Complessità computazionale

— ○ —

Algoritmi probabilistici e Quantum Computing

Piero A. Bonatti

Università di Napoli Federico II

Laurea Magistrale in Informatica

Tema della lezione

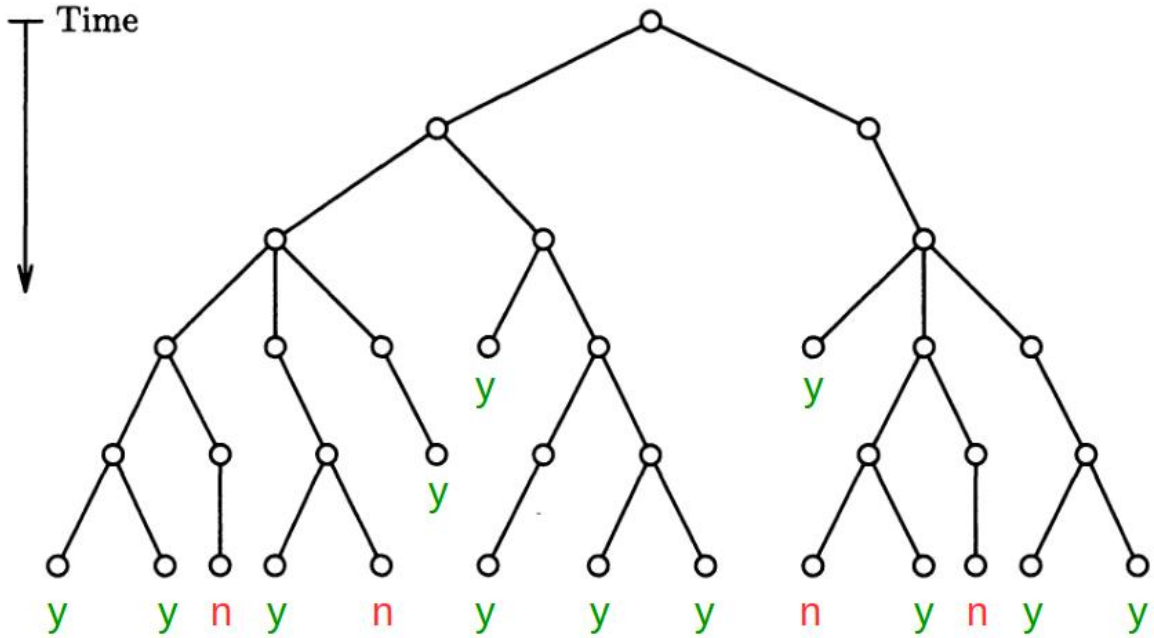
Algoritmi probabilistici

Il nondeterminismo potrebbe essere sfruttato anche in altri modi:

- Accettazione "a maggioranza"
  - se la maggioranza dei run termina in "yes" allora si accetta
  - Ma come verificarlo senza esplorare un numero esponenziale di run?
- Accettazione probabilistica
  - Generiamo una computazione *a caso* della MdT nondeterministica
  - Se la probabilità che la risposta ("yes" o "no") sia corretta è maggiore della probabilità che non lo sia...
  - ...allora ripetendo il procedimento un numero sufficiente di volte avremo la risposta corretta *con probabilità alta a piacere*

Tema della lezione

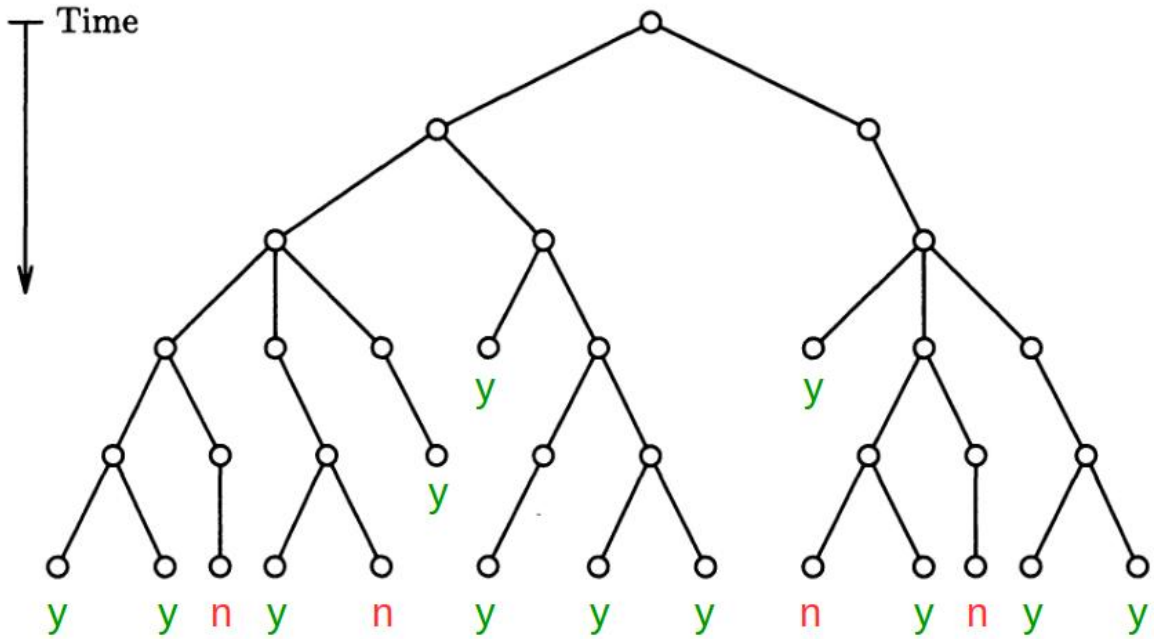
Esempio 1



- Per questa MdT e questo input, 12 computazioni su 16 restituiscono "yes"
- Accettazione "a maggioranza": "si"
- Ma come verificare la maggioranza senza esplorare un numero esponenziale di computazioni?

Tema della lezione

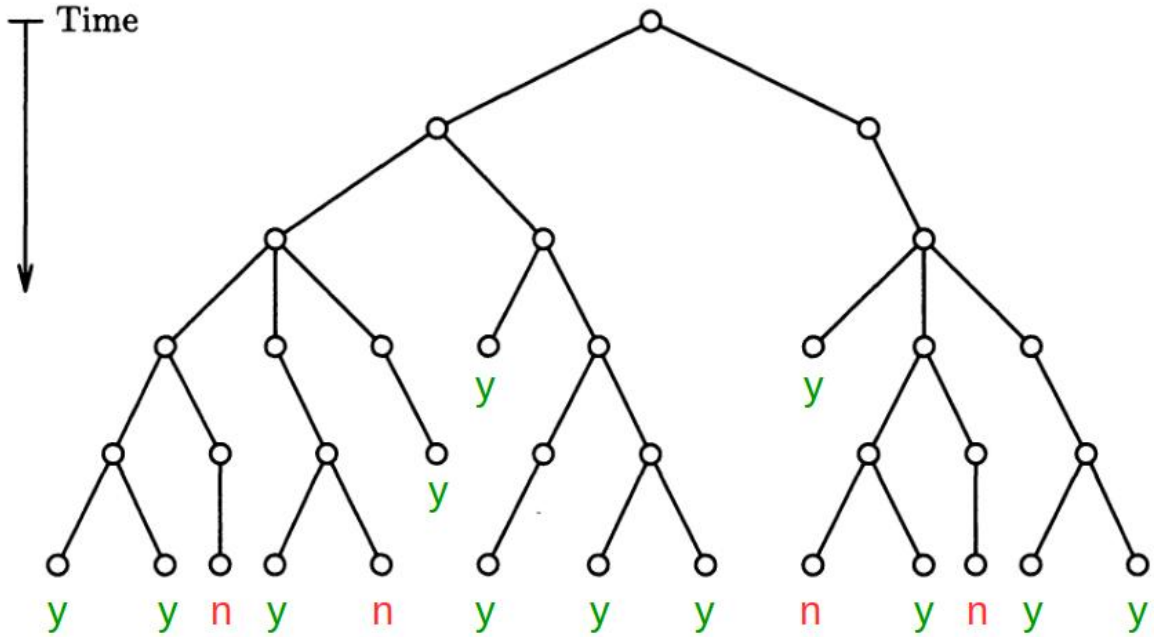
Esempio 2



- Verifica probabilistica: con selezione dei run equiprobabile
  - la probabilità che al primo tentativo esca "yes" è $12/16 = 75\%$
  - la probabilità che in 3 tentativi escano almeno 2 "yes" è $> 84\%$
  - la probabilità che in 5 tentativi escano almeno 3 "yes" è $> 89\%$
- Quanti tentativi servirebbero per superare - che so - il 99%?
  - polinomiali o esponenziali?

Tema della lezione

Esempio 3



- Caso più generale: ogni run ha una probabilità diversa di essere generato (succederà coi computer quantistici)
- Se ogni run che termina in "no" ha probabilità 5 volte maggiore di ogni run che termina in "yes",
- allora la probabilità di estrarre un "no" al primo tentativo diventa il 62% (e cresce col numero di tentativi). Non più a maggioranza

Tema della lezione

- Formalizzeremo queste intuizioni
  - introducendo le classi **PP**, **BPP**, **BQP**
- **BQP** formalizza la classe di problemi risolubili efficientemente da computer quantistici
  - che descriveremo brevemente
- Studieremo le relazioni tra queste classi e quelle più tradizionali per prevedere quali problemi verranno resi "facili" dai computer quantistici e quali no

MdT nondeterministiche "standardizzate"

- In questo ramo della Complexity Theory, si adottano alcune assunzioni sulle MdT nondeterministiche
  - non cambiano la potenza espressiva delle MdT
  - non cambiano la complessità dei problemi (almeno da **P** in su)

1. Le assumiamo *precise*
   - tutte le computazioni terminano esattamente nello stesso numero di passi
2. ogni stato ha esattamente 2 scelte nondeterministiche

- Chiaramente questo facilita il calcolo della probabilità dei run
  - gli alberi di computazione sono alberi binari perfettamente bilanciati

La classe **PP**

Definizione

$L \in \mathbf{PP}$ sse esiste una MdT N (standardizzata) che termina in tempo polinomiale e tale che

$$x \in L \text{ sse } N \text{ accetta } x \text{ “a maggioranza”}$$

ovvero più della metà delle computazioni di N per l'input x terminano in “yes”

La classe **PP**

- Un problema completo per **PP**:

MAJSAT

Given a boolean expression ϕ, è vero che la maggioranza dei suoi truth assignment la soddisfa?

Relazioni con **NP**

Teorema 11.3

NP $\subseteq$ PP

- Si pensa che il contrario non sia vero
- Considerate il certificato naturale per una istanza di MAJSAT con n proposizioni atomiche:
  - $2^{n-1} + 1$ truth assignments che la soddisfano...

Dimostrazione che **NP** $\subseteq$ **PP** (Teorema 11.3)

- **(cenni)** Prendete un qualunque $L \in \mathbf{NP}$, quindi deciso da qualche MdT N in tempo polinomiale $p(n)$, e trasformate N in una N' che accetta a maggioranza.

1. Introducete un nuovo stato iniziale con 2 scelte nondeterministiche:
   - eseguire N
   - continuare con scelte nondeterministiche binarie per $p(|x|)$ passi, terminando sempre in "yes"
2. Ciascuna scelta iniziale porta a $2^{p(|x|)}$ computazioni
3. N' accetta a maggioranza sse almeno $2^{p(|x|)} + 1$ run accettano x, sse almeno 1 run di N accetta

QED

PP vs PSPACE

Teorema

PP $\subseteq$ PSPACE

- **Prova (cenni)**: Con spazio polinomiale, possiamo simulare tutti i run di una MdT nondeterministica N
  - esplorando l'albero delle computazioni *depth-first*
  - riutiizzando lo spazio dei run precedenti
- Su due ulteriori nastri possiamo contare tutti i run e quelli che accettano x
  - spazio polinomiale (perchè?)
- Infine verificare che il secondo contatore è $>$ della metà del primo è possibile in tempo (e spazio) polinomiale

QED

Altre proprietà di **PP**

Proposizione

PP è chiusa rispetto ai complementi

- Basta *quasi* invertire "yes" e "no". Sia $L \in \mathbf{PP}$. L viene deciso "a maggioranza" da una MdT N in tempo polinomiale
- Caso facile: $x \in \bar{L}$ sse x *non* viene accettato da N sse la maggior parte dei run termina con "no"
- Quindi la N' ottenuta da N scambiando gli stati finali accetta $\bar{L}$ a maggioranza
- Problema: se x rigettato perchè "yes" e "no" in parità. Si può risolvere più in generale (vedere la nota 11.5.17)

QED

La classe **BPP**
(Bounded error Probabilistic Polynomial time)

Problema pratico con **PP**

- Come già osservato, per verificare se una MdT nondeterministica N accetta un input x
  - dovremmo contare quanti run ha
  - dovremmo contare quanti run accettano x
- Il primo conteggio è facile se N è standardizzata
  - $2^{f(|x|)}$, se f è la funzione di complessità di N
- Per il secondo, non sappiamo che simulare i run di N
  - che sono in mumero esponenziale
- Per questo si considera una condizione di accettazione diversa
  - soggetta ad errori

  ma sempre basata sul fatto che la maggioranza dei run accetta i membri del linguaggio

Accettazione probabilistica

L'idea di base

- Assumiamo nuovamente che la MdT N sia standardizzata e che tutti i suoi run siano equiprobabili
- Se $x \in L$ sse la maggioranza dei run di N accetta x allora
  - invece di generare tutti i run, posso generarne k a caso
  - al crescere di k la probabilità che la maggioranza di loro accetti/rigetti x si avvicina a 1 sse la maggioranza dei run accetta/rigetta
- Resterà una probabilità non nulla che la maggioranza dei run generati casualmente non rifletta la maggioranza effettiva
  - in quel caso otteniamo una conclusione errata circa $x \stackrel{?}{\in} L$

Qual è la probabilità di errore?

Accettazione probabilistica

L'idea di base – 2

- Aumentando il numero di tentativi k possiamo ridurre la probabilità di errore a piacimento
- Ma si può dimostrare che se ci si accontenta di una accettazione a maggioranza stretta, allora per ridurre la probabilità di errore sotto una soglia prefissata ϵ può servire un k esponenzialmente grande
- La soluzione è ovvia:
  - Richiedere una chiara maggioranza

  Questo porta direttamente a **BPP**

La classe **BPP**

Definizione

$L \in$ **BPP** sse esiste una MdT N nondeterministica (standardizzata) che opera in tempo polinomiale tale che

- se $x \in L$ allora almeno $\frac{3}{4}$ dei run di N accettano x
- se $x \notin L$ allora almeno $\frac{3}{4}$ dei run di N rigettano x

- **Nota**: la scelta di $\frac{3}{4}$ è convenzionale: qualunque altra soglia s t.c. $\frac{1}{2} < s \leq 1$ produrrebbe la stessa classe
- L'importante è che sia espressa come *percentuale* dei run
  - il suo valore aumenta all'aumentare dei run

Natura convenzionale della soglia

- Supponiamo che N decida L con una maggioranza $\frac{1}{2} + \epsilon \neq \frac{3}{4}$
- Si può dimostrare che dopo $2k+1$ tentativi la probabilità che la maggioranza dei risultati dia un risultato errato è

$$e^{-2\epsilon^2 k}$$

- Quindi per ridurla a meno di $1/4$ (come richiesto da **BPP**) basta iterare N per $2k+1$ volte dove

$$k = \left\lceil \frac{\ln 2}{\epsilon^2} \right\rceil$$

  Notare che k è costante (dipende solo da N)
- Di conseguenza la nuova macchina con soglia a $\frac{3}{4}$ rallenta solo di un fattore costante
- Con calcoli analoghi si mostra che la soglia desiderata di errore in **BPP** si raggiunge in un numero polinomiale di passi

Relazioni tra **BPP** e le altre classi

Proposizione

$\mathbf{BPP} \subseteq \mathbf{PP}$

- Banale: l'accettazione a maggioranza qualificata di **BPP** implica quella a maggioranza semplice

Proposizione

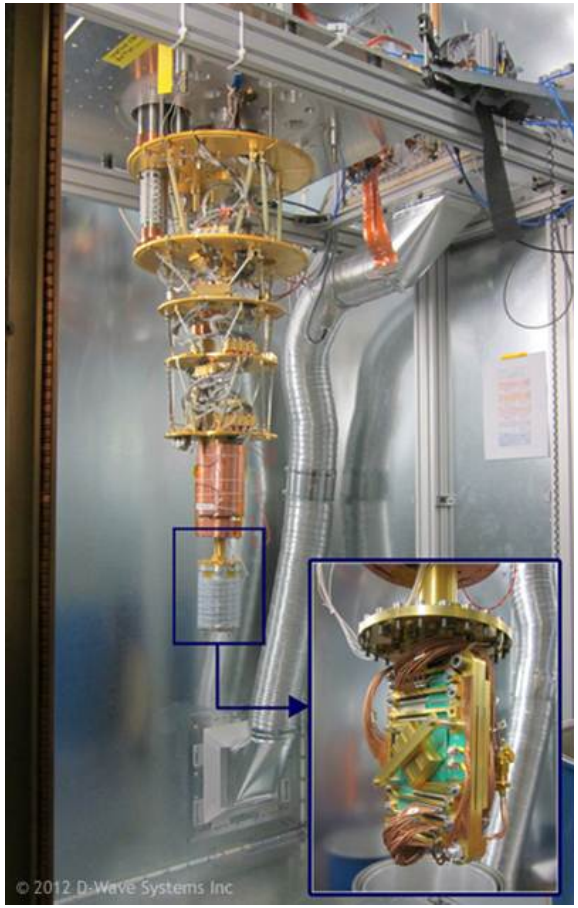
$\mathbf{BPP} = co\mathbf{BPP}$

- Banale: le condizioni di accettazione e rigetto sono simmetriche, basta invertire "yes" e "no"...
- Non è noto se $\mathbf{BPP} \subseteq \mathbf{NP}$ ma si può dimostrare:

Teorema 17.12

$\mathbf{BPP} \subseteq \Sigma_2\mathbf{P}$ (quindi anche $\mathbf{BPP} \subseteq \Sigma_2\mathbf{P} \cap \Pi_2\mathbf{P}$)

I computer quantistici una micro-introduzione



I qubit

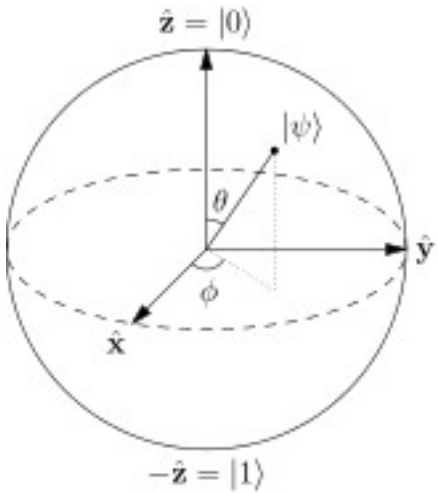
I qubit (o bit quantistici)

- Sono l'analogo del bit nel calcolo quantistico
  - seguono le leggi della meccanica quantistica
- Possono trovarsi "simultaneamente" nei due stati 0 e 1
  - in meccanica quantistica si parla di *sovrapposizioni di stati*
  - con probabilità diversa, in generale
- Si parla di stati *puri*: la probabilità di essere in stati diversi da 0 o 1 è nulla

I qubit

- I qubit hanno una rappresentazione vettoriale
  - due vettori, denotati $|0\rangle$ e $|1\rangle$ corrispondono a 0 e 1 tradizionali
  - un qubit può anche essere una combinazione lineare di $|0\rangle$ e $|1\rangle$

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$



α e β possono essere numeri complessi

$$\alpha = \cos\left(\frac{\theta}{2}\right)$$

$$\beta = e^{i\phi}\sin\left(\frac{\theta}{2}\right)$$

- α e β sono detti *probability amplitudes*
- Quando il valore di $|\psi\rangle$ viene misurato, lo stato del qubit collassa a $|0\rangle$ con probabilità $|\alpha|^2$ e a $|1\rangle$ con probabilità $|\beta|^2$
  - $|\alpha|^2 + |\beta|^2 = 1$

Il calcolo quantistico è probabilistico

- Queste proprietà rendono i qubit potenzialmente adatti all'implementazione di algoritmi probabilistici
- I qubit possono essere elaborati mediante *gate quantistici* che generalizzano i classici *and*, *or*, *nand*...
  - matematicamente: sono delle matrici che trasformano i valori dei qubit in ingresso
- l'output sarà un array di qubit la cui misura dovrà restituire la risposta esatta con probabilità "nettamente" maggiore di $1/2$
  - così ripetendo il calcolo k volte ridurremo la probabilità di errore alla soglia desiderata

Realizzazione fisica dei qubit

Physical support	Name	Information support	$\|0\rangle$	$\|1\rangle$
Photon	Polarization encoding	Polarization of light	Horizontal	Vertical
	Number of photons	Fock state	Vacuum	Single photon state
	Time-bin encoding	Time of arrival	Early	Late
Coherent state of light	Squeezed light	Quadrature	Amplitude-squeezed state	Phase-squeezed state
Electrons	Electronic spin	Spin	Up	Down
	Electron number	Charge	No electron	One electron
Nucleus	Nuclear spin addressed through NMR	Spin	Up	Down
Optical lattices	Atomic spin	Spin	Up	Down
Josephson junction	Superconducting charge qubit	Charge	Uncharged superconducting island (Q=0)	Charged superconducting island (Q=2 extra Cooper pair)
	Superconducting flux qubit	Current	Clockwise current	Counterclockwise current
	Superconducting phase qubit	Energy	Ground state	First excited state
Singly charged quantum dot pair	Electron localization	Charge	Electron on left dot	Electron on right dot
Quantum dot	Dot spin	Spin	Down	Up

Il problema della coerenza

- Gli stati "puri" (combinazioni lineari di $|0\rangle$ e $|1\rangle$) "degradano" rapidamente (*decoherencing*)
  - diventando un mix di più stati
  - a causa delle interazioni termodinamiche con l'ambiente
  - questo impedisce il corretto calcolo quantistico
- La soluzione attualmente consiste nell'isolare il processore quantistico, ad es.
  - temperature prossime allo zero assoluto
  - schermando i campi magnetici
- I qubit rimangono coerenti per qualche minuto nei casi migliori

Il problema della coerenza

- Temperatura del chip: $-273°C$
  - (0,015 Kelvin sopra lo zero assoluto)
- Schermatura riduce campo magnetico a 1/50.000 di quello terrestre
- Capacità di memoria dichiarata: 1000 qubit. Decoherencing = ???
- 192 linee da-verso il chip per I/O e controllo
- Non è chiaro quanto sia programmabile
  - gli "algoritmi" sono circuiti
  - molti prototipi eseguono un solo algoritmo

Quantum Turing Machines (QTM)

Quantum Turing Machines

- Sono più convenienti dei circuiti per certe analisi di complessità
  - e sarebbero più facili da programmare dei circuiti quantistici...
- È stato provato che hanno la stessa potenza delle *famiglie uniformi* di circuiti quantistici
  - Esiste un algoritmo "semplice" che dato l'input costruisce il circuito che lo deve analizzare
  - Così si adatta la dimensione del circuito a quella dell'input
- Poichè stati e nastro sono memorizzati con qubit, la QTM si troverà in ogni istante in una *sovrapposizione* di configurazioni
- L'output (probabilistico) si ottiene misurando la configurazione finale

Quantum Turing Machines

Definizioni formali

Definizione (QTM)

È una quadrupla $\langle K, \Sigma, \delta, s \rangle$ dove K, Σ, s sono come nelle MdT ma

$$\delta : K \times \Sigma \to \bar{\mathbb{C}}^{K \times \Sigma \times \{\leftarrow, \rightarrow, -\}}$$

- $\bar{\mathbb{C}}$ è un insieme di numeri complessi *calcolabile velocemente*
  - $\alpha \in \bar{\mathbb{C}}$ sse esiste una MdT che dato k calcola il k-esimo bit delle parti reale e immaginaria di α in tempo polinomiale in k
- $\bar{\mathbb{C}}^{K \times \Sigma \times \{\leftarrow, \rightarrow, -\}}$ è l'insieme delle funzioni

$$f : K \times \Sigma \times \{\leftarrow, \rightarrow, -\} \to \bar{\mathbb{C}}$$

$\delta(q, \sigma)$ associa una *probability amplitude* ad ogni azione applicabile in (q, σ)

Quantum Turing Machines

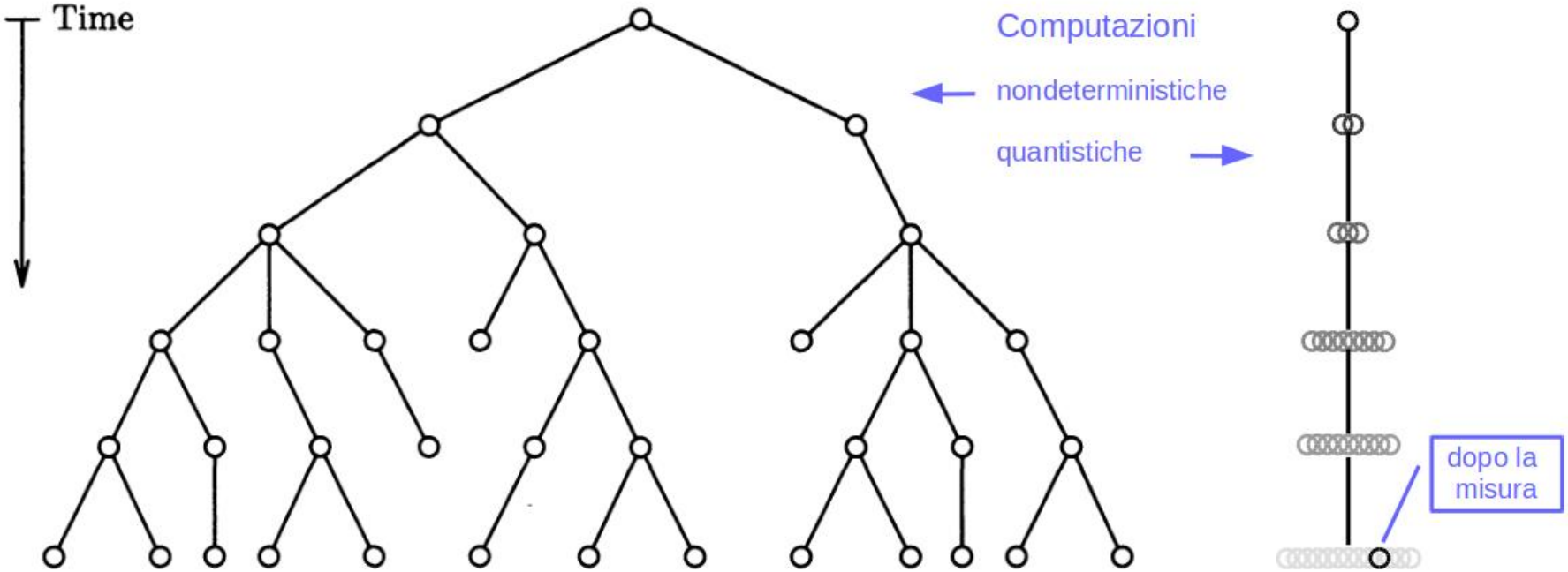
Configurazioni e transizioni

- Configurazioni di una QTM Q: come per le MdT: $c = (q, w, u)$
- In ogni istante Q si trova in una *sovrapposizione di configurazioni*

$$\sum_c \alpha_c \cdot |c\rangle \tag{1}$$

- Se inizialmente Q è nella configurazione $c_0 = (s, w\sigma, u)$, al passo successivo sarà nella sovrapposizione (1) tale che:
  - c spazia sulle configurazioni ottenute applicando a c_0 una qualunque delle possibili azioni $a = (q', \sigma', \{\leftarrow, \rightarrow, -\})$
  - $\alpha_c = \delta(s, \sigma)(a)$ (la probability amplitude dell'azione)
- Iterando, si ottengono le sovrapposizioni ai passi successivi

Quantum Turing Machines



- La misura della sovrapposizione finale fa collassare la configurazione quantistica su una delle configurazioni tradizionali finali
- rispettando le probabilità associate a ogni configurazione

Quantum Turing Machines

Misura della configurazione

- Formalmente, se la QTM si trova nella sovrapposizione

$$\sum_i \alpha_i \cdot |c_i\rangle$$

- allora la probabilità di osservare $|c_i\rangle$ è $|\alpha_i|^2$
- se succede, allora la nuova sovrapposizione diventa $|c_i\rangle$
- quindi la misurazione *deve* essere fatta *solo alla fine*
- È quindi importante che la QTM sia *precisa*, cioè che tutte le sue computazioni abbiano esattamente la stessa lunghezza

La classe **BQP** (Bounded-error Quantum Polynomial-time)

Definizione

$L \in$ **BQP** se e solo se esiste una QTM Q tale che per ogni input x la probabilità di osservare una configurazione con "yes" nella misura finale è

- $\geq \frac{2}{3}$ se $x \in L$
- $\leq \frac{1}{3}$ se $x \notin L$ (x viene rigettato con probabilità $\geq \frac{2}{3}$)

- Notare l'analogia con **BPP**
- Cambia il modo in cui vengono attribuite le probabilità alle configurazioni finali
  - in **BPP** sono equiprobabili, in **BQP** dipendono dalla δ di Q

La classe **BQP**

scelta della soglia

- La soglia di $\frac{2}{3}$ è convenzionale, come per **BPP**
- Dato un limite desiderato all'errore, si può ottenere una risposta che lo soddisfa
- iterando Q un numero polinomiale di volte

Relazioni tra **BQP** e le altre classi di complessità

Teorema

BPP $\subseteq$ **BQP**

- Non sorprende, vista la maggiore generalità delle distribuzione di probabilità sulle configurazioni finali delle QTM
- Si *pensa* che **BPP** $\neq$ **BQP** (ennesima questione aperta)
- Corollario: **P** $\subseteq$ **BQP** (perchè **P** $\subseteq$ **BPP**)
- Si pensa che **NP** $\not\subseteq$ **BQP** (!)

Relazioni tra **BQP** e le altre classi di complessità

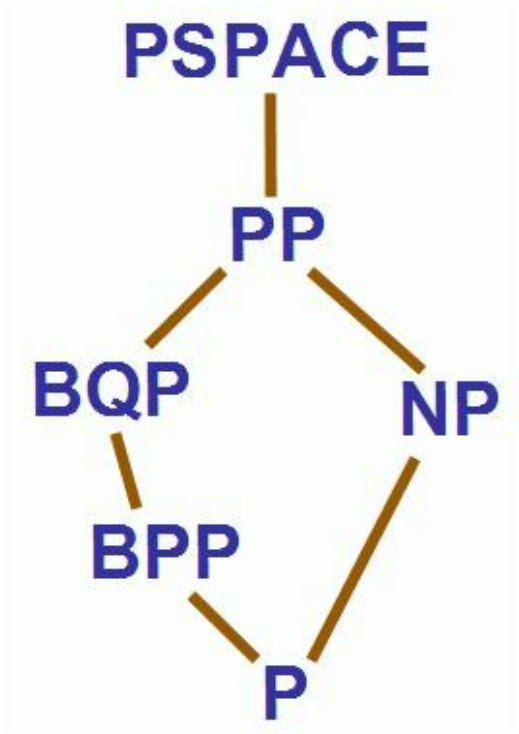
Teorema

BQP $\subseteq$ **PP**

- Di nuovo, non sorprendente visto che la condizione di accettazione di **PP** è più debole (non richiede maggioranze di 2/3)
- Corollari/conseguenze:
  1. **BQP** $\subseteq$ **PSPACE** (perchè **PP** $\subseteq$ **PSPACE**)
  2. Se **BQP** $\neq$ **PSPACE** allora **P** $\neq$ **PSPACE** (almeno altrettanto difficile da dimostrare)

Ricapitolando

Cosa le QTM (non) possono fare in tempo polinomiale



Problemi con speedup interessanti

Nota: da soluzioni precise a errore limitato

- Calcolo della Trasformata di Fourier
  - per spazi n-dimensionali passa da $O(n \log n)$ a $O(\log^2 n)$
- Fattorizzazione di interi
  - da tempo esponenziale a polinomiale con *algoritmo di Shor*
  - usa l'algoritmo per la trasformata di Fourier
- Ricerca in una lista non ordinata S (*algoritmo di Grover*):

  data $f : S \to \{0, 1\}$ trovare un $x \in S$ tale che $f(x) = 1$

  - notare similitudine con i problemi in **NP**
  - se il costo di $f(x)$ è $O(m)$, allora il costo totale passa da $O(m \cdot |S|)$ a $O(m \cdot \sqrt{|S|})$

Speedup garantito per **NP**

- Lo speedup quadratico della ricerca nelle liste non ordinate si riflette sui problemi in **NP**

Teorema

Ogni problema in **NP** può essere risolto in tempo $O(\sqrt{2^n}) = O(2^{n/2})$

- **Prova per** SAT (cenni): Usare l'algoritmo di Grover per trovare un truth assignment che soddisfa la formula data
- Si può verificare se la risposta è corretta in tempo deterministico polinomiale
- Se necessario, iterare k volte. Facendo crescere k polinomialmente si può abbassare la probabilità di errore a piacere

Note sullo speedup per **NP**

- Il calcolo è $O(2^{n/2})$ volte più veloce (speedup esponenziale)
- Tuttavia il costo resta esponenziale ($O(2^{n/2})$)
- Sebbene i problemi in **NP** possano essere risolti in spazio polinomiale, la soluzione di Grover richiede di *materializzare lo spazio delle soluzioni*
  - per SAT, la lista di *tutti i* 2^n *truth assignment*
  - spazio esponenziale
  - al momento questo speedup ha interesse solo teorico

Note sullo speedup per la fattorizzazione

- Se ne avremo una implementazione "utilizzabile in pratica", i principali crittosistemi a chiave pubblica non saranno più sicuri
  - RSA, Diffie-Hellman, Elliptic curve cryptography
  - di conseguenza i protocolli SSL e SSH
  - fattorizzando si ottiene la chiave privata da quella pubblica

  Potremo calcolare la chiave privata da quella pubblica
- Invece i crittosistemi simmetrici (a chiave privata) come AES e Twofish e le hash functions come SHA sembrano "immuni"[1]
  - l'algoritmo di Grover accelera gli attacchi a forza bruta (ricerca nello spazio delle chiavi) di un fattore $2^{n/2}$
  - ma il costo resta $O(2^{n/2})$
  - raddoppiando la lunghezza della chiave si ripristina il livello di sicurezza (costo aggiuntivo (de)cifratura solo polinomiale)

[1] https://cryptome.org/2016/01/CNSA-Suite-and-Quantum-Computing-FAQ.pdf, Sez. "Quantum computing threats"

Altre note sull'algoritmo di Schor

- Esegue la fattorizzazione in tempo polinomiale con errore limitato a piacere
- Quindi si legge frequentemente che la fattorizzazione *appartiene a* **BQP**
- Dove sta l'errore?
- È un *errore di tipo*
  - **BQP** è una classe di linguaggi (problemi decisionali)
  - la fattorizzazione restituisce un insieme di numeri (è un *function problem*!)
- Pare che nessuno si sia mai preso la briga di correggere questo aspetto. Si veda la discussione su

Materiale di riferimento

- Papadimitriou: Parte III
  - Capitolo 11, parte del paragrafo 11.2 (solo **PP** e **BPP**)
- Quantum computing
  - L'articolo `divulgativo-cs191_qc4.pdf`
  - Chi fosse interessato ad ulteriori dettagli formali li può trovare nell'articolo `Bernstein-Vazirani.pdf`

Esercizi (1)

1. Se $L \in \mathbf{BPP}$, è possibile ridurlo a QSAT_i? E a $\overline{\mathrm{QSAT}_i}$?
2. Si può risolvere GEOGRAPHY con un algoritmo a errore limitato in tempo polinomiale?
3. Se posso risolvere L con errore limitato in tempo polinomiale mediante un computer quantistico, posso risolverlo anche con un calcolatore tradizionale usando un algoritmo probabilistico che restituisce la risposta corretta con probabilità $> 1/2$?
4. E posso ridurre l'errore a piacimento in tempo polinomiale?

Esercizi (2)

1. È possibile risolvere $\text{TSP}(\text{D})$ in tempo polinomiale con errore limitato mediante un algoritmo quantistico?
2. Se un problema L è risolubile in tempo polinomiale con un algoritmo quantistico a errore limitato, quanto spazio occorre per risolverlo con un algoritmo tradizionale?
3. Dire se SCHÖNFINKEL-BERNAYS SAT è risolubile in tempo polinomiale con un algoritmo quantistico a errore limitato
4. Quanto bisogna allungare le chiavi per proteggere AES dagli attacchi basati sull'algoritmo di Grover? E per proteggere RSA?