

CAPITOLO 1

ALGORITMO: SEQUENZA FINITA DI PASSI INTERPRETABILI DA UN' ESECUTORE

VENGONO CLASSIFICATI IN BASE A DIVERSE ASPETTI:

PASSI CONTEMPORANEAMENTE:

- ↳ SEQUENZIALI UN PASSO ALLA VOLTA
- ↳ PARALLELI PIÙ PASSI PER VOLTA

METODO DI SCELTA:

- ↳ DETERMINISTICI SCELTA IN BASE AD UN CRITERIO PRE-DETERMINATO
- ↳ PROBABILISTICI SCELTA CASUALE
- ↳ NON DETERMINISTICI ESPLOANO TUTTE LE SCELTE CONTEMPORANEAMENTE

TERMINAZIONE:

- ↳ CORRETTO RISPETTO AD UN PROBLEMA, SE AD OGNI I PRODUCE UNA S DESIDERATA

I STANZA DATI DI INGRESSO, S SOLUZIONI, P PROBLEMA

$$P \subseteq I \times S$$

PROBLEMA

- ↳ DECISIONE RISPOSTA BINARIA
- ↳ RICERCA GENERICA S BASATA SU I
- ↳ OTIMIZZAZIONE SOLUZIONE OTIMA IN BASE AD UN CRITERIO

DATO UN P POSSONO ESSERCI DIVERSI ALGORITMI
CHE LO RISOLVONO, VENGONO COMPARATI IN BASE A:

- TEMPO DI CALCOLO
- , MEMORIA
- BANDA TRASMISSIONE

STRUTTURA DATI: UN' INSIEME DI DATI LOGICAMENTE CORRELATI
E OPPORTUNAMENTE MEMORIZZATI, PER I quali SONO
DEFINITI DEGLI OPERATORI DI COSTRUZIONE, SELEZIONE E MANIPOLAZIONE

CAPITOLO 2

NON È SEMPRE POSSIBILE TROVARE UN ALGORITMO CHE RISOLVE UN DATO PROBLEMA

PROBLEMA

- ↳ INDECIDIBILE SE NON AMMETTE ALGORITMI CORRETTI
- ↳ DECIDIBILE SE AMMETTE ALGORITMI CORRETTI

CLASSICO PROBLEMA INDECIDIBILE È QUELLO DELLA TERMINAZIONE (TURING 1937):

DATO UN ALGORITMO E UNA QUALUNQUE ISTANZA DEI SUOI DATI DI INPUT, STABILIRE SE L'ESECUZIONE TERMINA O NO

PROBLEMI DECIDIBILI

- TRATTABILI: RISOLVIBILI IN TEMPO POLINOMIALE
- INTAINSECAMENTE INTRATTABILI: NON RISOLVIBILI IN TEMPO POLINOMIALE NEHMEND DA UN ALGORITMO NON DETERMINISTICO.

PROBLEMI DI DECISIONE TRATTABILI

- ↳ CONTENUTI IN P SE RISOLTI CON ALGORITMO DETERMINISTICO
- ↳ CONTENUTI IN NP SE RISOLTI CON ALGORITMO NON DETERMINISTICO

$$P \subseteq NP$$

DATI DUE PROBLEMI DI DECISIONE $P \subseteq I_1 \times \{0,1\}$

$\in P_2 \subseteq I_2 \times \{0,1\}$, P_1 È RIDUCIBILE A P_2

SE ESISTE UN ALGORITMO DETERMINISTICO TRATTABILE
CHE CALCOLA $f: I_1 \rightarrow I_2 : H, s \in \{0,1\}$

$$(i, s) \in P_1 \Leftrightarrow (f(i), s) \in P_2$$

ALLORA:

- SE $P_2 \in P$ ANCHE $P_1 \in P$
- SE $P_1 \notin P$ ANCHE $P_2 \notin P$

TEOREMA DI Cook (1971): OGNI PROBLEMA NP È RIDUCIBILE IN TEMPO POLINOMIALE AL PROBLEMA
DELLA SODDISFACIBILITÀ

DATA UN'ESPRESSIONE LOGICA IN FORMA NORMALE
CONGUNTIVA, STABILIRE SE ESISTE UN ASSEGNAZIONE
DI VALORI DI VERA ALLE SUO VARIAZIONI CHE LA
rende vera.

UN PROBLEMA NP È DETTO NP-COMPLETO SE IL
PROBLEMA DELLA SODDISFACIBILITÀ È RIDUCIBILE
AD ESSO IN TEMPO POLINOMIALE.

CAPITOLO 3

GLI ALGORITMI CHE RISOLVONO LO STESSO PROBLEMA SONO CONFRONTATI IN BASE ALLA LORO EFFICIENZA. IL TEMPO DI ESECUZIONE VIENE ESPRESO COME UNA FUNZIONE IN DIMENSIONE DEI DATI DI INGRESSO: $T(n)$

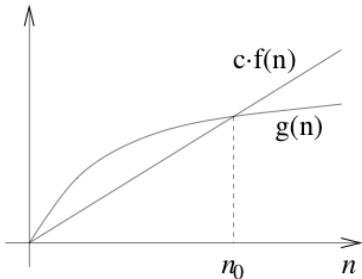
VIENE CALCOLATO IN 3 DIVERSI CASI:

- CASO OTIMO
- CASO PESSIMO
- CASO MEDIO

PER OTENERE UN PARAMETRO DI VALUTAZIONE SI CONFRONTA L'ANDAMENTO CON DELLE FUNZIONI NOTE, IN 3 DIVERSI MODI:

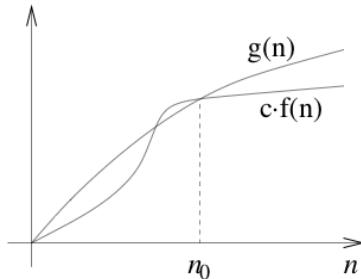
LIMITE ASINTOTICO
SUPERIORE

O



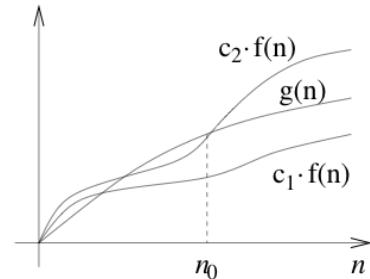
LIMITE ASINTOTICO
INFERIORE

Ω



LIMITE ASINTOTICO
STRETTO

Θ



CAPITOLO 4

UN'ARRAY È UNA STRUTTURA STATICA E OMogenea.

ALGORITMO DI VISITA

FUNZIONAMENTO :

ATTRAVERSA LINEARMENTE L'ARRAY

COMPLESSITÀ:

$$O(n)$$

ALGORITMO RICERCA LINEARE ARRAY

FUNZIONAMENTO :

ATTRAVERSA LINEARMENTE L'ARRAY FINO A TROVARE
L'ELEMENTO RICERCATO.

COMPLESSITÀ :

• CASO OTTIMO : $O(1)$

IL PRIMO ELEMENTO È QUELLO RICERCATO

• CASO PESSIMO : $O(n)$

L'ULTIMO ELEMENTO È QUELLO RICERCATO / NON C'E'

ALGORITMO RICERCA BINARIA:

Search for 47

0	4	7	10	14	23	45	47	53
---	---	---	----	----	----	----	----	----

FUNZIONAMENTO:

RICHIEDE UN'ARRAY ORDINATO,
DIVIDE A META' L'ARRAY
SE L'ELEMENTO CENTRALE

È QUELLO DA CERCARE SI FIA IL CASO OTTIMO,
IN CASO CONTRARIO SI CONTINUA A DIVIDERE LA
PARTE PIÙ VICINA AL VALORE.

COMPLESSITÀ:

- CASO OTTIMO: $T(n) = O(1)$
- CASO PESSIMO: $T(n) = O(\log n)$

OLTRE CHE PER LA COMPLESSITÀ, GLI ALGORITMI
POSSENO ANCHE ESSERE CONFRONTATI IN BASE A:

- STABILITÀ: SE NON ALTERA L'ORDINE RELATIVO DI ELEMENTI DELL'ARRAY AVENUTA STESSA CHIAVE.
- AZIONE SUL POSTO: SE LA DIMENSIONE DELLE STRUTTURE AUXILIARI È INDEPENDENTE DAL NUMERO DI DATI.

INSERT SORT

È UN'ALGORITMO SUL POSTO

FUNZIONAMENTO :

SELEZIONA IN ORDINE DI

MEMORIZZAZIONE GLI ELEMENTI

E LI FA SCORRERE VERSO SINISTRA

FINCHÉ NON SONO NAGLIUM O UGUALI DEGLI ELEMENTI.

CASO OTTIMO :

ARRAY GIÀ ORDINATO : $T(n) = O(n)$

CASO PESSIMO :

ARRAY INVERSAMENTE ORDINATO : $T(n) = O(n^2)$

SELECT SORT

È UN'ALGORITMO SUL POSTO E STABILE

FUNZIONAMENTO :

PRENDERE IL VALORE MINIMO

DELLA SEQUENZA, SCAMBIALO CON

QUELLO ATTUALE, E RICOMINCIARE CON LA SOTTO SEQUENZA
RIMASTA.

5 3 4 1 2

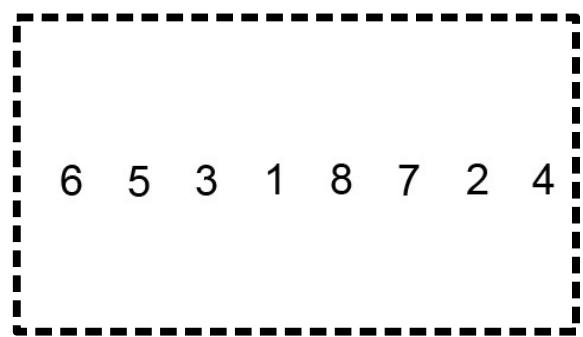
COMPLESSITÀ : NON ESSENDO DIPENDENTE DALLA
DISPOSIZIONE PREMESSA DEGLI ELEMENTI HA SEMPRE
COMPLESSITÀ : $T(n) = O(n^2)$

BUBBLE SORT

È UN'ALGORITMO SUL POSTO E STABILE

FUNZIONAMENTO :

PRENDE 2 A 2 GLI ELEMENTI,
CONFRONTARLI E INVERTIRLI SE IL SINISTRO È
PIÙ GRANDE DEL DESTRO.



COMPLESSITÀ : NON ESSENDO DIPENDENTE DALLA
DISPOSIZIONE PREGRESSA DEGLI ELEMENTI HA SEMPRE

$$\text{COMPLESSITÀ} : T(n) = O(n^2)$$

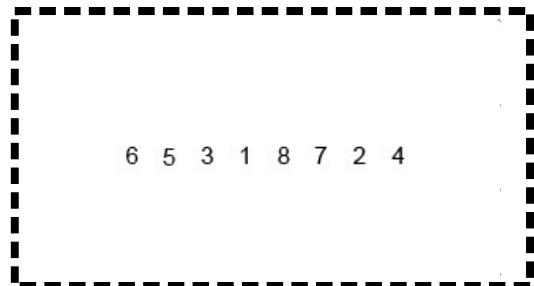
MERGE SORT

È UN'ALGORITMO STABILE

NON SOL POSTO PERCHÉ SFUOCIA UN SECONDO ARRAY

FUNZIONAMENTO :

DIVIDE RICORSIVAMENTE L'ARRAY IN SOTTOARRAY
ORDINA PEZZO PER PEZZO RIPETENDO ALLA FINE
ALL'ARRAY CON LUNGHEZZA INIZIALE



COMPLESSITÀ : NON ESSENDO DIPENDENTE DALLA
DISPOSIZIONE PREGRESSA DEGLI ELEMENTI HA SEMPRE

$$\text{COMPLESSITÀ} : T(n) = O(n \cdot \log n)$$

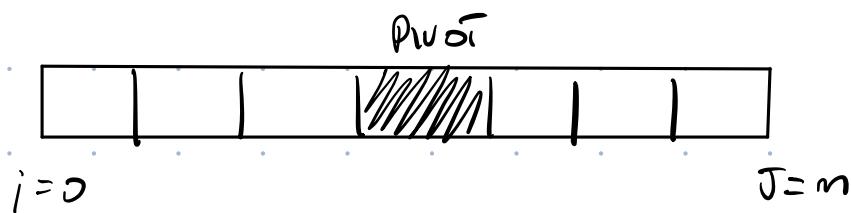
QUICKSORT

È UN ALGORITMO SU C POSSO

FUNZIONAMENTO:

6 5 3 1 8 7 2 4

VIENE PRESO UN NUMERO CHILO "PIVOT" E DUE INDICI.
UN'INDICE PARTE DALL'ESTREMO DESTRO, L'ALTRO DALL'ESTREMO SINISTRO



IL PIVOT DI DESTRA SCARNE FIN QANDO NON TROVA
UN VALORE \geq pivot.

IL PIVOT DI SINISTRA SCARNE FIN QANDO NON TROVA
UN VALORE $<$ pivot.

TROVATI I DUE VALORI, VENTONO SCAMBIATI.

SI CONTINUA COSÌ FINCHÉ I DUE INDICI NON SI
INCROCIANO; A QUEL PUNTO SI DIVIDE L'ARRAY
IN DUE PARTI CON CENTRO IL PIVOT È
SI RIPETE RIUSCIVAMENTE SU QUESTE DUE.

COMPLESSITÀ:

CASO OTTIMO:

PIVOT VALORE MEDIANO

$$T(n) = O(n \cdot \log n)$$

CASO PESSIMO

PIVOT VALORE MIN O MAX

$$T(n) = O(n^2)$$

CASO MEDIO:

$$T(m) = O(m \cdot \log n)$$

HEAPSORT

SUL POSIZ.

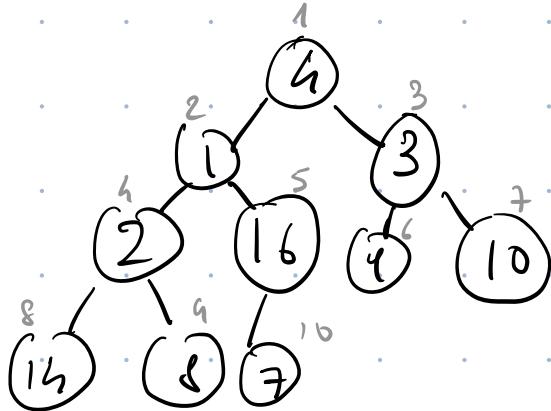
FUNZIONAMENTO:

L'ALLEG. VIENE TRASFORMATO

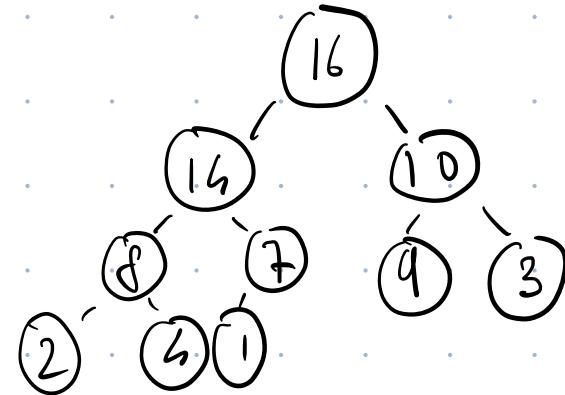
IN UN HEAP.

1	2	3	4	5	6	7	8	9	10
4	1	3	2	16	9	10	14	8	7

HEAP



MAX HEAP



SCAMBIANO LA RADICE CON L'ULTIMO ELEMENTO, CHE STA AL GIUSTO POSTO.

A QUESTO PUNTO RIPETIAMO L'ALBERO IN UN MAX HEAP E RIPETIAMO IL PROCESSO FIN QUANDO L'HEAP NON È UN VUOTO;

COMPLESSITÀ:

NON DIPENDE DALLA PREDISPOSIZIONE DELL'ARRAY QUINDI $O(m \cdot \log n)$

CAPITOLO 5

SI DICE LISTA UNA TRIPPA $L = (E, t, S)$:

- E È UN INSIEME DI ELEMENTI
- $t \in E$ È DETTO TESTA
- S È UNA RELAZIONE BINARIA SU E , $S \subseteq E \times E$

UNA LISTA È DETTA ORDINATA SE LE CHIAVI CONTENUTE NEL SUOI ELEMENTI SONO DISPOSTE CREANDO UNA RELAZIONE DI ORDINE TOTALE.

VISITA

COMPLESSITÀ : $O(n)$

RICERCA

COMPLESSITÀ :

- CASO OTTIMO : $O(1)$
- CASO PESSIMO : $O(n)$

INSERIMENTO / RIMOZIONE

COMPLESSITÀ :

- CASO OTTIMO : $O(1)$
- CASO PESSIMO : $O(n)$

UNA CODA È UNA LISTA GOVERNATA IN
BASE AL PRINCIPIO FIFO.

INSERIMENTO : $O(1)$

RIMOZIONE : $O(1)$

UNA PILA È UNA LISTA GOVERNATA IN
BASE AL PRINCIPIO LIFO.

INSERIMENTO : $O(1)$

RIMOZIONE : $O(1)$

CAPITOLO 6

SI DICE ALBERO CON RADICE, UNA TRIPLOTA $T = (N, r, B)$

- N È UN INSIEME DI NODI
- r È LA RADICE $r \in N$
- B È UNA RELAZIONE BINARIA SU N

SI DICE ALBERO LIBERO UN ALBERO

SI DICE GRADO DELL'ALBERO IL NUMERO MASSIMO DI FIGLI DI UN SUO NODO.

SI DICE ALTEZZA DELL'ALBERO IL NUMERO MASSIMO DI NODI CHE SI ATTRAVERSANO NEL PERCORSO DI T CHE VA DALLA RADICE ALLA FOGLIA PIÙ DISTANTE

SI DICE AMPIETTA DELL'ALBERO IL NUMERO MASSIMO DI NODI CHE SI TROVA ALLO STESSO LIVELLO.

UN ALBERO DI GRADO $d > 1$ È ALTEZZA b , PUÒ CONTENERE UN MASSIMO NUMERO DI NODI PARI A:

$$\frac{d^b - 1}{d - 1}$$

OGNI ALBERO NON BINARIO È TRASFORMABILE IN
TALE CON LA TRASFORMAZIONE FRATELLO - FIGLIO.

VISITA

ANTICIPATA, SIMMETRICA, POSTICIPATA.

COMPLESSITÀ: $T(m) = O(m)$

D'ORA IN AVANTI CONSIDEREMO L'ALBERO COME BINARIO DI RICERCA,
E GLI ALGORITMI MANterranno QUESTA PROPRIETÀ

RICERCA / INSERIMENTO / RIMOZIONE

COMPLESSITÀ:

- CASO OTTIMO: $O(1)$
- CASO PESSIMO: $O(\log m)$
- CASO MEDIO: $O(m)$

BILANCIAMENTO

UN' ALBERO BINARIO DI RICERCA È DETTO:

- **BILANCIATO PERFETTAMENTE** SE PER OGNI NODO IL NUMERO DI NODI DEL SOTTOALBERO SINISTRO È DI QUELLO DESTRO DIFFERISCONO AL PIÙ DI 1.
- **AUL-BILANCIATO** L'ALTEZZA DEL SUO SOTTOALBERO SINISTRO È DI QUELLO DESTRO DIFFERISCONO AL PIÙ DI 1.

• ROSSO - NERO COORDINANDO I NODI

- LA RADICE È NERA
- LA SENTINELLA È NERA.
- SE UN NODO È ROSSO (FIGLI SONO NERI)
- PER OGNI NODO, IL NUMERO DI FIGLII NERI DA ATTRAVERSARE PER RAGGIUNGERE UNA SENTINELLA È LO STESO.

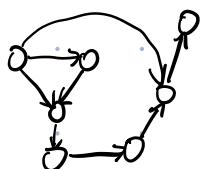
CAPITOLO 7

GRAFO = (V, E)

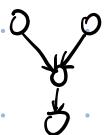
INDIRETTO }
DIRECTO }

PESATO SE $\exists w: E \rightarrow \mathbb{R}$

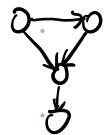
SOTTOGRAFO PARTE DEI VERTICI
SOTTOGRAFO INDOTTO PARTE DEI VERTICI + ARCHI
GRAFO TRASPOSTO STESSI VERTICI MA ARCHI INVERTITI



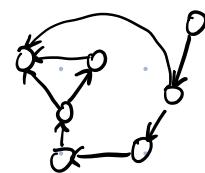
GRAFO



SOTTOGRAFO



SOTTOGRAFO
INDOTTO



GRAFO
TRASPOSTO

VERTEICE

GRADO USCENTE NUMERO ARCHI USCENTI
GRADO ENTRANTE NUMERO ARCHI ENTRANTI
GRADO USCENTE + ENTRANTE
TERMINALE USCENTE = 0, ENTRANTE > 0.
INIZIALE USCENTE > 0, ENTRANTE = 0
ISOLATO USCENTE = 0, ENTRANTE = 0

G È DIRECTO COMPLETO SE TUTTI I VERTICI SONO CONNESSI TRA ZO

PERCORSO

SEMPLICE SE TUTTI I VERTICI CHE PASSA SONO DIVERSI
CICLO ULTIMO VERTICE COINCIDE COL PRIMO

GRAFO DIRECTO

CONNESO SE $\forall v_1, v_2 \exists$ PERCORSO DA v_1 A v_2
FORTEMENTE CONNESO SE \exists PERCORSO DA v_1 A $v_2 \in V_1$ A V_2
CIClico SE CONTIENE UN CICLO

GRAFO INDIRETTO : GLI ARCHI SONO BIDIREZIONALI

RAPPRESENTAZIONE

► MATRICE DI ADIACENZA

SPAZIO IN MEMORIA $O(|V|^2)$

ESISTENZA ARCO $O(1)$

CONVENIENTE PER GRAFI STATICI

► LISTA DI ADIACENZA

SPAZIO IN MEMORIA $O(|V| + |E|)$

ESISTENZA ARCO $O(|V| + |E|)$

CONVENIENTE PER GRAFI SParsi

IN C

STRUTTURA

{ VALORE

VERTEICE SUCC

LISTA ARCHI

STRUTTURA

{ PESO

VERTEICE ADIACENTE

ARCO SUCC

PROBLEMI

► ORDINAMENTO TOPOLOGICO

► COMPONENTI FORTEMENTE CONNESSE

► ALBERO RICOPRENTE MINIMO

► PERCORSO PIÙ BREVE

► VISITA

► RICERCA

VISITA

→ **AMPIETTA** : I VERTICI VENGONO ATTRAVERSATI IN ORDINE DI DISTANZA CRESCENTE DAL VERTICE DI PARTENZA. ITERATIVO E STRUTTURA UNA CODA.

• COMPLESSITÀ : $T(|V|, |E|) = O(|V| + |E|)$

→ **PROFONDITÀ** : ELABORA IL VALORE DEL VERTICE IN CUI SI È GIUNTI, ALLO STESSO MODO SI VISITANO POI QUELLI ADIACENTI. RICORSIVO E UTILIZZA UNA PILA.

• COMPLESSITÀ : $T(|V|, |E|) = O(|V| + |E|)$

RICERCA

SE OPPORTUNAMENTE MODIFICATI I DUE ALGORITMI DI VISITA POSSONO ESSERE UTILIZZATI PER LA RICERCA.

COMPLESSITÀ :

• CASO OTTIMO : $T(|V| + |E|) = O(1)$

• CASO PESSIMO : $T(|V| + |E|) = O(|V| + |E|)$

ORDINAMENTO TOPOLOGICO PER GRAFI DIRETTI E ACICLICI

SI PUÒ FARE USO DEGLI ALGORITMI DI PROFONDITÀ, CONVIENE CREARE DIRETTAMENTE UNA LISTA IN CUI VENGONO INSERITI MAN MANO CHI VENGONO COLORATI DI NERO.

COMPLESSITÀ : $T(|V| + |E|) = O(|V| + |E|)$

ALGORITMI PER ALBERO RICOPRENTE MINIMO

→ ALGORITMO DI KRUSKAL

COSTRUISCE L'ALBERO PARTENDO

DA TANTI ALBERI QUANTI

SONO I SINGOLI VERTICI,

AD OGNI PASSO INCLUDE

L'ARCO DI PESO MINIMO CHE COLLEGÀ DUE
DIVERSI ALBERI LIBERI.

COMPLESSITÀ: $T(|V| + |E|) = O(|E| \cdot \log |V|)$

→ ALGORITMO DI PRIM

PARTENDO DA UN'ALBERO LIBERO

COSTITUITO DA UN SINGOLO

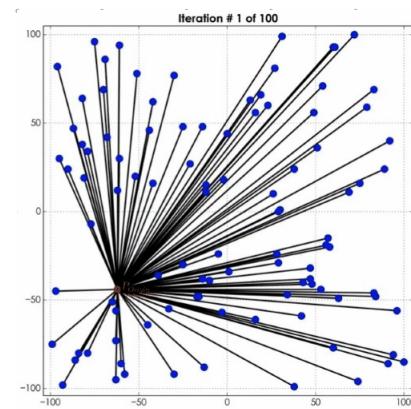
VERTICE, AD OGNI PASSO

INCLUDE L'ARCO DI PESO

MINORE CHE COLLEGÀ UN

VERTICE DELL'ALBERO LIBERO, AD UNO CHE

NON NE FA PARTE.



COMPLESSITÀ: $T(|V| + |E|) = O(|E| \cdot \log |V|)$

PERCORSO PIÙ BREVE

- SE IL GRAFO CONTIENE UN GRAFO DI PESO NEGATIVO POTREBBE DIVERGERE A $-\infty$

VENGONO CONSIDERATI SOLO I PERCORSI SIMPLICI.

CIASCU^NO DEI 3 ALGORITMI CALCOLA I PERCORSI PIÙ BREVI DA UN VERTICE DATO E TUTTI GLI ALTRI VERTICI, PRODUCENDO UN'ALBERO CON RADICE NEL VERTICE DATO.

I GRAFI VENGONO INIZIALIZZATI CON UN ALGORITMO LA CUI COMPLESSITÀ È: $O(|V|)$

GLI ALGORITMI SFRUTTANO UN ALGORITMO DI RISOLUZIONE DEGLI ARCHI CON COMPLESSITÀ: $O(1)$

ALGORITMO DI BELLMAN-FORD

CONSENTE LA PRESENZA DI ARCHI DI PESO NEGATIVO E PROCEDÈ ALLA RIDUZIONE SISTEMATICA DELLA STIMA DELLA DISTANZA MINIMA SFRUTTANDO IL FATTO CHE UN PERCORSO MINIMO NON PUÒ CONTENERE PIÙ DI $|V|-1$ ARCHI.

RESTITUISCE I SE NON CI SONO CICLI DI PESO NEGATIVO, ALTRIMENTI 0,

COMPLESSITÀ: $O(|V| \cdot (|V| + |E|))$

VARIANTE PIÙ EFFICIENTE SU GRAFI DI RETI SENZA CICLI: $O(|V| + |E|)$

ALGORITMO DI DIJKSTRA

SI APPLICA SOLO A GRAFI DI PESO POSITIVO

COMPLESSITÀ : $O(|V| \cdot \log |V| + |E|)$

TECNICHE ALGORITMICHE

DIVIDE ET IMPERA DALL'ALTO VERSO IL BASSO

- DIVIDERE L'ISTANZA DEI DATI DI INGRESSO
 - RISOLVERE RICORSIVAMENTE IL PROBLEMA
 - COMBINARE LE SOLUZIONI DELLE SOTTOISTANZE
- APPLICATION: MERGESORT

PROGRAMMAZIONE DINAMICA DAL BASSO VERSO L'ALTO

- IDENTIFICARE DEI SOTOPROBLEMI
 - PREDISPORRE UNA TABELLA PER LE SOLUZIONI DEI SOTOPROBLEMI
 - CALCOLARE LE SOLUZIONI NELLA TABELLA
 - RESTITUIRE LE SOLUZIONI DEL PROBLEMA ORIGINALE
- APPLICATION: ALGORITMO ITERATIVO FATTORIALE

TECNICA GOLOSA PER PROBLEMI DI OTIMIZZAZIONE

- SOLUZIONE OTTIMA GLOBALE PUÒ ESSERE OTTENUTA INCREMENTALMENTE COMPIENDO QUANDO RICHIESTO SCELTE LOCALMENTE OTTIME

APPLICATION: KRUSKAL, PRIM, BELLMAN-FORD

TENTATIVI E REVOCHE PROBLEMI SENZA REGOLA FISSA

- PROCEDERE PER TENTATIVI TORNANDO INDietro QUANDO UN TENTATIVO FALLISCE

COMPLESSITÀ : $O(m^2)$

APPLICATIONS :

8.4 * Tecnica per tentativi e revoca

- Esistono problemi per i quali non si riesce a trovare una regola fissa di computazione per determinare la soluzione. Per tali problemi non si può che **procedere per tentativi**, ritornando indietro sui propri passi ogni volta che il tentativo corrente fallisce.
- Concettualmente, un algoritmo per tentativi e revoca visita in profondità il suo albero di decisione, il quale comprende tutte le scelte che possono presentarsi durante l'esecuzione. Di conseguenza, la **complessità asintotica nel caso pessimo** (fallimento di tutti i tentativi) è **inevitabilmente esponenziale**.