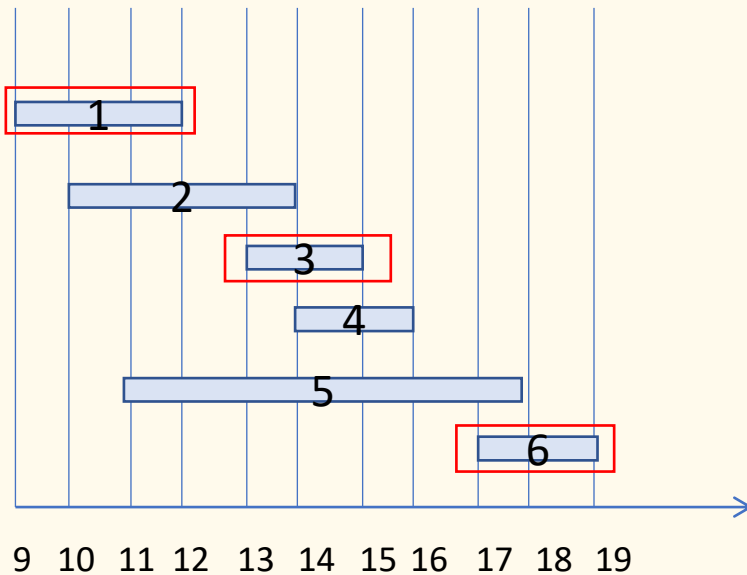
	n	$n \log_2 n$	n^2	n^3	1.5^n	2^n	$n!$
$n = 10$	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	4 sec
$n = 30$	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	18 min	10^{25} years
$n = 50$	< 1 sec	< 1 sec	< 1 sec	< 1 sec	11 min	36 years	very long
$n = 100$	< 1 sec	< 1 sec	< 1 sec	1 sec	12,892 years	10^{17} years	very long
$n = 1,000$	< 1 sec	< 1 sec	1 sec	18 min	very long	very long	very long
$n = 10,000$	< 1 sec	< 1 sec	2 min	12 days	very long	very long	very long
$n = 100,000$	< 1 sec	2 sec	3 hours	32 years	very long	very long	very long
$n = 1,000,000$	1 sec	20 sec	12 days	31,710 years	very long	very long	very long

accettabile

non accettabile

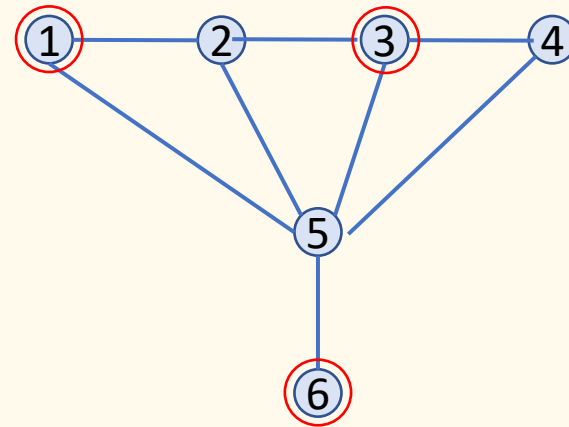
Altra soluzione

Grafo della compatibilità: ogni nodo è un intervallo; due nodi collegati se si accavallano



Soluzione $S_1 = \{1, 3, 6\}$

\Rightarrow



Insieme di **nodì indipendenti** =
tale che ogni coppia di nodi NON è
collegata

Problema più generale

Problema dell'insieme indipendente

Input: grafo

Output: insieme indipendente di cardinalità **massima**

Una soluzione del problema dell'**insieme indipendente** fornisce una soluzione del problema dello **scheduling di attività**.

Purtroppo però il problema dell'**insieme indipendente** è un problema «difficile»

Riassumendo

- Dal problema reale al **problema computazionale**
- Bisogna conoscere più **tecniche di progettazione** di algoritmi
- Necessario saper **analizzare** l'efficienza (tempo/spazio)
- Dimostrare **correttezza**
- Molti problemi si possono esprimere con i **grafi**

Difficoltà

- Bisogna cambiare **mentalità**
- Prendere tempo a **progettare** e **valutare**; riparare in corso d'opera è più costoso
- c'è bisogno di (un po' di) attenzione e **rigore** metodologico
- .. di (un po' di) **matematica**
- ... di **creatività**...
- Accettare di seguire una **tecnica**, ben fondata e sperimentata

Obiettivi formativi...

- Estrapolare il problema computazionale
- Riconoscere (analogie e variazioni con) problemi noti
- Scegliere tecnica/he più appropriata/e
- Applicarla/e correttamente
- Valutare l'efficienza

ESERCIZI

Nei compiti che seguono, potete provare a formalizzare il problema computazionale alla base del problema «reale» descritto.

Appello 25 gennaio 2021

Quesito 3 (20 punti)

I signori Braccino hanno comprato una **bicicletta elettrica** per i loro 6 figli. Siccome è il loro **unico** mezzo di locomozione, ogni mattina i ragazzi si ritrovano, ognuno esprime la propria richiesta per la bicicletta e cercano la soluzione che accontenti il maggior numero di loro. Oggi Ada ne avrebbe bisogno dalle 10 alle 14, Bea dalle 11 alle 12, Camillo dalle 15 alle 18, Dante dalle 12 alle 16, Elena dalle 13 alle 15 e Fabio dalle 15 alle 17.

Quanti ragazzi al massimo potranno usufruire della bicicletta? **Giustificate** la risposta, indicando quale algoritmo, fra quelli studiati, avete utilizzato.

Appello 14 febbraio 2017

Quesito 2 (23 punti) (*San Valentino*)

Oggi è San Valentino e volete comprare un regalo per il vostro partner. Andate nel suo negozio preferito e selezionate un insieme S di n oggetti che sicuramente sarebbero graditi. Ma, guardando nel vostro portafoglio, vi accorgete che con la somma che avete, di certo non potete comprarli tutti! Decidete allora di assegnare ad ogni oggetto il presunto valore di gradimento e di selezionare quindi un insieme di oggetti di S che abbiano un valore di gradimento massimo, ma che non superi la somma che avete a disposizione. Il problema non è però di facile soluzione, anche perché non avete molto tempo a disposizione, in quanto il negozio sta per chiudere.

Formalizzate il problema reale in un problema computazionale e risolverlo nel modo più efficiente possibile con la tecnica che ritenete più opportuna. E' necessario descrivere l'algoritmo soluzione e valutarne l'efficienza.

Appello 29 gennaio 2015

Quesito 2 (24 punti)

Dopo la Laurea in Informatica avete aperto un campo di calcetto che ha tantissime richieste e siete diventati ricchissimi. Ciò nonostante volete guadagnare sempre di più, per cui avete organizzato una sorta di asta: chiunque volesse affittare il vostro campo (purtroppo è uno solo), oltre ad indicare da che ora a che ora lo vorrebbe utilizzare, deve dire anche quanto sia disposto a pagare. Il vostro problema è quindi scegliere le richieste compatibili per orario, che vi diano il guadagno totale maggiore. Avete trovato una soluzione, ma è molto lenta. Vi ricordate allora che al corso di Algoritmi vi avevano tempestato con le varie tecniche di progettazione di algoritmi: potranno esservi utili (almeno una volta nella vita)?

Formalizzate il problema reale in un problema computazionale e risolverlo in maniera che sia il più efficiente possibile. Giustificate le risposte.

DEFINIRE PROBLEMA COMPUTAZIONALE

Quesito (22 punti) (*Campi di calcetto*)

Dopo il successo del vostro primo campo di calcetto, avete aperto molti altri campi di calcetto, all'interno di un unico complesso. Ogni giorno raccogliete le richieste per utilizzare i vostri campi, ognuna specificata da un orario di inizio e un orario di fine. Oramai avete un numero di campi sufficiente ad accontentare sempre tutte le richieste. Volete però organizzare le partite nei campi in modo da accontentare tutti, senza che vi siano sovrapposizioni di orari, ma con il minimo numero possibile di campi (la manutenzione costa!).

Formalizzate il problema reale in un problema computazionale.