Introduzione agli Algoritmi

Prof. Emanuela Fachini

- Contenuto del corso
- Motivazioni
- Qualche definizione iniziale

Pagina del corso: http://twiki.di.uniroma1.it/twiki/view/Intro_algo/AD/WebHome

Il libro

Strutture dati e algoritmi in Java di Goodrich Michael T. - Tamassia Roberto Zanichelli, 2007.

pagina del libro (in inglese):

http://ww0.java4.datastructures.net/

I lucidi utilizzati a lezione saranno messi in rete sulla pagina del corso, non sono utilizzabili per lo studio in alternativa al libro. Servono come appunti per rivedere i contenuti di una lezione seguita e come individuazione dell'argomento sviluppato durante una lezione persa.

Gli esami

5 appelli: giugno, luglio, settembre, gennaio e febbraio. Sempre nella prima e quarta settimana dopo la fine delle lezioni

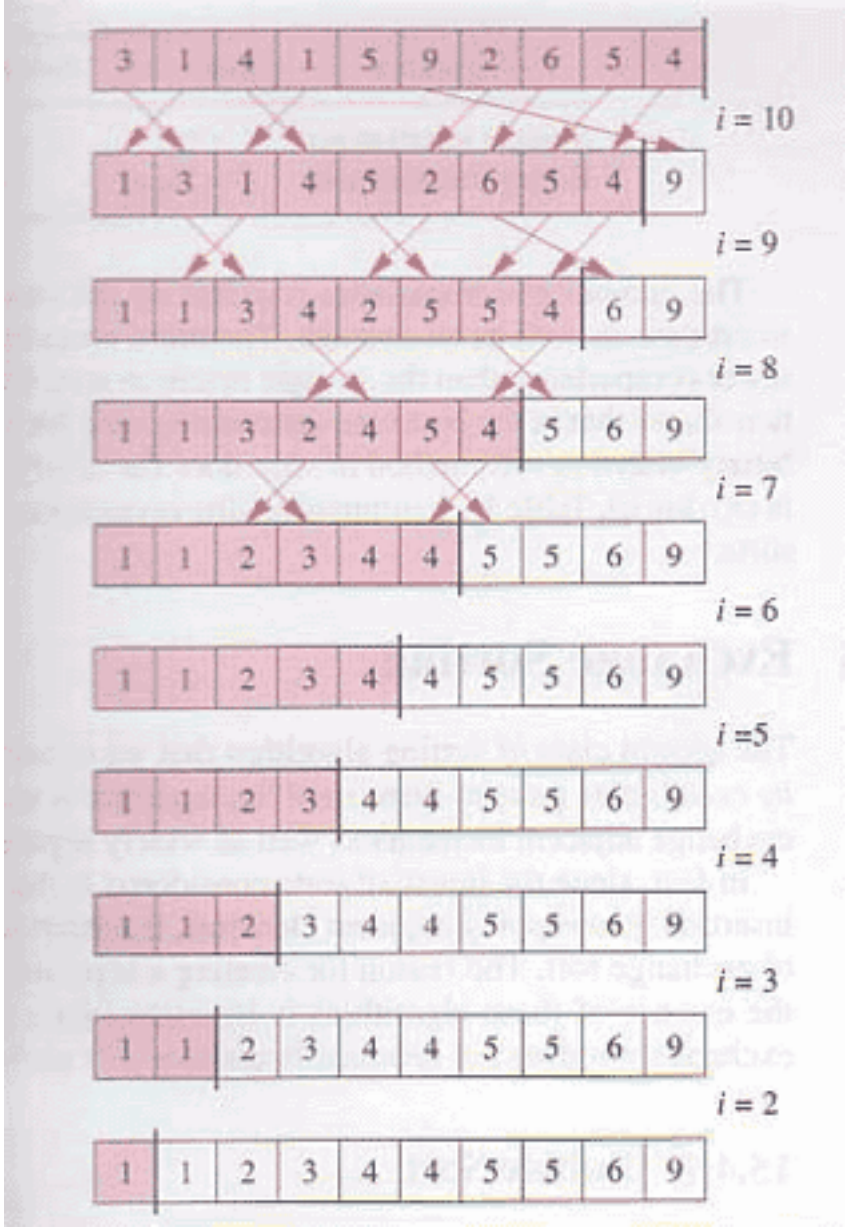
Modalità: l'esame consiste in una prova scritta ed un'eventuale prova orale.

Prova intermedia: è prevista a metà semestre e riguarderà gli argomenti sviluppati fino a quel momento. Chi supera la prova intermedia all'esame farà solo la seconda parte della prova scritta, nei primi 2 appelli

Bonus: mezzo punto in più a chi conclude l'esame nella sessione estiva e un altro mezzo punto a coloro che hanno anche superato la prova intermedia.

Cosa si studierà?

Algoritmi per problemi di base come l'ordinamento di elementi memorizzati in una lista omogenea (array)



Cosa si studierà?

1. Algoritmi per problemi di base come l'ordinamento,

2. la ricerca di un elemento in un insieme o in una collezione (insieme con ripetizioni),
+ altre operazioni come inserimento e cancellazione di un elemento da un insieme o collezione.

Cosa si studierà?

1. **Algoritmi per problemi di base come l'ordinamento,**

2. **la ricerca di un elemento in un insieme o in una collezione (insieme con ripetizioni),**

3. **la gestione di code di vario tipo**

In conclusione:

Studieremo algoritmi per problemi di base come

- **l'ordinamento,**
- **la ricerca di un elemento in un insieme o in una collezione (insieme con ripetizioni),**
- **la gestione di code**

Per descrivere gli algoritmi dovremo rivedere/studiare alcune

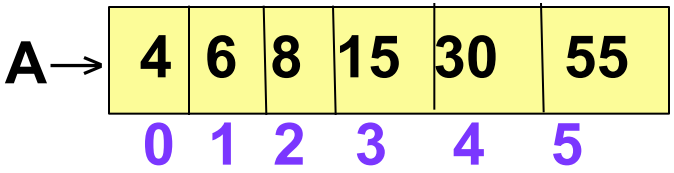
strutture dati elementari

Cos'è una struttura dati?

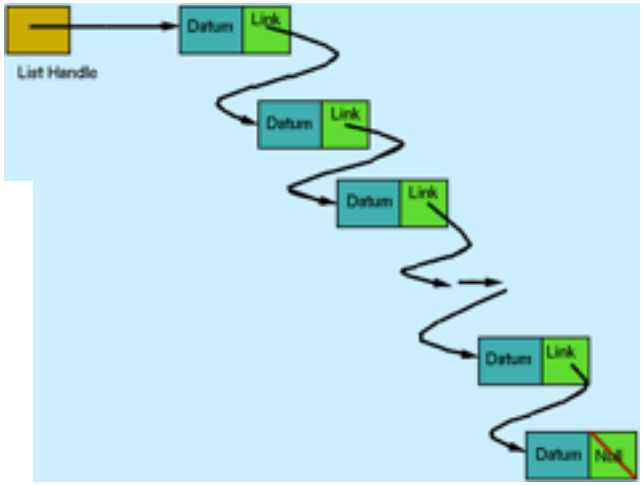
Definizione: una struttura dati è un modo di organizzare i dati in memoria

Esempi:

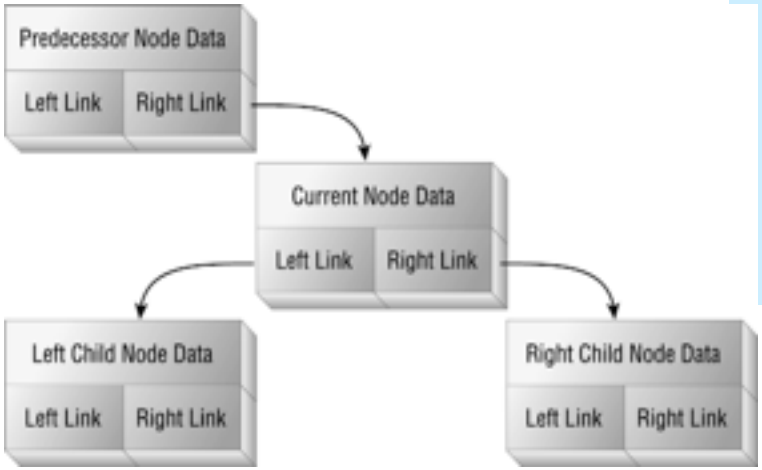
arrays (liste di interi)



liste concatenate (liste linkate)



alberi



...

Perchè ci interessano?

Gli algoritmi e le strutture dati in generale ci interessano perchè non c'è applicazione informatica che non si basi su un algoritmo e non c'e' algoritmo che non si basi su una struttura dati!

Esempi:

- **trovare informazioni in rete**
- **proteggere le informazioni che sono trasmesse (crittografia)**
- **calcolare il percorso meno costoso per collegare tra loro dei luoghi**
- **assegnare posti ai passeggeri in un aereo, treno, autobus....**
- **...**

Perchè ci interessano?

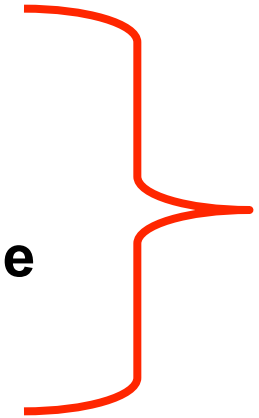
Gli algoritmi e delle strutture dati che studieremo in questo corso sono ben noti e implementati.

Ma è opportuno conoscerli bene perchè

- **possono essere utilizzati come moduli in algoritmi più complessi**
- **sono abbastanza semplici da essere utilizzati per introdurre importanti metodi e tecniche di**
  - ✓ **progettazione e analisi degli algoritmi e**
  - ✓ **dimostrazioni di correttezza**
- **Il loro studio è un allenamento mentale al problem solving: l'arte di costruire nuove soluzioni algoritmiche per i problemi**

Problem solving

1. Definizione del problema
2. Progetto dell'algoritmo che lo risolve
3. Analisi dell'algoritmo (correttezza, complessità di tempo e di spazio)
4. Implementazione in un linguaggio di programmazione
5. Testing
6. [manutenzione]



Corsi di Programmazione

Definizione di problema e istanze

Un problema è ben definito quando è chiaro che cosa ci aspettiamo in output in funzione dell' input.

Un'istanza di un problema è un esempio di input.

Esempio di problema ben definito:

Problema 1: Si vuole calcolare il massimo comun divisore tra due numeri interi

Input: due numeri interi

Istanza: 27,18

Output: il MCD dei due numeri in input

Definizione di problema - 2

Esempio: definizione del problema dell'ordinamento: data una lista di n elementi A= <a1,a2,...,an> presi da un insieme totalmente ordinato, secondo un particolare ordinamento, si tratta di produrre una lista ordinata degli n elementi, in uno dei possibili versi (crescente o descrescente) dell'ordinamento.

Istanza del problema: una sequenza di numeri A = <5,8,7,6,10,43,67>, con l'ordinamento naturale
Output: A=<5,6,7,8,10,43,67> crescente
o A= <67,43,10,8,7,6,5> decrescente

Istanza del problema: una sequenza di parole
B=<pesche, mele, pere, ciliege,albicocche>, con l'ordinamento crescente lessicografico (alfabetico)
Output, relativo all'istanza:
B = <albicocche,ciliegie,mele,pere,pesche, uva>

Istanza del problema: una sequenza di parole
B=<pesche, mele, pere, ciliege,albicocche>, con l'ordinamento crescente quasi lessicografico (per lunghezza e a parità di lunghezza in alfabetico)
Output, relativo all'istanza:
B = < uva,mele,pere,pesche,ciliegie, albicocche>

Progetto dell'algoritmo

...And that, in simple terms, is how you increase your ranking on search engines."

Cos'è un algoritmo?

Definizione informale:
un algoritmo è una sequenza finita di istruzioni da eseguire in sequenza, ciascuna eseguibile in tempo finito, e tale da produrre un determinato risultato in un tempo finito

Esempio:

algoritmo di Euclide (~300 a.c.) per il calcolo del MCD
INPUT: due numeri interi *a* e *b*
precondizione: *a* e *b* ≠ 0
Finché *a* ≠ *b*
se *a* > *b* allora ad *a* assegna *a* - *b*
se *a* < *b* allora a *b* assegna *b* - *a*
OUTPUT *b (qui a=b)*

Descrizione di un algoritmo

1. **Può essere descritto in linguaggio naturale (p.e. italiano), utilizzando uno pseudocodice o dei diagrammi,…**
2. **L'input può mancare ma non l' output!**
3. **Ogni istruzione deve essere precisa e chiara**
4. **L'algoritmo deve terminare dopo un numero finito di passi**
5. **Ogni istruzione che lo compone deve essere abbastanza elementare e eseguibile in tempo finito**

Esempio di non algoritmo:

1. **Impara il cinese**
2. **Traduci in cinese**

Da dove viene la parola?

La traduzione latina, risalente al XII secolo, di un libro di Muhammad ibn Mūsa 'l-Khwārizmī era intitolata "l-Khwārizmī sui numeri indiani", che fu interpretato come "algoritmi sui numeri indiani".

Muhammad ibn Mūsa 'l-Khwārizmī è un Matematico persiano, nato nel 780 circa e morto nel 850 , famoso per il libro "Kit āb al-djabr wa 'l-muq ābala " il primo libro che tratta in modo sistematico le equazioni lineari e di secondo grado e dal cui titolo è nato anche il termine algebra (al-djabr)

Analisi di un algoritmo: correttezza

Correttezza:

un algoritmo è (totalmente) corretto quando termina su tutti gli input previsti ed è corretto, cioè produce l'output atteso.

tecniche di prova: induzione e invariati di ciclo

Analisi di un algoritmo: complessità

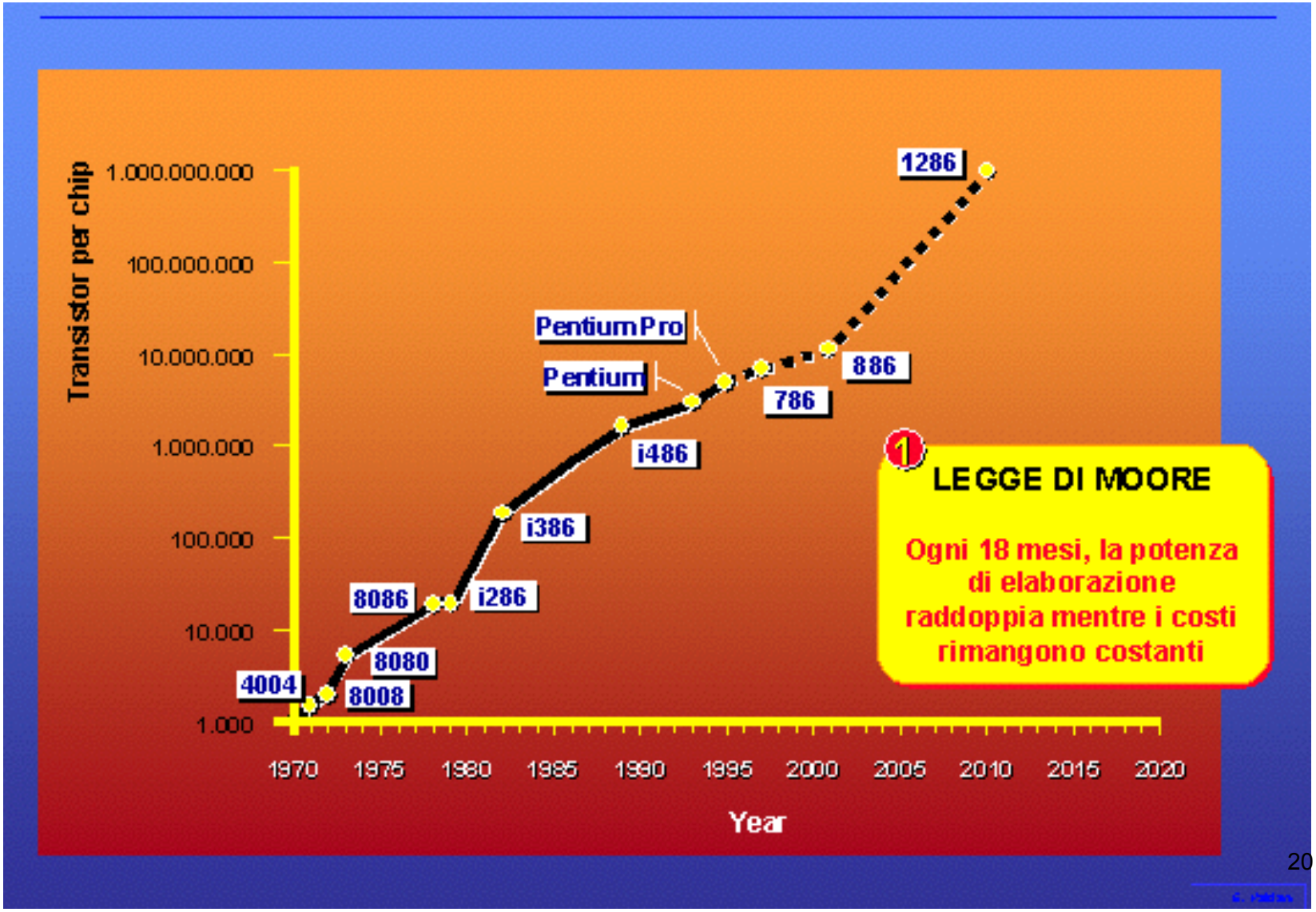
Tra due algoritmi che risolvono correttamente lo stesso problema, con quale criterio decido quale implementare?

La prima cosa da esaminare sono le risorse di calcolo necessarie per ottenere la risposta: si sceglierà quello più conveniente nel contesto.

Ma come valutare questo aspetto?

Risorse di quale macchina?

Negli ultimi anni la potenza di calcolo è aumentata di circa un miliardo di volte!



algoritmi/programmi

programmi	algoritmi
scritti in un linguaggio di programmazione	scritti in pseudocodice o anche linguaggio naturale strutturato
eseguiti su un computer	eseguiti su un modello di calcolo

Modello di calcolo

Essenzialmente in un modello di calcolo dobbiamo poter dire quali operazioni hanno costo costante.

Per stabilire questo dobbiamo avere un

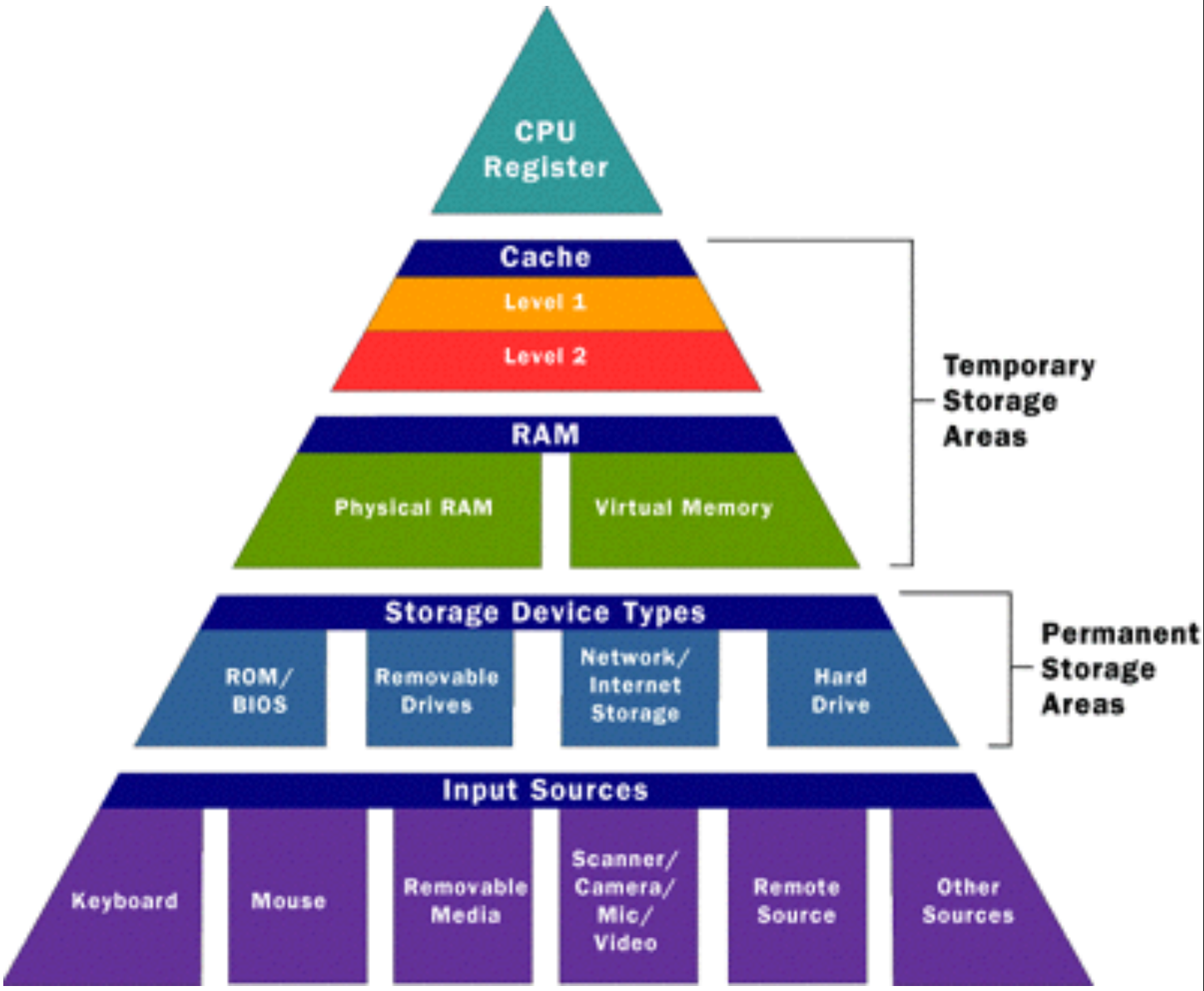
modello della memoria

Memorie

Ci sono molti tipi di memoria in un computer reale.

La Random Access Memory è presente su ogni computer, in alcune varianti.

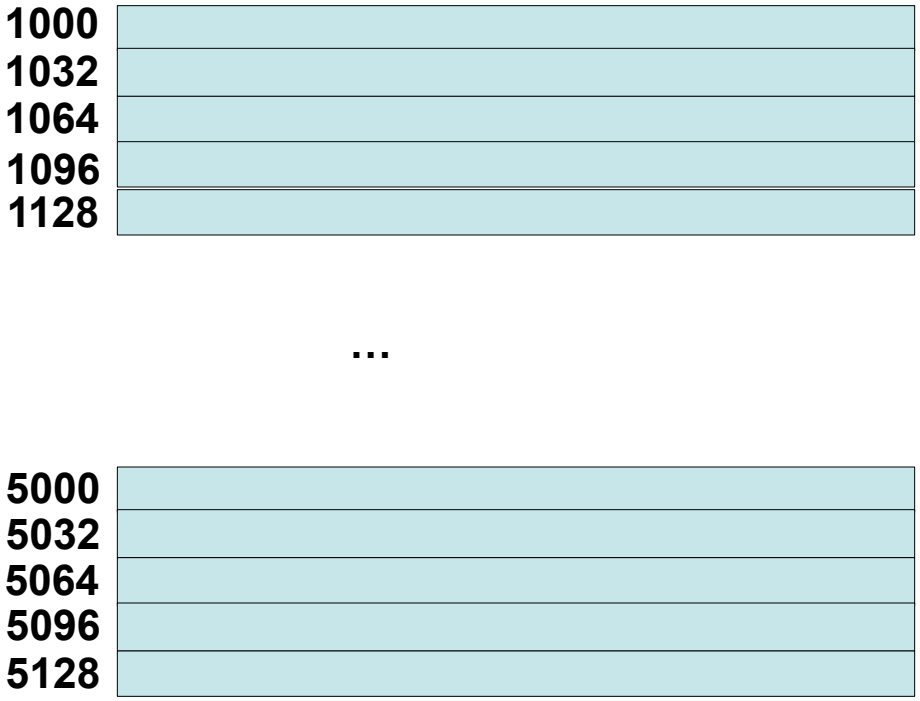
Si accede a una locazione di memoria RAM tramite il suo indirizzo, in tempo costante.



Rappresentazione in memoria

Volendo astrarre dalla singola macchina e le sue caratteristiche tecnologiche, vediamo la memoria esclusivamente come una Random Access Memory, cioè una sequenza finita di locazioni contigue di memoria, ciascuna in grado di contenere una sequenza finita di bit, o parola di memoria.

Esempio, in cui una parola è di 32 bits



Ogni posizione nella sequenza è individuata da un indirizzo di memoria, normalmente espresso con un numero intero positivo, tramite il quale si accede al contenuto di quella cella di memoria.

Modello di calcolo

Essenzialmente in un modello di calcolo dobbiamo poter dire quali operazioni hanno costo costante.

Sono di costo costante, per esempio:

1. **assegnazioni di valori alle variabili**
2. **chiamate di funzioni (metodi)**
3. **valutazione di espressioni aritmetiche**
4. **confronto tra due numeri**
5. **l'accesso all'i-simo elemento di una lista di interi**
6. **rientro da una chiamata**

Calcolo complessità

Se chiamiamo primitive le operazioni di costo costante, potremo dire che per valutare la complessità di un algoritmo su un certo input si dovrà contare il numero di esecuzioni delle operazioni primitive.

Il conteggio così ottenuto sarà prossimo al reale tempo di calcolo in una qualsiasi implementazione dell'algoritmo.

Questo valore in generale dipende dalla dimensione dell'input.

Analisi: complessità degli algoritmi

L'analisi della complessità di un algoritmo (corretto!!) consiste nel valutare la quantità di tempo (e/o di spazio) di memoria richieste per la sua esecuzione

Serve per poter

1. **predire il comportamento su input di grandezza qualsiasi, (senza implementarlo)**
2. **scegliere il giusto algoritmo/struttura dati in un dato contesto confrontando soluzione diverse, dal punto di vista del consumo delle risorse (tempo, spazio)**
3. **decidere se la nostra soluzione è la migliore possibile per un problema**
4. **decidere se il nostro algoritmo è ottimale per il nostro problema (cioè che non è possibile dimostratamente trovarne uno migliore)**

Conclusione

L'analisi della complessità di tempo o spazio di una algoritmo si fa avendo in mente un modello di calcolo in cui

1. **la memoria è di tipo RAM**
2. **le operazioni primitive sono di costo costante.**

Si calcola approssimatamente il numero delle esecuzioni delle operazioni di costo costante, le primitive, in funzione della dimensione dell'input. (p.e. per una lista il numero dei suoi elementi)