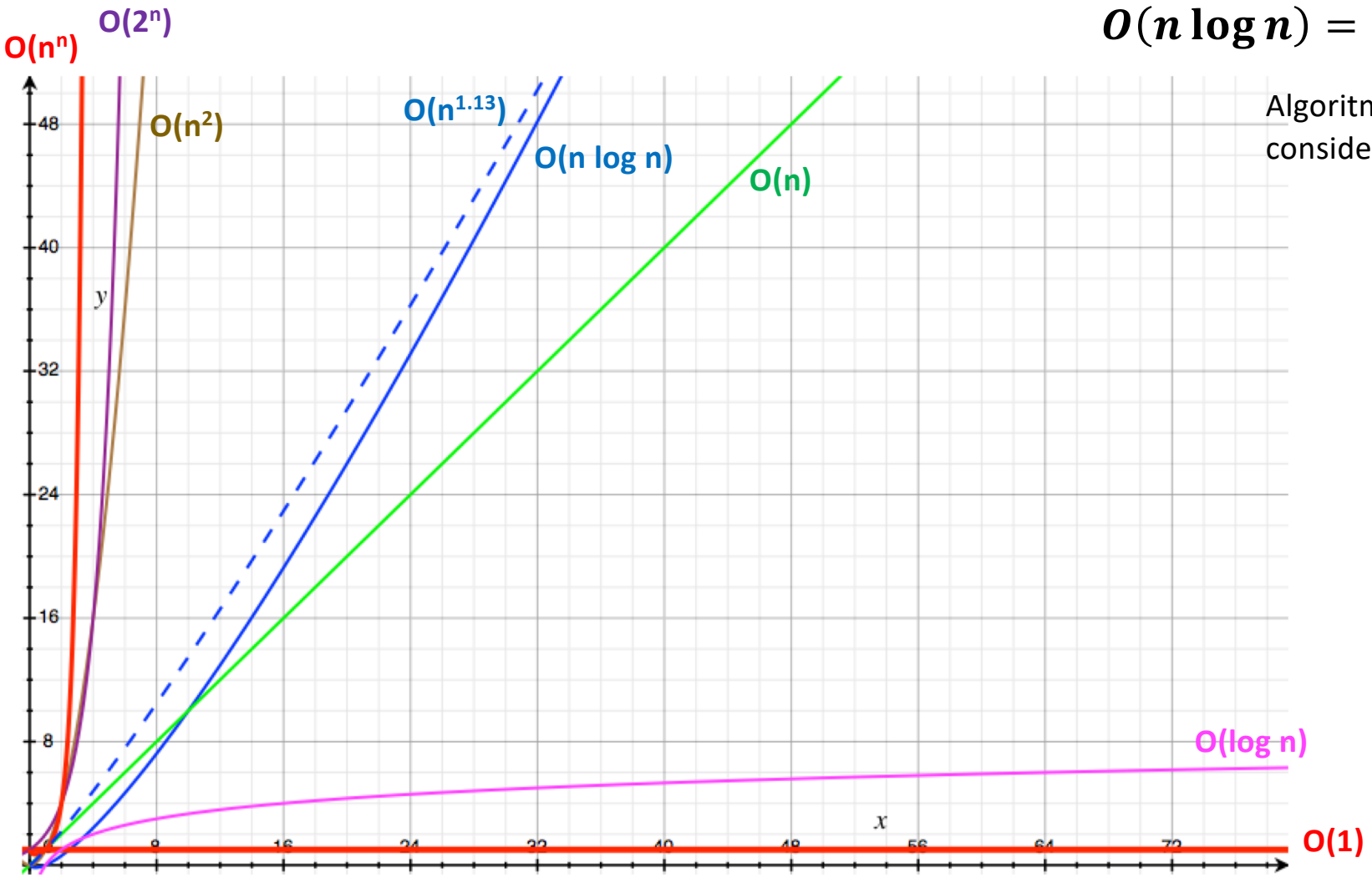
Teoria della complessità

Liliana Lo Presti

Outline

- Famiglie di problemi
- Problemi non-deterministici
- Soddisfacibilità
- Teorema di Cook

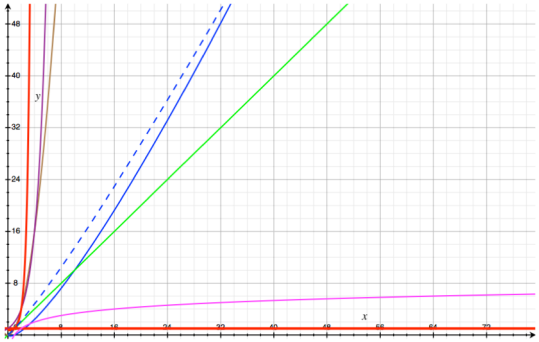
Complessità computazionale



$$O(n \log n) = O(n^{1+\varepsilon})$$

Algoritmi O(nlog n) sono considerati polinomiali!

Famiglie di problemi



Problemi

Trattabili

è possible fornire algoritmi efficienti e sono considerati “facili”. Es.: Ordinamento

Richiedono un tempo di esecuzione polinomiale

Difficili

Gli unici algoritmi conosciuti richiedono un tempo di esecuzione esponenziale

Facile verificare soluzione

Richiedono un tempo polinomiale per verificare la soluzione

Difficile verificare soluzione

Tipi di problemi

Problemi $P \subseteq I \times S$

Di **decisione**:
Richiedono una risposta binaria, quindi S={0,1}
Es.
- Se elemento appartiene a dizionario;
- Se un grafo è connesso;
- Se esiste un cammino tra due nodi con costo inferior a k

Di **ricerca**:
Richiedono una soluzione s in S
Es.:
- Trovare un cammino tra due nodi
- Trovare l'elemento mediano

Di **ottimizzazione**:
Si vuole trovare la soluzione migliore s* nell'insieme delle possibili soluzioni S
Es.:
- Trovare il cammino di costo minimo tra due nodi

Molti problem sono modellati come problem di decisione
I Problemi di ottimizzazione spesso non sono più difficili dei problem di decisione

Classe di complessità e tipi di problemi

- **Definizione:** una classe di complessità è un insieme di problemi risolvibili con le stesse risorse di calcolo

- Un problema P può esser pensato come una relazione tra istanze di ingresso I e soluzioni S ovvero $P \subseteq I \times S$

- Possiamo anche considerare un predicato $p: I \times S \rightarrow \{0,1\}$ definite come segue

$$\forall (i,s) \in I \times S, p(i,s) = \begin{cases} 1, & (i,s) \in P \\ 0, & (i,s) \notin P \end{cases}$$

La classe P

- È la classe dei problem risolvibili in tempo polinomiale nella dimensione dell'istanza di ingress n

$$P = \{Time(n^c)\}_{c=0}^{\infty}$$

- Analoga definizione esiste per lo spazio, ovvero la classe Pspace

$$PSpace = \{Space(n^c)\}_{c=0}^{\infty}$$

Algoritmi di ordinamento? Algoritmi di visita?

La classe ExpTime

- È la classe dei problem risolvibili in tempo esponenziale nella dimensione dell'istanza di ingress n

$$ExpTime = \{Time(2^{n^c})\}_{c=0}^{\infty}$$

- Un algoritmo in P può avere accesso al più ad un numero polinomiale di celle di memoria quindi

$$P \subseteq PSpace$$

- Inoltre, (supponendo per semplicità 2 stati per le n^c possibili celle)

$$PSpace \subseteq ExpTime$$

- Tuttavia, sappiamo che $P \subset ExpTime$ (si dimostra con Time hierarchy Theorem)

Certificati e verificabilità

- Per alcuni problem decisionali, oltre che una risposta binaria, è possible restituire anche un oggetto **che permetta di giustificare l'eventuale risposta positive**
- Tale risposta è denominato ***certificato***
- Es. Per la connetività di un grafo basta considerare un albero ricoprente
- Dato un certificato, vogliamo verificarlo. Ciò richiede tempo computazionale. Il costo per la verifica del certificate esprime in un certo senso la complessità del problema

La classe NP

- Informalmente, è la classe dei problemi decisionali che ammettono certificati verificabili in tempo polinomiale

- Es.: Il problema della connettività appartiene alla classe NP perchè possiamo verificare il certificato (cioè che un albero sia un albero ricoprente) in tempo polinomiale

Non determinismo

- In un algoritmo *deterministico*, il risultato di uno step è determinate univocamente dallo stato della computazione (input e step precedent. Es. Calcolo del minimo, selectionSort…)

- In un algoritmo *non determini*stico, l'algoritmo prende decisioni randomiche. Quindi, la computazione non-deterministica risulta in un albero delle possibili esecuzioni in cui, ad ogni nodo, l'algoritmo prende una decisione in modo non tdeterministico.

- Algoritmo *determinato* restituisce lo stesso valore ogni volta che viene eseguito sulla stessa istanza.
- Es*. Il Quicksort randomizzato è un algoritmo non-deterministico ma determinato*

La classe NP

- È la classe dei problem decisionali risolvibili da un algoritmo non-deterministico in tempo polinomiale nella dimensione dell'istanza di ingress n

$$NP = \{NTime(n^c)\}_{c=0}^{\infty}$$

- Possiamo immaginare due fasi:
  - Nella fase non-deterministica, l'algoritmo attraverso decisioni non deterministiche produce un certificate
  - Nella fase deterministica, il certificate viene verificato in tempo polinomiale

- Un algoritmo deterministico è un caso particolare di un algoritmo non-deterministico, quindi vale che:

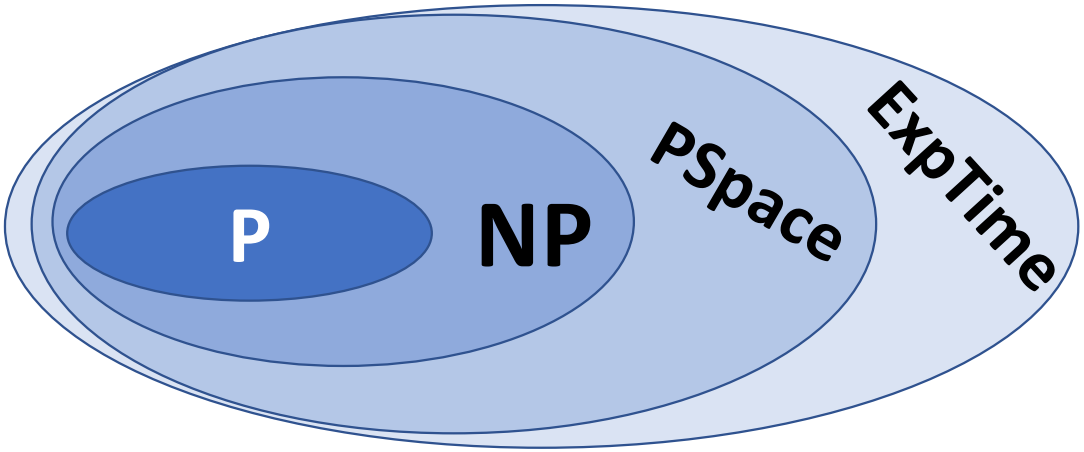
$$P \subseteq NP$$

Gerarchia delle classi di problemi

- Poichè $P \subseteq PSpace$, ed un algoritmo NP ha una fase di verifica che deve essere polinomiale, deve essere $NP \subseteq PSpace$
- In generale:

$$P \subseteq NP \subseteq PSpace \subseteq ExpTime$$

- Si congettura che le inclusioni siano proprie



Riducibilità

- Dati due problem decisionali, è possible ridurre polinomialmente uno dei due problem nell'altro *se esiste un algoritmo con costo polinomiale* che permette di trasformare una istanza positive del primo problema in una istanza positiva del secondo problema e una istanza negativa del primo problema in una negativa del secondo problema

- Perchè è importante? Perchè se trovo un algoritmo risolutivo per il secondo problema riesco a risolvere anche il primo.

NP-Hard e NP-Complete

- Un problema decisionale T si dice NP-Hard se ogni problema Q in NP è riducibile polinomialmente a T. Un problema NP-Hard è tanto difficile quanto il più difficile dei problem in NP

- Un problema decisionale T si dice NP-Complete se T è in NP ed è NP-Hard. Questa classe rappresenta l'insieme dei problem NP più difficile. Esistono centinaia di problemi in questa classe. Es. Commesso viaggiatore, il problema dello zaino, colorazione dei vertici…

- Di conseguenza: se esistesse un problema NP-Complete che fosse nella classe P, allora potremmo concludere che P = NP

Riepilogo

- Famiglie di problem
- Classi computazionali
- N vs NP