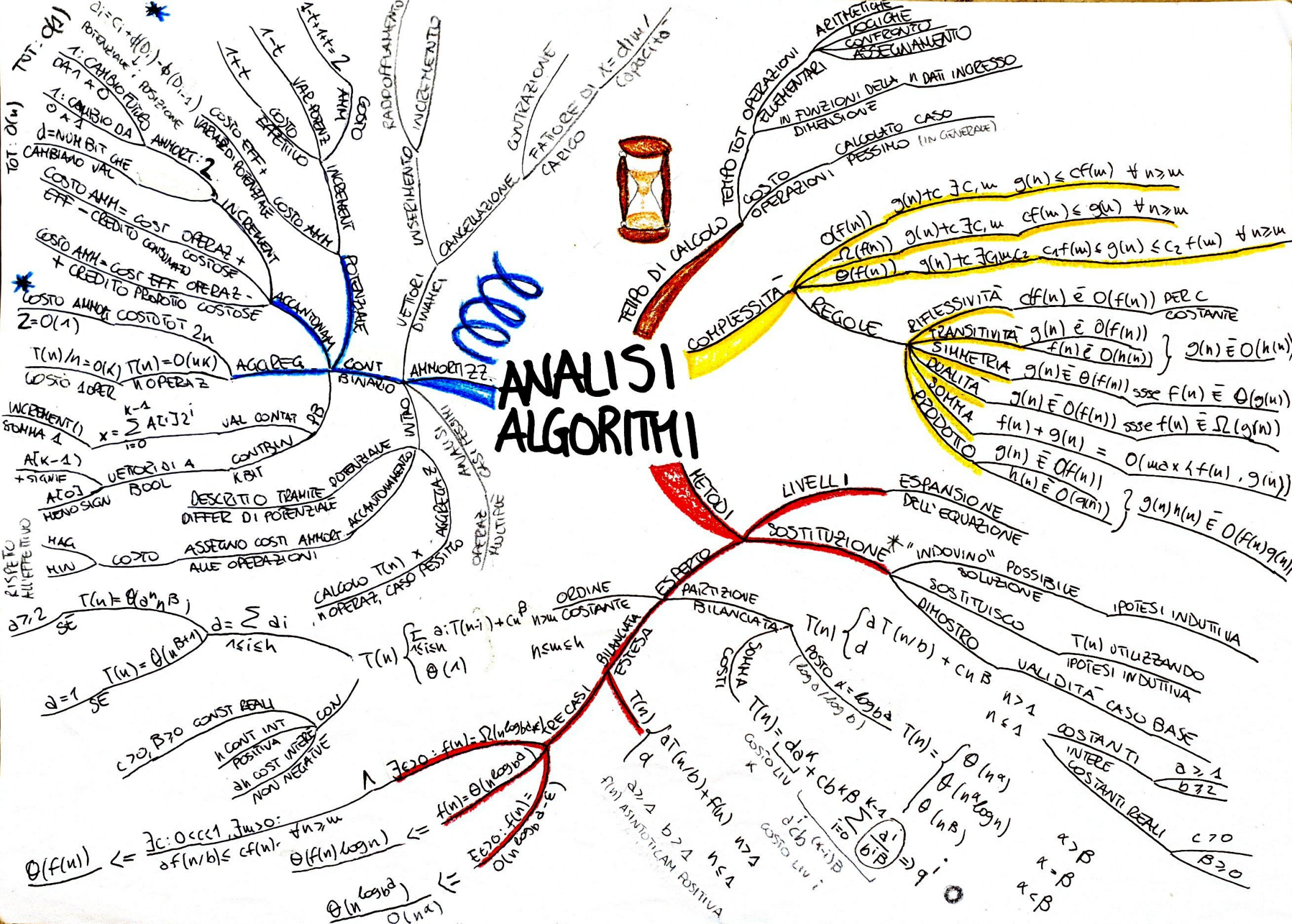
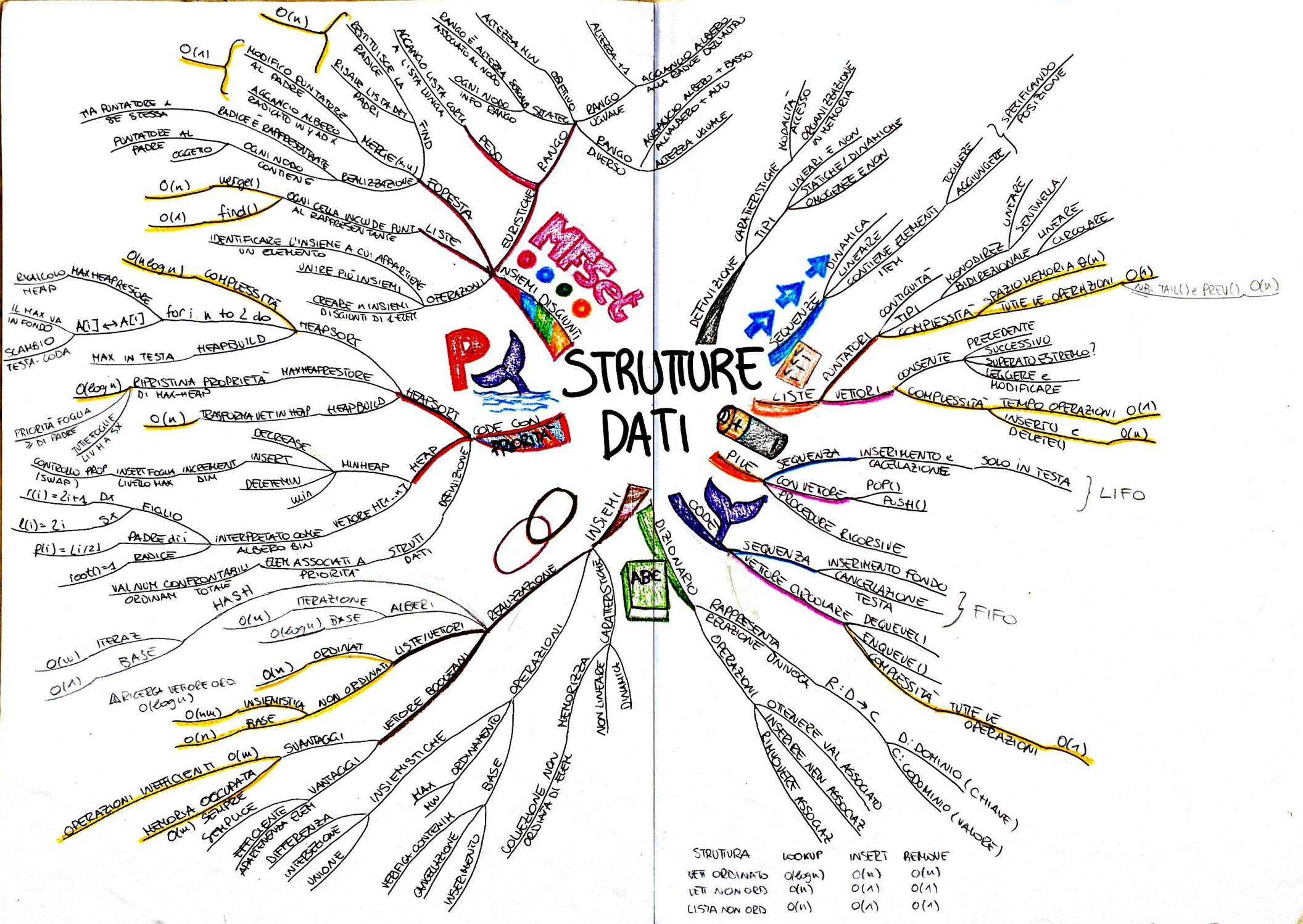
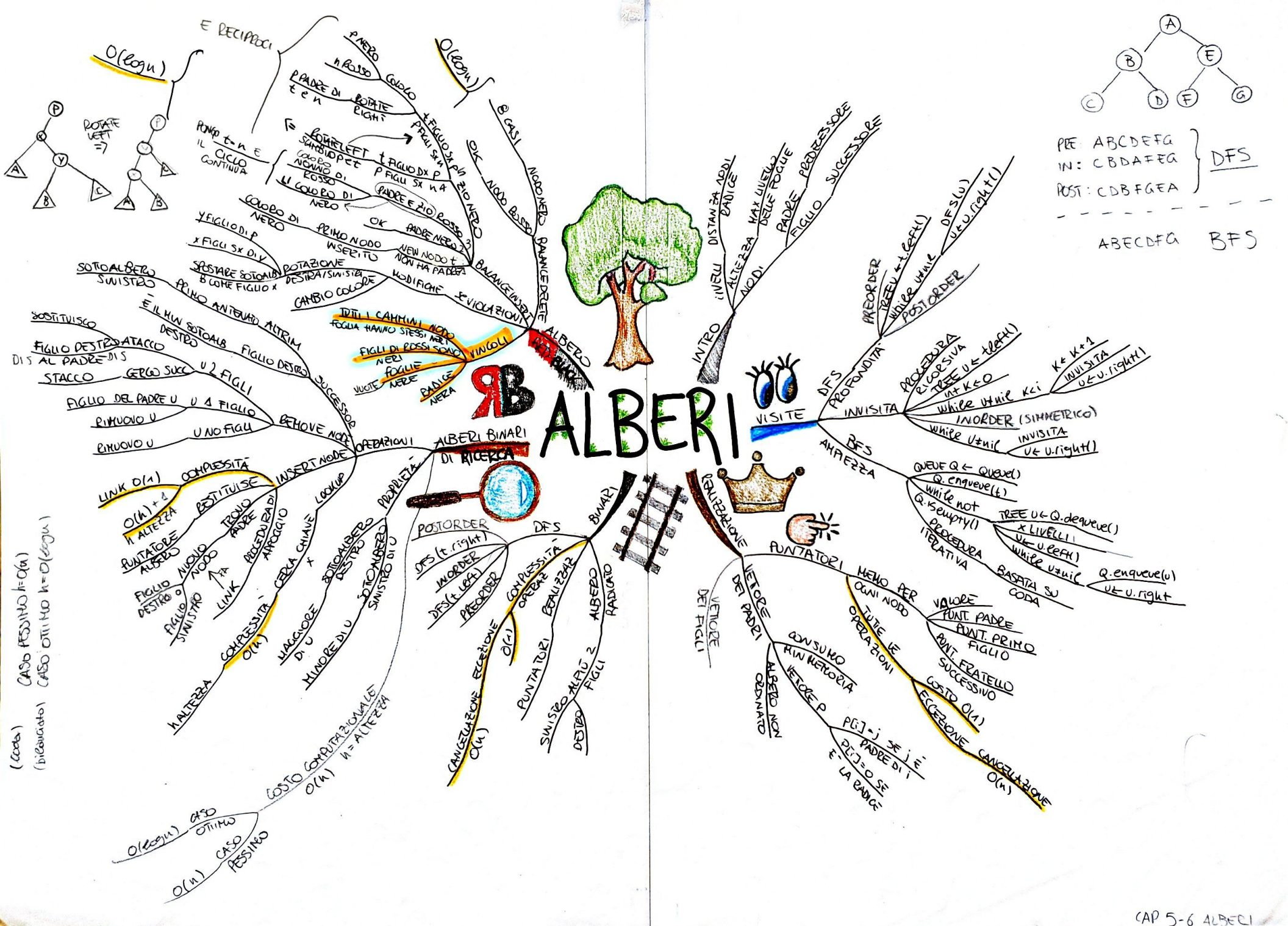
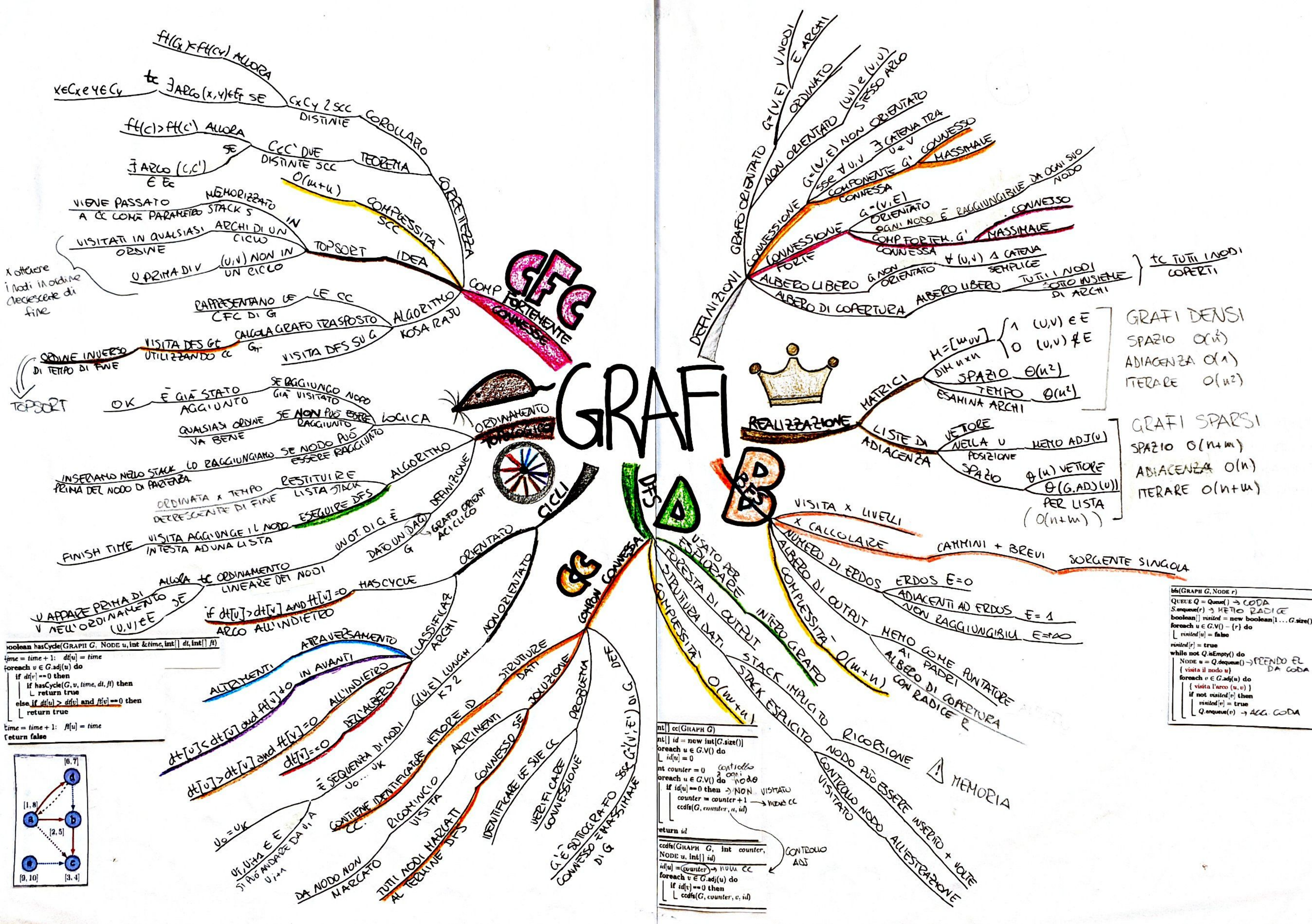
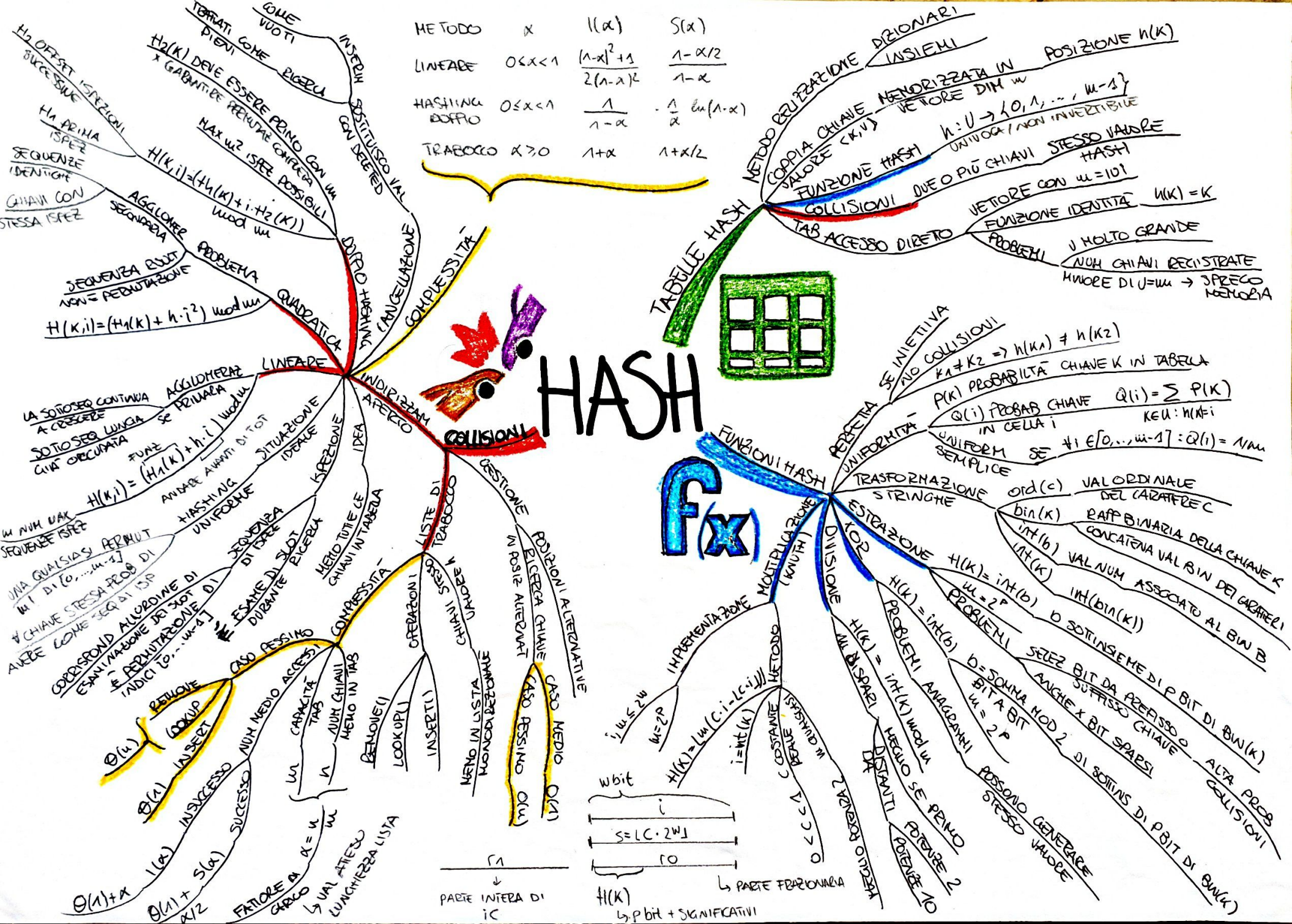

ANALISI ALGORITMI
TEMPO DI CALCOLO
COMPLESSITÀ
REGOLE
METODI
POTENZIARE
CONT BINARIO
AMMORT
ESPANSIONE
SOSTITUZIONE
ESPERTO
BILANCIATA ESTESA
PARTIZIONE BILANCIATA
VETTORI DINAMICI
ANALISI CASI



STRUTTURA	LOOKUP	INSERT	REMOVE
VET ORDINATO	O(log n)	O(n)	O(n)
VET NON ORD	O(n)	O(1)	O(1)
LISTA NON ORD	O(n)	O(1)	O(1)


ALBERI
ALBERI BINARI DI RICERCA
RB
OPERAZIONI
INSERT NODE
LOOKUP
CANCELLAZIONE
PROPRIETÀ
ALTEZZA
COMPLESSITÀ
O(h)
O(n)
O(log n)
CASO PESSIMO h=O(n)
CASO OTTIMO h=O(log n)
(bilanciato)
(lista)
ROTATE LEFT
ROTATE RIGHT
BALANCEINSERT
CASI
NODO ROSSO
NODO NERO
RADICE NERA
FOGLIE NERE
VISITE
DFS
BFS
INVISITA
INORDER (SIMMETRICO)
PREORDER
POSTORDER
ALTEZZA
LIVELLI
PUNTATORI
VETTORE DEI PADRI
VETTORE DEI FIGLI
ALBERO RADICATO
ALBERO NON ORDINATO
INTRO
PADRE
FIGLIO
ANTENATO
DISCENDENTE
PROFONDITÀ
ALTEZZA
NODI
LIVELLI
DISTANZA NODI RADICE
PRE: ABCDEFG
IN: CBDAFEG
POST: CDBFGEA
DFS
ABECDFG
BFS


GRAFI
DEFINIZIONI
GRAFO ORIENTATO G=(V,E) V NODI E ARCHI
ORDINATO (u,v) e (v,u) STESSO ARCO
NON ORIENTATO G=(V,E) NON ORIENTATO
CONNESSIONE sse ∀ u,v ∃ CATENA TRA u e v
COMPONENTE G' CONNESSA MASSIMALE
CONNESSIONE FORTE G=(V,E) ORIENTATO OGNI NODO È RAGGIUNGIBILE DA OGNI SUO NODO CONNESSO
COMP FORT.M. G' CONNESSA MASSIMALE
ALBERO LIBERO G NON ORIENTATO ∀ (u,v) ∃ CATENA SEMPLICE
ALBERO DI COPERTURA ALBERO LIBERO TUTTI I NODI SOTTO INSIEME DI ARCHI
tc TUTTI I NODI COPERTI
REALIZZAZIONE
MATRICI M=[m uv] 1 (u,v) ∈ E, 0 (u,v) ∉ E
DIM n×n
SPAZIO Θ(n²)
TEMPO Θ(n²)
ESAMINA ARCHI
GRAFI DENSI SPAZIO O(n²) ADIACENZA O(1) ITERARE O(n²)
LISTE DI ADIACENZA VETTORE NELLA u POSIZIONE MEMO ADJ(u)
SPAZIO Θ(n) VETTORE Θ(G.ADJ(u)) PER LISTA (O(n+m))
GRAFI SPARSI SPAZIO O(n+m) ADIACENZA O(n) ITERARE O(n+m)
BFS
VISITA × LIVELLI
× CALCOLARE CAMMINI + BREVI SORGENTE SINGOLA
NUMERO DI ERDOS ERDOS E=0
ADIACENTI AD ERDOS E=1
NON RAGGIUNGIBILI E=+∞
ALBERO DI OUTPUT MEMO COME PUNTATORE AI PADRI
ALBERO DI COPERTURA CON RADICE r
COMPLESSITÀ O(m+n)
bfs(GRAPH G, NODE r)
QUEUE Q = Queue() → CODA
S.enqueue(r) → METTO RADICE
boolean[] visited = new boolean[1 ... G.size()]
foreach u ∈ G.V() − {r} do
visited[u] = false
visited[r] = true
while not Q.isEmpty() do
NODE u = Q.dequeue() → PRENDO EL DA CODA
{ visita il nodo u }
foreach v ∈ G.adj(u) do
{ visita l'arco (u, v) }
if not visited[v] then
visited[v] = true
Q.enqueue(v) → AGG. CODA
DFS
USATO PER ESPLORARE INTERO GRAFO
FORESTA DI OUTPUT
STRUTTURA DATI STACK IMPLICITO RICORSIONE ⚠ MEMORIA
STACK ESPLICITO NODO PUÒ ESSERE INSERITO + VOLTE
CONTROLLO NODO VISITATO ALL'ESTRAZIONE
COMPLESSITÀ O(m+n)
CC COMPON CONNESSA
DEF G' È SOTTOGRAFO sse G'(V',E') DI G CONNESSO E MASSIMALE DI G
PROBLEMA VERIFICARE CONNESSIONE
SOLUZIONE IDENTIFICARE LE SUE CC
CONNESSO SE ALTRIMENTI
STRUTTURE DATI VETTORE ID CONTIENE IDENTIFICATORE CC
RICOMINCIO VISITA DA NODO NON MARCATO
TUTTI NODI MARCATI AL TERMINE DFS
int[] cc(GRAPH G)
int[] id = new int[G.size()]
foreach u ∈ G.V() do
id[u] = 0
int counter = 0 controllo ogni nodo
foreach u ∈ G.V() do
if id[u] == 0 then → NON VISITATO
counter = counter + 1 → nuova CC
ccdfs(G, counter, u, id)
return id
CONTROLLO ADJ
ccdfs(GRAPH G, int counter, NODE u, int[] id)
id[u] = counter → NUM CC
foreach v ∈ G.adj(u) do
if id[v] == 0 then
ccdfs(G, counter, v, id)
CICLI
DEFINIZIONE G(V,E) LUNGH K ≥ 2
È SEQUENZA DI NODI u0 ... uK
ui, ui+1 ∈ E SI PUÒ ANDARE DA ui A ui+1
u0 = uK
ORIENTATO GRAFO ORIENT ACICLICO
NON ORIENTATO
CLASSIFICAZ ARCHI
ATTRAVERSAMENTO ALTRIMENTI
IN AVANTI dt[u] < dt[v] and ft[v] ≠ 0
ALL'INDIETRO dt[u] > dt[v] and ft[v] = 0
DELL'ALBERO dt[v] = 0
HASCYCLE if dt[u] > dt[v] AND ft[v] = 0 ARCO ALL'INDIETRO
boolean hasCycle(GRAPH G, NODE u, int &time, int[] dt, int[] ft)
time = time + 1; dt[u] = time
foreach v ∈ G.adj(u) do
if dt[v] == 0 then
if hasCycle(G, v, time, dt, ft) then
return true
else if dt[u] > dt[v] and ft[v] == 0 then
return true
time = time + 1; ft[u] = time
return false
[6, 7] [1, 8] [2, 5] [9, 10] [3, 4]
a b c d e
ORDINAMENTO TOPOLOGICO
DEFINIZIONE DATO UN DAG G UN OT. DI G È UN ORDINAMENTO LINEARE DEI NODI tc SE (u,v) ∈ E ALLORA u APPARE PRIMA DI v NELL'ORDINAMENTO
LOGICA SE RAGGIUNGO NODO GIÀ VISITATO È GIÀ STATO AGGIUNTO OK
SE NON PUÒ ESSERE RAGGIUNTO QUALSIASI ORDINE VA BENE
SE NODO PUÒ ESSERE RAGGIUNTO LO RAGGIUNGIAMO INSERIAMO NELLO STACK PRIMA DEL NODO DI PARTENZA
ALGORITMO RESTITUIRE LISTA STACK ORDINATA × TEMPO DECRESCENTE DI FINE
ESEGUIRE DFS VISITA AGGIUNGE IL NODO IN TESTA AD UNA LISTA FINISH TIME
CFC COMP FORTEMENTE CONNESSE
ALGORITMO KOSARAJU
VISITA DFS SU G
CALCOLA GRAFO TRASPOSTO G_T
VISITA DFS G_T UTILIZZANDO CC ORDINE INVERSO DI TEMPO DI FINE ⇓ TOPSORT
LE CC RAPPRESENTANO LE CFC DI G
IDEA TOPSORT VIENE PASSATO A CC COME PARAMETRO MEMORIZZATO IN STACK S
VISITATI IN QUALSIASI ORDINE ARCHI DI UN CICLO
u PRIMA DI v (u,v) NON IN UN CICLO
x ottenere i nodi in ordine decrescente di fine
COMPLESSITÀ SCC O(m+n)
CORRETTEZZA
TEOREMA C,C' DUE DISTINTE SCC SE ∃ ARCO (C,C') ∈ Ec ft(C) > ft(C') ALLORA
COROLLARIO Cx Cy 2 SCC DISTINTE tc ∃ ARCO (x,y) ∈ Et SE x∈Cx e y∈Cy ft(Cx) < ft(Cy) ALLORA


HASH
COLLISIONI
INDIRIZZAM APERTO
DOPPIO HASHING
LINEARE
QUADRATICA
COMPLESSITÀ
INSERM
SOSTITUISCO VAL CON DELETED
COME VUOTI
TRATTATI COME PIENI
RICERCA
H2(K) DEVE ESSERE PRIMO CON M
GARANTIRE PERMUTARE COMPLETA
MAX M2 ISPEZ POSSIBILI
H1 OFFSET ISPEZIONI SUCCESSIVE
H1 PRIMA ISPEZ
H(K,i)=(H1(K)+i·H2(K)) MOD M
SEQUENZE IDENTICHE
CHIAVI CON STESSA ISPEZ
AGGLOMER SECONDARIA
PROBLEMA
SEQUENZA PROVE NON = PERMUTAZIONE
H(K,i)=(H1(K)+h·i²) MOD M
AGGLOMERAZ PRIMARIA
SE LA SOTTOSEQ CONTINUA A CRESCERE
SOTTOSEQ LUNGA CHE OCCUPATA
FUNZ
H(K,i)=(H1(K)+h·i) MOD M
ANDARE AVANTI DI TOT
SITUAZIONE IDEALE
IDEA
ESAME DI SLOT DURANTE RICERCA
ESAME DI SLOT DURANTE INSERZIONE
HASHING UNIFORME
SEQUENZA DI ISPEZ
COINCIDONO ALL'ORDINE DI ESAMINAZIONE DEI SLOT
E' PERMUTAZIONE DI [0,...,M-1]
INDICI [0,...,M-1]
M NON VUOI SEQUENZE ISPEZ
UNA QUALSIASI PERMUT DI [0,...,M-1]
CHIAVE STESSA PROVE DI ISP
AVERE COME SEQ QUELLA
LISTE DI TRABOCCO
OPERAZIONI
CHIAVI STESSO VALORE
MENO IN LISTA MONDA DEZIONE
COMPLESSITÀ
CASO PESSIMO
Θ(n)
RICERCA
LOOKUP
INSERT
Θ(1)
INSUCCESSO
SUCCESSO
NUM MEDIO ACCESSI
CAPACITÀ TAB
M
N
NUM CHIAVI MENO IN TAB
BENCHMARK
LOOKUP()
INSERT()
QUESTIONE
POSIZIONI ALTERNATIVE
RICERCA CHIAVE IN POSIZ ALTERNAT
CASO PESSIMO O(n)
CASO MEDIO O(1)
Θ(1)+α
Θ(1)+α/2
1(α)
S(α)
FATTORE DI CARICO α = n/m
↳ VAL ATTESO LUNGHEZZA LISTA
METODO | α | 1(α) | S(α)
LINEARE | 0≤α<1 | (1-α)²+1 / 2(1-α)² | 1-α/2 / 1-α
HASHING DOPPIO | 0≤α<1 | 1/(1-α) | -1/α ln(1-α)
TRABOCCO | α≥0 | 1+α | 1+α/2
r1
↓
PARTE INTERA DI iC
w bit
i
s=⌊C·2^w⌋
r0
H(K)
↳ p bit + SIGNIFICATIVI
↳ PARTE FRAZIONARIA
TABELLE HASH
METODO REALIZZAZIONE
DIZIONARI
INSIEMI
COPPIA CHIAVE VALORE <K,V>
MEMORIZZATA IN VETTORE DIM M
POSIZIONE h(K)
FUNZIONE HASH
h: U → {0,1,...,M-1}
UNIVOCA / NON INVERTIBILE
COLLISIONI
DUE O PIÙ CHIAVI STESSO VALORE HASH
TAB ACCESSO DIRETTO
VETTORE CON M=10^9
FUNZIONE IDENTITÀ
h(K) = K
PROBLEMI
U MOLTO GRANDE
NUM CHIAVI REGISTRATE MINORE DI U=M → SPRECO MEMORIA
FUNZIONI HASH
PERFETTA
SE INIETTIVA
NO COLLISIONI
K1≠K2 => h(K1) ≠ h(K2)
UNIFORMITÀ
P(K) PROBABILITÀ CHIAVE K IN TABELLA
Q(i) PROBAB CHIAVE IN CELLA i
Q(i) = Σ P(K) K∈U : h(K)=i
UNIFORM SEMPLICE
SE ∀i ∈[0,...,m-1] : Q(i) = 1/M
TRASFORMAZIONE STRINGHE
ord(c) VAL ORDINALE DEL CARATTERE C
bin(k) RAPP BINARIA DELLA CHIAVE K
CONCATENA VAL BIN DEI CARATTERI
int(b) VAL NUM ASSOCIATO AL BIN B
int(k) int(bin(k))
ESTRAZIONE
KOR
h(K) = int(b) MOD m
m = 2^p
PROBLEMI
b SOTTINSIEME DI p BIT DI bin(k)
SCELGO BIT DA PREFISSO O SUFFISSO CHIAVE
ANCHE X BIT SPARSI
ALTA PROB COLLISIONI
b = SOMMA MOD 2 DI SOTTINS DI p BIT DI bin(k)
BIT A BIT
m = 2^p
POSSONO GENERARE STESSO VALORE
DIVISIONE
h(K) = int(K) MOD m
PROBLEMI ANAGRAMMI
MEGLIO SE PRIMO DISTANTI DA POTENZE DI 2
POTENZE 2 E 10
MOLTIPLICAZIONE (KNUTH)
h(K) = ⌊m(C·i - ⌊C·i⌋)⌋
i = int(k)
C COSTANTE REALE
0 < C < 1
m QUASI SPARE
IMPLEMENTAZIONE
i ≤ w
m = 2^p
h(K) = ⌊m(C·i - ⌊C·i⌋)⌋
i = int(k)
w BIT
s = ⌊C·2^w⌋
r1
r0
h(K)
↳ p bit + SIGNIFICATIVI
↳ PARTE FRAZIONARIA

CAMMINI MINIMI

Input	Complessità	Approccio
Pesi positivi, grafo denso	$O(n \cdot n^2)$	Applicazione ripetuta dell'algoritmo di Dijkstra
Pesi positivi, grafo sparso	$O(n \cdot (m \log n))$	Applicazione ripetuta dell'algoritmo di Johnson
Pesi negativi	$O(n \cdot nm)$	Applicazione ripetuta di Bellman-Ford, scousigliata
Pesi negativi, grafo denso	$O(n^3)$	Algoritmo di Floyd e Warshall
Pesi negativi, grafo sparso	$O(nm \log n)$	Algoritmo di Johnson per sorgente multipla

MULTI

- FLOYD WARSHALL
  - $d^k[x][y] = \min(d^{k-1}[x][y], d^{k-1}[x][k] + d^{k-1}[k][y])$ RICORSIVA
  - $k = 0$ $w(x,y)$
  - $k > 0$
  - ESEMPI
    - PESO ARCO $d^0[x][y]$
    - ∞
  - $d^k[x][y]$
  - DISTANZA K-VINCOLATA
  - $\exists p^k_{xy}$ $w(p^k_{xy})$
  - COMPLESSITÀ $O(n^3)$
- CHIUSURA TRANSITIVA
  - DATO GRAFO $G(V,E)$
  - GRAFO $G^*(V,E^*)$ ORIENTATO
  - $(u,v) \in E^*$ SE CONTIENE ALMENO UN CAMMINO $u \rightsquigarrow v$ IN G
- SORGENTE MULTIPLA
- CAMMINI MIN K-VINCOLATI
  - ESEMPI
    - NESSUN VINCOLO p^n_{xy}
    - ARCO p^1_{xy}
    - NON PASSA PER $v_{k+1} \dots v_n$ p^k_{xy}
    - CAMMINO COSTO MIN TRA x e y

± BF BELLMAN FORD

- ALGORITMO
  - $O(n)$ Q.enqueue(s) INSERIMENTO
  - $O(1)$ U = Q.dequeue() ESTRAZIONE
  - $O(1)$ QUEUE Q = Queue(); Q.add(s) INIZIALIZZAZ
  - $O(n)$ CICLO CODA DIN n CODA
  - NODO PUÒ ESSERE ESTRATTO + VOLTE
  - ANCHE CON PESI NEGATIVI
- $O(nm)$ COMPLESSITÀ
- $O(nm)$

+ DIJKSTRA

- CODA PRIORITÀ BASATA VETTORE 1959
- ALGORITMO
  - INIZIALIZZAZIONE PRIORITYQ S ← PriorQ(); S.insert(s,0) $O(n)$
  - ESTRAZIONE MIN U = S.deleteMin() $O(n)$
  - INSERIMENTO CODA S.insert(v, d[u] + w(u,v)) $O(1)$
  - AGGIORNO PRIORITÀ S.decrease(u, d[u] + w(u,v)) $O(1)$
- CORRETTEZZA ∀ NODO ESTRATTO 1 VOLTA ALL'ESTRAZIONE DISTANZA MINIMA
- COMPLESSITÀ $O(n^2)$

+ JOHNSON

- COMPLESSITÀ $O(n \log n + m)$
- NODO ESTRATTO 1 VOLTA
- CODA CON PRIORITÀ
- UTILIZZA UN HEAP BINARIO

+ FREDMAN TARJAN

- COMPLESSITÀ
- HEAP DI FIBONACCI
- $O(n \log n + m)$

TH TEOREMA BELLMAN

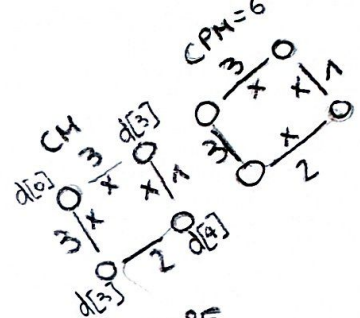
- DIM OTTIMA ⇒ CONDIZIONI
  - SOL AMMISSIBILE SSR T È OTTIMA
  - $d[v] \leq d[u] + w(u,v)$ ∀ ARCO $(u,v) \in E$
  - $d[v] = d[u] + w(u,v)$ ∀ ARCO $(u,v) \in T$
  - ALB DI COPERTURA
  - SOMMA ARCHI MINORE
  - TUTTI VERTICI
  - ALCUNI ARCHI
  - SE $(u,v) \in T$ ⇒ $d[v] = d[u] + w(u,v)$
  - SE $(u,v) \notin T$ ⇒ $d[v] \leq d[u] + w(u,v)$
  - ALTRIMENTI USING CAMMINO + CORTO
  - ASSURDO
- CONDIZIONI ⇐ OTTIMA
  - T → se non ottimo
  - $T' < T$ ⇒ ASSURDO
  - ASSURDO

DEF PB

- DATO CAMMINO $p = \langle v_1 \dots v_k \rangle$
- INPUT
- OUTPUT
- COSTO CAMMINO $= w(p) = \sum_{i=2}^{k} w(v_{i-1}, v_i)$
- GRAFO ORIENTATO
- NODO SORGENTE s
- FUNZIONE PESO $w: E \to \mathbb{R}$
- CAMMINO DA s A v COSTO MINIMO $\forall v \in V$

ALBERI

- CAMMINI MINIMI
- COPERTURA PESO MINIMO
- COPERTURA CONTIENE



Dijkstra	$O(n^2)$	Pesi positivi, grafi densi
Johnson	$O(m \log n)$	Pesi positivi, grafi sparsi
Fredman-Tarjan	$O(m + n \log n)$	Pesi positivi, grafi densi, dimensioni molto grandi
Bellman-Ford	$O(mn)$	Pesi negativi
	$O(m+n)$	DAG
BFS	$O(m+n)$	Senza pesi

DIVIDE ET IMPERA

STRASSEN

$T(n) = \begin{cases} 7T(n/2) + n^2 & n>1 \\ 1 & n=1 \end{cases}$

$\Theta(n^{2.81})$ COMPLESSITÀ

RICORRENZA

$\Theta(n^3)$ COMPLESSITÀ

$T(n) = \begin{cases} 8T(n/2) + n^2 & n>1 \\ 1 & n=1 \end{cases}$ RICORRENZA

DIVIDE ET IMPERA

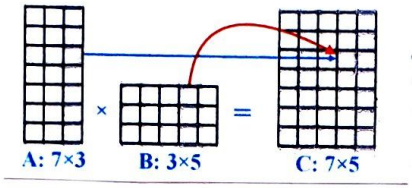
in 4 MATRICI $n/2 \times n/2$ — SUDDIVIDO MATRICI $n \times n$

MOLTIPLICAZIONE MATRICI

$\Theta(n^3)$ QUADRATI — $\Theta(n_i \cdot n_k \cdot n_j)$ — COMPLESSITÀ

FORMULA $c_{ij} = \sum_{k=1}^{n_k} a_{i,k} \cdot b_{k,j}$

SOMMATORIA RIGA COLONNA



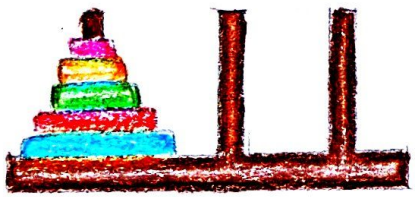
IDEA

PB SUDDIVISO IN SOTTOPB — TOP DOWN

RISOLTI RICORSIVAMENTE

3 FASI — DIVIDE — RISOLVE SOTTOPB RICORSIVAMENTE

IMPERA — COMBINA — UNISCE SOLUZ SOTTOPB



TORRI DI HANOI

GIOCO — n dischi — 3 PIOLI

SCOPO — IMPILARE DISCHI A DESTRA ORDINE DECRESCENTE

DIVIDE ET IMPERA

n-1 DISCHI DA SRC A MIDDLE

1 DISCO DA SRC A DEST

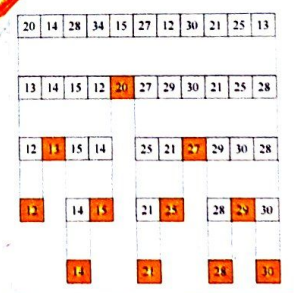
n-1 DISCHI DA MIDDLE A DEST

RICORRENZA $T(n) = 2T(n-1) + 1$

COMPLESSITÀ $O(2^n)$

QUICKSORT

HA FATTORI NON NIGLIORI DI MERGESORT



DIVISO SEMPRE IN $n/2$ — $\Theta(n \log n)$ OTTIMO

VETT DIVISO IN 0 e $n-1$ — $\Theta(n^2)$ PESSIMO

COSTO QUICK — COSTO PIVOT $\Theta(n)$

COMPLESSITÀ

3 FASI

NON FA NULLA — COMBINA

RICHIAMO RICORSIVO QUICKSORT — ORDINA I SOTTOVET — IMPERA

DIVIDE

MAGGIORI DI P — Dx

SPOSTA GLI ELEMENTI

MINORI DI P — Sx

SCEGLIE VALORE P PIVOT (PERNO)

PER EVITARLO — TECNICHE EURISTICHE — PESSIMO $O(n^2)$

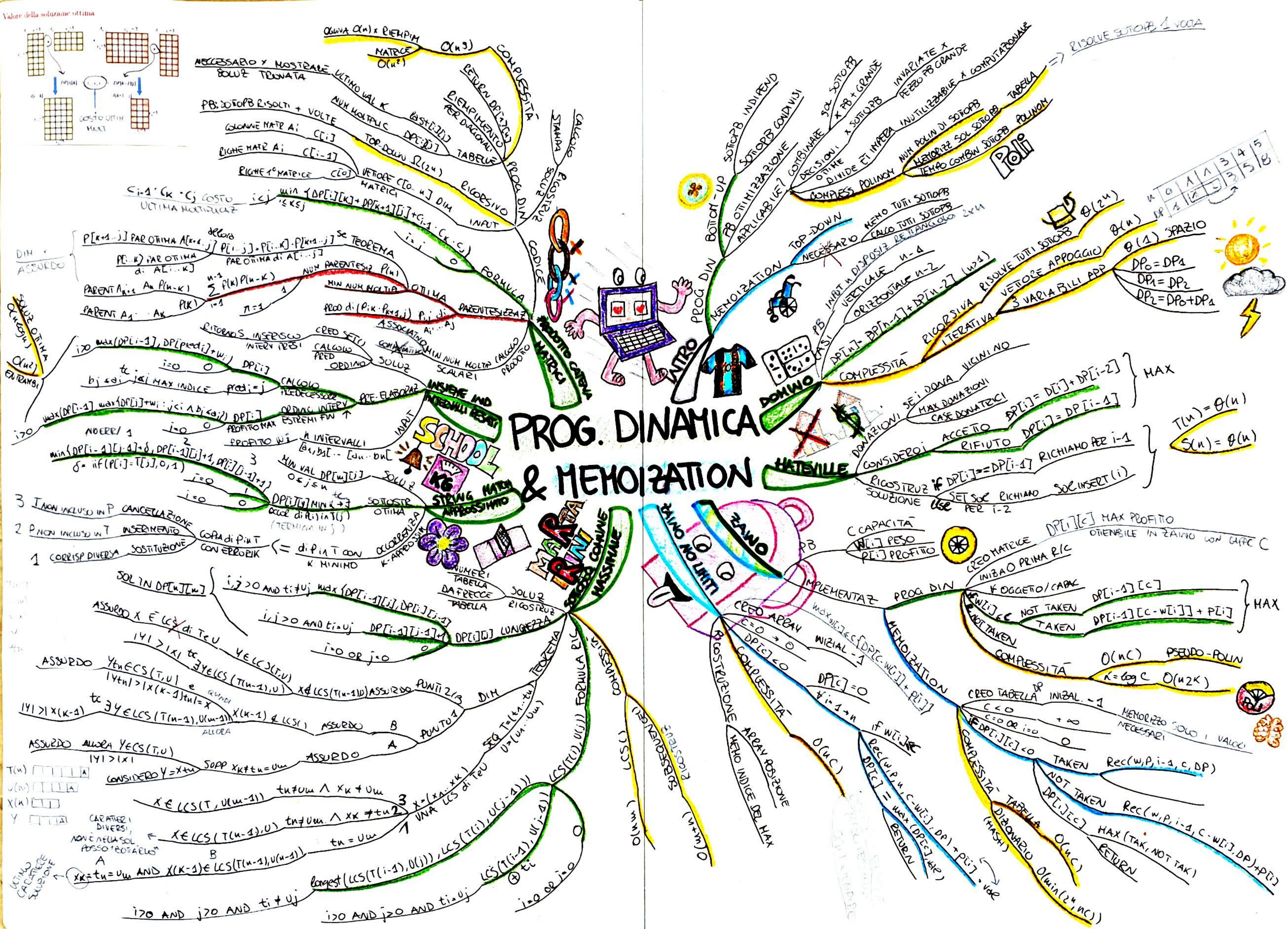
MEDIO $O(n \log n)$

COMPLESSITÀ

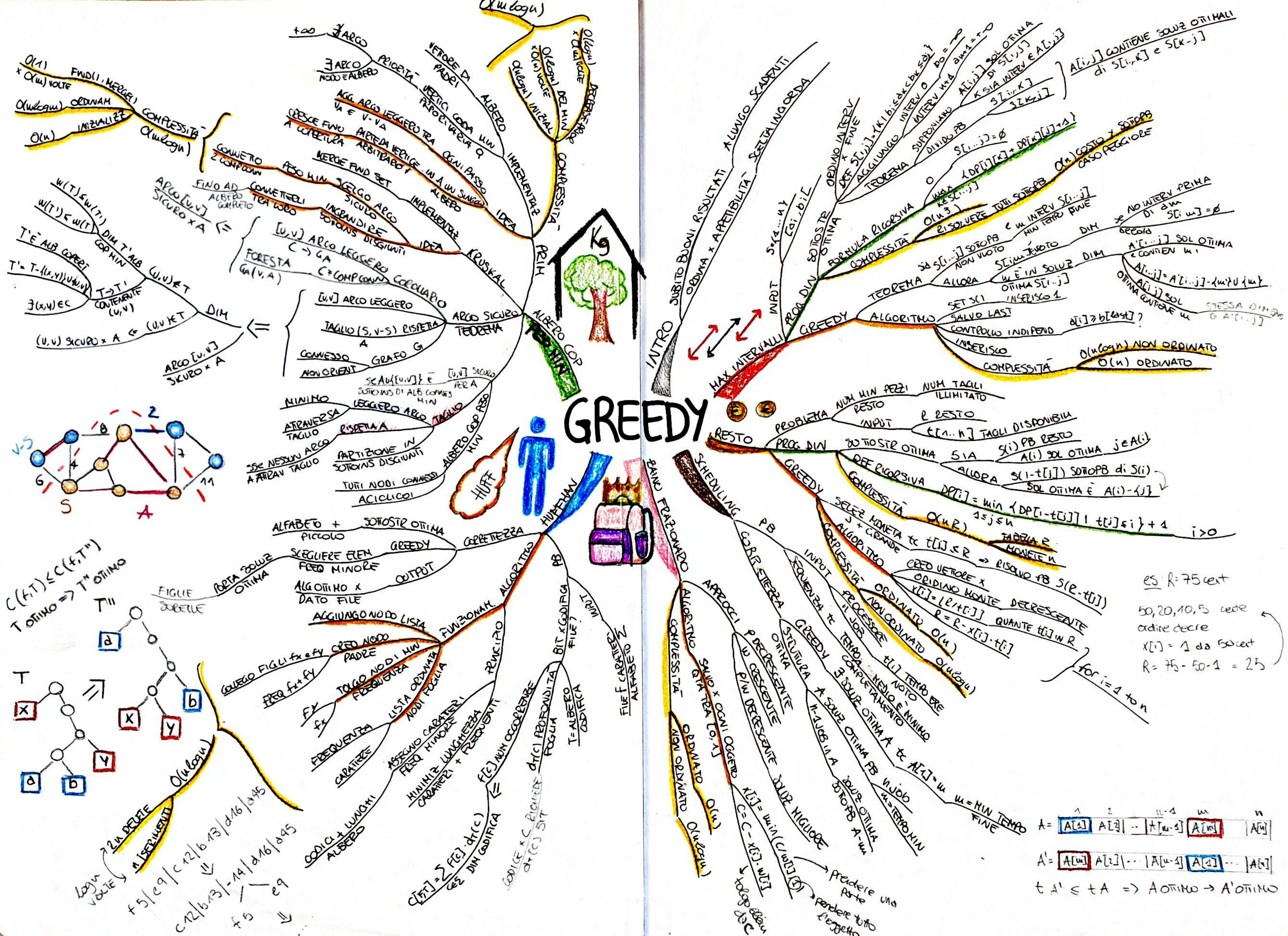
INPUT

VETTORE A[1..n]

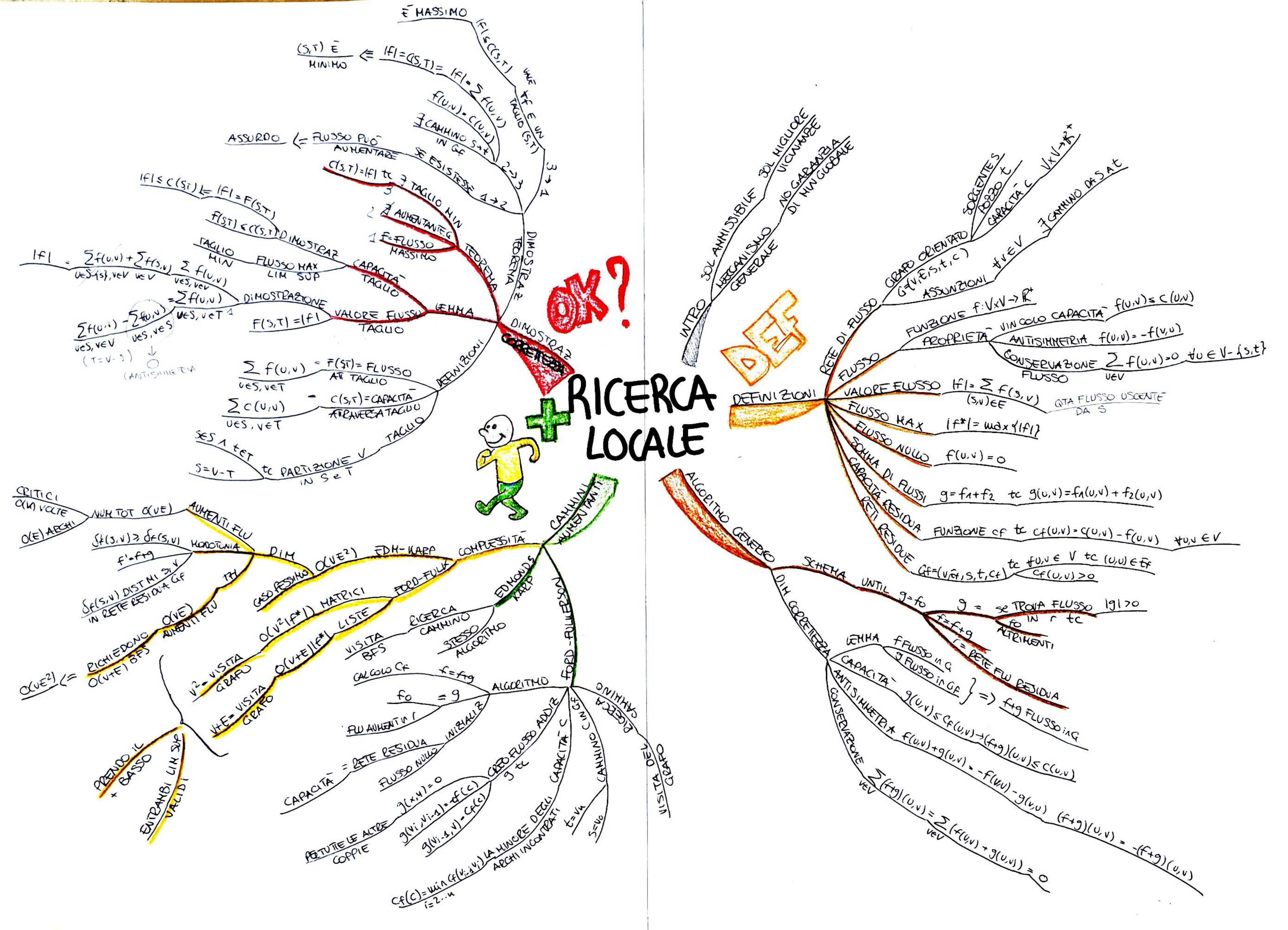
$1 \le i, k \le n$



Valore della soluzione ottima
PROG. DINAMICA & MEMOIZATION
INTRO
PROG DIN
BOTTOM-UP
SOTTOPB INDIPEND
SOTTOPB CONDIVISI
PB OTTIMIZZAZIONE
APPLICABILE
COMBINARE SOL SOTTOPB
DECISIONI x SOTTOPB
x PB + GRANDE
INVARIATE x PEZZO PB GRANDE
DIVIDE ET IMPERA
INUTILIZZABILE x COMPUTAZIONALE
NUM POLIN DI SOTTOPB
MEMORIZZ SOL SOTTOPB
TABELLA
=> RISOLVE SOTTOPB 1 VOLTA
COMPLESS POLINOM
TEMPO COMBIN SOTTOPB POLINOM
MEMOIZATION
TOP-DOWN
NECESSARIO
MEMO TUTTI SOTTOPB
CALCO TUTTI SOTTOPB
DOMINO
PB
INPUT n DISPOSIZ RETTANGOLO 2xn
CASI
VERTICALE n-1
ORIZZONTALE n-2
DP[n] = DP[n-1] + DP[n-2] (n>1)
RICORSIVA
RISOLVE TUTTI SOTTOPB
Θ(2^n)
ITERATIVA
VETTORE APPOGGIO
Θ(n)
3 VARIABILI APP
Θ(1) SPAZIO
DP0 = DP1
DP1 = DP2
DP2 = DP0 + DP1
COMPLESSITÀ
HATEVILLE
DONAZIONI
SE i DONA VICINI NO
MAX DONAZIONI
CASE DONATRICI
CONSIDERO i
ACCETTO
DP[i] = D[i] + DP[i-2]
RIFIUTO
DP[i] = DP[i-1]
RICHIAMO PER i-1
MAX
T(n) = Θ(n)
S(n) = Θ(n)
RICOSTRUZ SOLUZIONE
if DP[i] == DP[i-1]
else
SET SOL RICHIAMO PER i-2
SOL.INSERT(i)
ZAINO
PB
CAPACITÀ
W[i] PESO
P[i] PROFITTO
DP[i][c] MAX PROFITTO OTTENIBILE IN ZAINO CON CAPAC C
IMPLEMENTAZ
PROG DIN
CREO MATRICE
INIZ A 0 PRIMA R/C
∀ OGGETTO/CAPAC
NOT TAKEN
DP[i-1][c]
TAKEN
DP[i-1][c-w[i]] + P[i]
MAX
COMPLESSITÀ
O(nC)
PSEUDO-POLIN
K = log C
O(n2^K)
MEMOIZATION
CREO TABELLA DP
INIZIAL -1
MEMORIZZO SOLO I VALORI NECESSARI
C<0
-∞
C=0 OR i=0
0
if DP[i][c] < 0
TAKEN
Rec(w, P, i-1, c, DP)
NOT TAKEN
Rec(w, P, i-1, c-w[i], DP) + P[i]
DP[i][c]
MAX(TAK, NOT TAK)
RETURN
COMPLESSITÀ TABELLA
DIZIONARIO (HASH)
O(nC)
O(min(2^n, nC))
ZAINO NO LIMITI
CREO ARRAY
C=0 → 0
DP[c] < 0
INIZIAL -1
max w[i]≤c {DP[c-w[i]] + P[i]}
DP[c] = 0
∀ i=1→n
if w[i] ≤ c
Rec(w, p, n, c-w[i], DP) + p[i]
DP[c] = max(DP[c], ...)
RETURN
COMPLESSITÀ
O(nC)
RICOSTRUZIONE
ARRAY POSIZIONE
MEMO INDICE DEL MAX
SOTTOSEQ COMUNE MASSIMALE
COMPLESSITÀ
LCS()
O(nm)
SUBSEQUENCE()
RICOSTRUZ
O(m+n)
FORMULA RIC
LCS(T(i), U(j))
0
i=0 OR j=0
LCS(T(i-1), U(j-1)) ⊕ ti
i>0 AND j>0 AND ti = uj
longest(LCS(T(i-1), U(j)), LCS(T(i), U(j-1)))
i>0 AND j>0 AND ti ≠ uj
TEOREMA
SEQ T = (t1 ... tn) U = (u1 ... um)
X = (x1 ... xk) UNA LCS di T e U
1 tn = um
xk = tn = um AND X(k-1) ∈ LCS(T(n-1), U(m-1))
2 tn ≠ um ∧ xk ≠ tn
X ∈ LCS(T(n-1), U)
3 tn ≠ um ∧ xk ≠ um
X ∈ LCS(T, U(m-1))
ULTIMO CARATTERE SOLUZIONE
CARATTERI DIVERSI, NON È NELLA SOL POSSO "BUTTARLO"
DIM
PUNTO 1
ASSURDO
CONSIDERO Y = X ⊕ tn
SOPP xk ≠ tn = um
ASSURDO ALLORA Y ∈ CS(T, U)
|Y| > |X|
X(k-1) ∉ LCS()
|Y| > X(k-1)
tc ∃ Y ∈ LCS(T(n-1), U(m-1))
PUNTI 2/3
X ∉ LCS(T(n-1), U) ASSURDO
ASSURDO ∀ tn ∈ CS(T, U)
|Y| > |X|
tc ∃ Y ∈ LCS(T(n-1), U)
Y ∈ LCS(T, U)
ASSURDO X È LCS di T e U
SOL IN DP[n][m]
LUNGHEZZA
DP[i][j]
0
i=0 OR j=0
DP[i-1][j-1] + 1
i, j > 0 AND ti = uj
max(DP[i-1][j], DP[i][j-1])
i, j > 0 AND ti ≠ uj
SOLUZ RICOSTRUZ
NUMERI TABELLA
DA FRECCE TABELLA
STRING MATCH APPROSSIMATO
OCCORRENZA K-APPROSSIM
di P in T CON K MINIMO
1 CORRISP DIVERSA SOSTITUZIONE
2 P NON INCLUSO IN T INSERIMENTO
3 T NON INCLUSO IN P CANCELLAZIONE
COPIA di P in T CON ERRORI K
SOTTOSTR OTTIMA
DP[i][j] MIN K
tc
occor di P(i) in T(j)
(TERMINA IN j)
i=0 0
j=0 i
min(DP[i-1][j-1] + δ, DP[i-1][j] + 1, DP[i][j-1] + 1)
δ = iif(P[i] = T[j], 0, 1)
SOLUZ
MIN VAL DP[m][j]
0 ≤ j ≤ n
INSIEME IND INTERVALLI PESATI
INPUT
K INTERVALLI
[a1, b1[... [an ... bn[
PROFITTO wi
PRE-ELABORAZ
ORDINO INTERV ESTREMI FIN ↑
CALCOLO PREDECESSORE
pred i = j
bj ≤ ai
j<i MAX INDICE
DP[i]
i=0 0
i>0 max(DP[i-1], DP[pred i] + wi)
PROFITTO MAX
max(DP[i-1], max{DP[j] + wi : j<i ∧ bj ≤ ai})
SOLUZ
CREO SET()
CALCOLO PRED
ORDINO
RITORNO S, INSERISCO INTERV PRESI
COMPLESSITÀ
SOLUZ OTTIMA
O(n log n)
O(n²)
ENTRAMBI
PRODOTTO CATENA MATRICI
PRODOTTO
MIN NUM MOLTIP SCALARI
CALCOLO PRODOTTO
ASSOCIATIVO
PARENTESIZZAZ
prod di (Pi·...·Pj) Pi,j di Ai·...·Aj
OTTIMA
MIN NUM MOLTIP
NUM PARENTESIZ P(n)
1 n=1
Σ P(k)P(n-k) n>1
PARENT A1 ... Ak P(k)
PARENT Ak+1 ... An P(n-k)
FORMULA
0 i=j
min{DP[i][k] + DP[k+1][j] + ci-1 · ck · cj} i<j
ci-1 · ck · cj COSTO ULTIMA MOLTIPLICAZ
TEOREMA
se P[i..j] = P[i..k] · P[k+1..j] allora P[i..k] PAR OTTIMA di A[i..k]
P[k+1..j] PAR OTTIMA A[k+1..j]
DIM x ASSURDO
CODICE
INPUT
VETTORE C[0..n] DIM MATRICI
C[0] RIGHE 1° MATRICE
C[i-1] RIGHE MATR Ai
C[i] COLONNE MATR Ai
RICORSIVO
TOP-DOWN Ω(2^n)
PB: SOTTOPB RISOLTI + VOLTE
TABELLE
DP[i][j] NUM MOLTIPLIC
last[i][j] ULTIMO VAL K
NECESSARIO x MOSTRARE SOLUZ TROVATA
PROG DIN
RIEMPIMENTO PER DIAGONALI
RETURN DP[1][n]
COMPLESSITÀ
O(n³)
MATRICE O(n²)
OGNUNA O(n) x RIEMPIM
RICOSTRUZ SOLUZ
STAMPA
CALCOLO



GREEDY
INTRO
SUBITO BUONI RISULTATI
A LUNGO SCADENTI
ORDINA x APPETIBILITÀ
SCELTA INGORDA
MAX INTERVALLI
INPUT
PROG DIN
SOTTOSTR OTTIMA
FORMULA RICORSIVA
COMPLESSITÀ
TEOREMA
ALGORITMO
GREEDY
O(nlogn) NON ORDINATO
O(n) ORDINATO
RESTO
PROBLEMA
NUM MIN PEZZI
NUM TAGLI ILLIMITATO
R RESTO
t[1...n] TAGLI DISPONIBILI
S(i) PB RESTO
A(i) SOL OTTIMA
DEF RICORSIVA
DP[i] = min { DP[i-t[j]] | t[j] ≤ i } + 1
i > 0
SELEZ MONETA tc t[i] ≤ R ⇒ RISOLVO PB S(R-t[i])
CREO VETTORE x
ORDINO MONETE DECRESCENTE
x[i] = ⌊R/t[i]⌋ QUANTE t[i] in R
R = R - x[i]·t[i]
for i=1 to n
es: R = 75 cent
50, 20, 10, 5 cent
ordine decre
x[1] = 1 da 50 cent
R = 75 - 50·1 = 25
SCHEDULING
PB
INPUT
PROCESSORE
n JOB
t[i] TEMPO ESE
CORRETTEZZA
GREEDY
STRUTTURA
A SOLUZ OTTIMA PB n JOB
A' = A[m] A[2] ... A[m-1] A[1] ... A[n]
tA' ≤ tA ⇒ A OTTIMO → A' OTTIMO
ZAINO FRAZIONARIO
APPROCCI
ALGORITMO
COMPLESSITÀ
ORDINATO O(n)
NON ORDINATO O(nlogn)
p/w DECRESCENTE
SOLUZ MIGLIORE
prendere una parte
prendere tutto l'oggetto
HUFFMAN
HUFF
CORRETTEZZA
SOTTOSTR OTTIMA
ALFABETO + PICCOLO
GREEDY
SCEGLIERE ELEM FREQ MINORE
OUTPUT
ALG OTTIMO x DATO FILE
FIGLIE SORELLE
PORTA SOLUZ OTTIMA
C(f,T) ≤ C(f,T'')
T OTTIMO ⇒ T'' OTTIMO
ALGORITMO
FUNZIONAM
AGGIUNGO NODO LISTA
CREO NODO PADRE
COLLEGO FIGLI fx e fy
FREQ fx + fy
TOLGO NODI MIN FREQUENZA
PRINCIPIO
LISTA ORDINATA NODI FOGLIA
FREQUENZA
CARATTERE
ASSEGNO CARATTERI FREQ MINORE
MINIMIZ LUNGHEZZA
CODICE + LUNGHI
ALBERO
C[f,T] = Σ f[c]·d_T(c)
PB
BIT x CODIFICA FILE?
T = ALBERO CODIFICA
FILE f CARATTERI ALFABETO
INPUT
O(nlogn)
2n DELETE
n INSERIMENTI
logn VOLTE
f5 | e9 | c12 | b13 | d16 | a45
c12 | b13 | 14 | d16 | a45
ALBERO COP MIN
PRIM
KRUSKAL
IDEA
IMPLEMENTAZ
COMPLESSITÀ
ARCO SICURO TEOREMA
COROLLARIO
[u,v] ARCO LEGGERO
TAGLIO (S, V-S) RISPETTA A
CONNESSO GRAFO G
NON ORIENT
FORESTA GA(V,A)
DIM
(u,v) SICURO x A
ARCO [u,v] SICURO x A
MINIMO
ATTRAVERSA TAGLIO
LEGGERO ARCO
RISPETTA A
PARTIZIONE IN SOTTOINS DISGIUNTI
TUTTI NODI CONNESSI
ACICLICO
FIND(i), MERGE()
O(1)
O(n) VOLTE
O(nlogn) ORDINAM
O(n) INIZIALIZZ
O(nlogn)
CONNETTO 2 COMPONENTI
PESO MIN
SCELGO ARCO SICURO
INGRANDIRE SOTTOINS DISGIUNTI
MERGE FIND SET
CRESCE FINO A COPERTURA
PARTE DA VERTICE ARBITRARIO
AGG ARCO LEGGERO TRA
CODA MIN PRIORITÀ Q
VETTORE DI PADRI
ALBERO
O(nlogn)
+∞
S
A



RICERCA LOCALE
OK?
DEF
INTRO
SOL AMMISSIBILE
SOL MIGLIORE
MECCANISMO GENERALE
VICINANZE
NO GARANZIA DI MIN GLOBALE
DEFINIZIONI
RETE DI FLUSSO
GRAFO ORIENTATO G=(V,E,s,t,c)
SORGENTE s
POZZO t
CAPACITÀ c V×V→ℝ+
ASSUNZIONI ∀v∈V ∃ CAMMINO DA s A t
FLUSSO
FUNZIONE f: V×V→ℝ+
PROPRIETÀ
VINCOLO CAPACITÀ f(u,v) ≤ c(u,v)
ANTISIMMETRIA f(u,v) = −f(v,u)
CONSERVAZIONE FLUSSO Σ v∈V f(u,v) = 0 ∀u ∈ V − {s,t}
VALORE FLUSSO |f| = Σ (s,v)∈E f(s,v)
QTA FLUSSO USCENTE DA S
FLUSSO MAX |f*| = max{|f|}
FLUSSO NULLO f(u,v) = 0
SOMMA DI FLUSSI g = f1 + f2 tc g(u,v) = f1(u,v) + f2(u,v)
CAPACITÀ RESIDUA FUNZIONE cf tc cf(u,v) = c(u,v) − f(u,v) ∀u,v ∈ V
RETI RESIDUE Gf = (V, Ef, s, t, cf) tc ∀u,v ∈ V tc (u,v) ∈ Ef cf(u,v) > 0
ALGORITMO GENERICO
SCHEMA
UNTIL g = f0
g = se TROVA FLUSSO |g| > 0 IN r tc
f0 ALTRIMENTI
f = f+g
r = RETE FLU RESIDUA
DIM CORRETTEZZA
LEMMA f FLUSSO IN G, g FLUSSO IN Gf } ⇒ f+g FLUSSO IN G
CAPACITÀ g(u,v) ≤ cf(u,v) → (f+g)(u,v) ≤ c(u,v)
ANTISIMMETRIA f(u,v) + g(u,v) = −f(v,u) − g(v,u) (f+g)(u,v) = −(f+g)(u,v)
CONSERVAZIONE Σ v∈V (f+g)(u,v) = Σ v∈V (f(u,v) + g(u,v)) = 0
CAMMINI AUMENTANTI
FORD-FULKERSON
ALGORITMO
f0 = g
CALCOLO Cf
f = f+g
INIZIALIZ
FLU AUMENT IN r
RETE RESIDUA
CAPACITÀ = FLUSSO NULLO
CREO FLUSSO ADDIZ g tc
g(x,y) = 0
g(vi, vi-1) = −cf(c)
g(vi-1, vi) = cf(c)
PER TUTTE LE ALTRE COPPIE
CAPACITÀ c
LA MINORE DEGLI ARCHI INCONTRATI
cf(c) = min cf(vi-1, vi) i=2..n
CAMMINO C IN Gf
t = vn
s = v0
RICERCA CAMMINO
VISITA DEL GRAFO
EDMONDS KARP
RICERCA CAMMINO
VISITA BFS
STESSO ALGORITMO
COMPLESSITÀ
FORD-FULK
O(V²|f*|) MATRICI
LISTE
O(V+E)|f*|
V² = VISITA GRAFO
V+E = VISITA GRAFO
EDM-KARP O(VE²)
CASO PESSIMO
DIM
AUMENTI FLU
NUM TOT O(VE)
CRITICI O(V) VOLTE
O(E) ARCHI
MONOTONIA δf(s,v) ≥ δf(s,v)
f' = f+g
δf(s,v) DIST MIN s,v IN RETE RESIDUA Gf
TH
O(VE) AUMENTI FLU
RICHIEDONO O(V+E) BFS
O(VE²) ⇐
PRENDO IL + BASSO
ENTRAMBI LIM SUP VALIDI
CORRETTEZZA
DEFINIZIONI
TAGLIO tc PARTIZIONE V IN S e T
s∈S ∧ t∈T
S = V − T
Σ u∈S, v∈T f(u,v) = F(S,T) = FLUSSO ATT TAGLIO
Σ u∈S, v∈T c(u,v) = C(S,T) = CAPACITÀ ATTRAVERSA TAGLIO
DIMOSTRAZ
LEMMA
VALORE FLUSSO TAGLIO F(S,T) = |f|
DIMOSTRAZIONE = Σ u∈S, v∈T f(u,v)
Σ u∈S, v∈V f(u,v) − Σ u∈S, v∈S f(u,v)
(T = V − S)
0 (ANTISIMMETRIA)
|f| = Σ v∈S−{s}, v∈V f(u,v) + Σ v∈V f(s,v) = Σ u∈S, v∈V f(u,v)
CAPACITÀ TAGLIO
FLUSSO MAX LIM SUP
TAGLIO MIN
F(S,T) ≤ C(S,T) DIMOSTRAZ
|f| ≤ C(S,T) ⇐ |f| = F(S,T)
TEOREMA
1 f = FLUSSO MASSIMO
2 ∄ AUMENTANTE Gf
3 C(S,T) = |f| tc ∃ TAGLIO MIN
DIMOSTRAZ TEOREMA
1→2 SE ESISTESSE
∃ CAMMINO s→t IN Gf
FLUSSO PUÒ AUMENTARE ⇒ ASSURDO
2→3
f(u,v) = c(u,v)
|f| = Σ f(u,v)
|f| = C(S,T) ⇒ (S,T) È MINIMO
3→1
∀f È UN TAGLIO (S,T)
VALE |f| ≤ C(S,T)
È MASSIMO

BACKTRACK

INTRO
- TIPI PB
  - ENUMERAZIONE
  - CONTEGGIO
  - RICERCA
  - OTTIMIZZAZIONE
- IDEA
  - TORNARE INDIETRO
    - RICORSIVO
    - ITERATIVO

ENUMER
- ORG GENERALE
  - VETTORE SCELTE S[1..n]
  - INSIEME SCELTE
- SCHEMA PARZIALE
  - INPUT
    - DATI PB
    - ITEM[] S
    - DATI PARZIALI
  - VERIFICA
    - CASO BASE
    - CASO RICORSIVO
- ALBERO DECISIONI
  - È SPAZIO DI RICERCA
  - RADICE → SOL PARZ VUOTA
  - NODI INTERNI → SOLUZ PARZ
  - FOGLIE → SOLUZ AMMIS

SOTTOINSIEMI
- RICORSIVO
  - SET C = {0,1}
  - ∀c ∈ C
  - S[i] = c
  - SUBSETREC(..., i+1)
  - Θ(n · 2^n)
- ITERATIVO
  - BITWISE AND &&
  - PRINTING
  - Θ(n · 2^n)

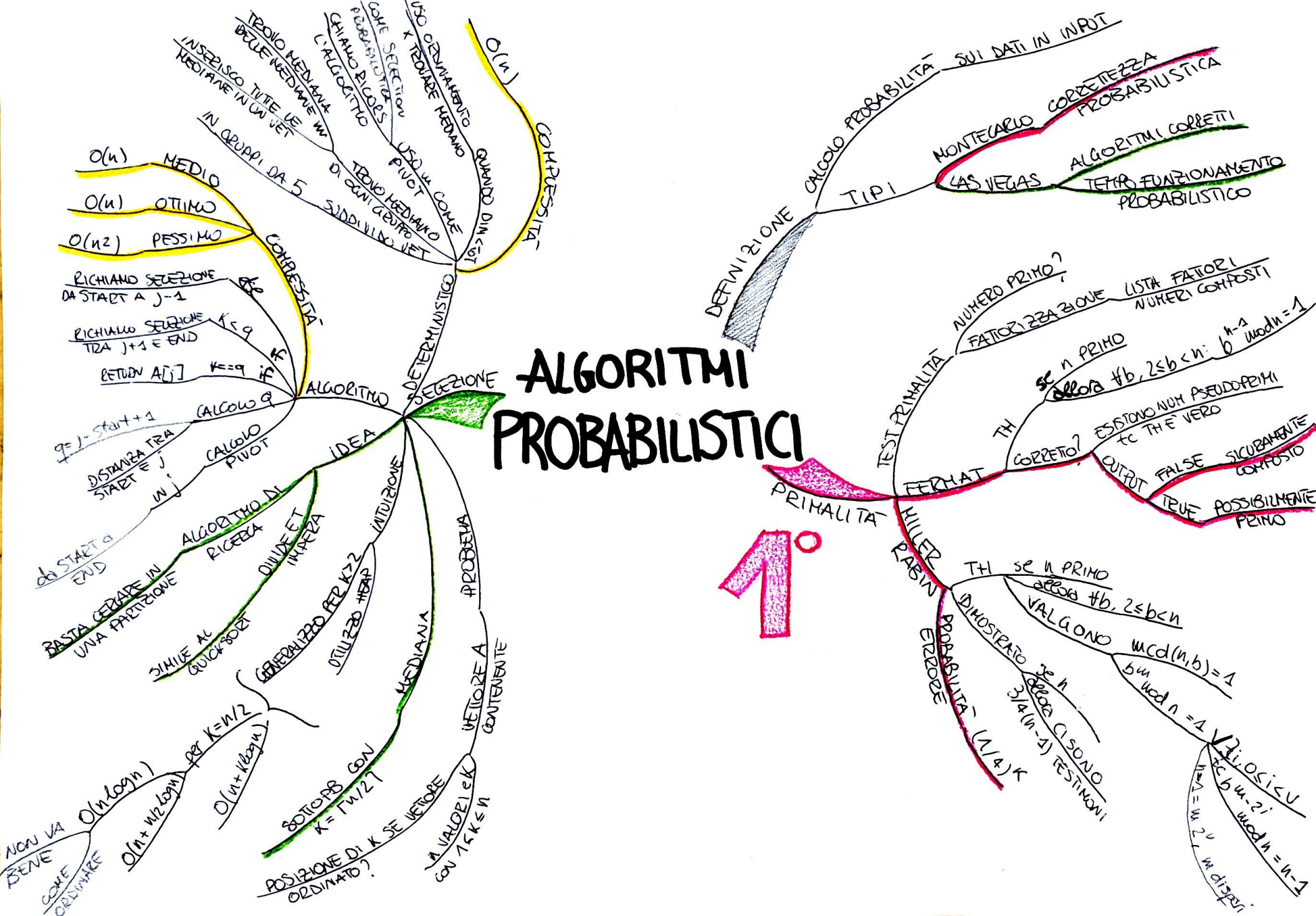
PERMUTAZIONI
- MODO 1
  - C = COPY(A)
  - ∀c ∈ C
    - A.INSERT(c)
    - CHIAMATA REC
    - A.REMOVE(c)
    - S[i] = c
  - STAMPA PERM DI INSIEME A
- MODO 2
  - for j = 1 to i
    - SWAP(S,i,j)
    - PERMREC(S,i-1)
    - SWAP(S,i,j)
  - STAMPA PERM VETTORE S
- O(n²n!)
  - O(n²) COPIE VETTORE
  - n! NUM FOGLIE

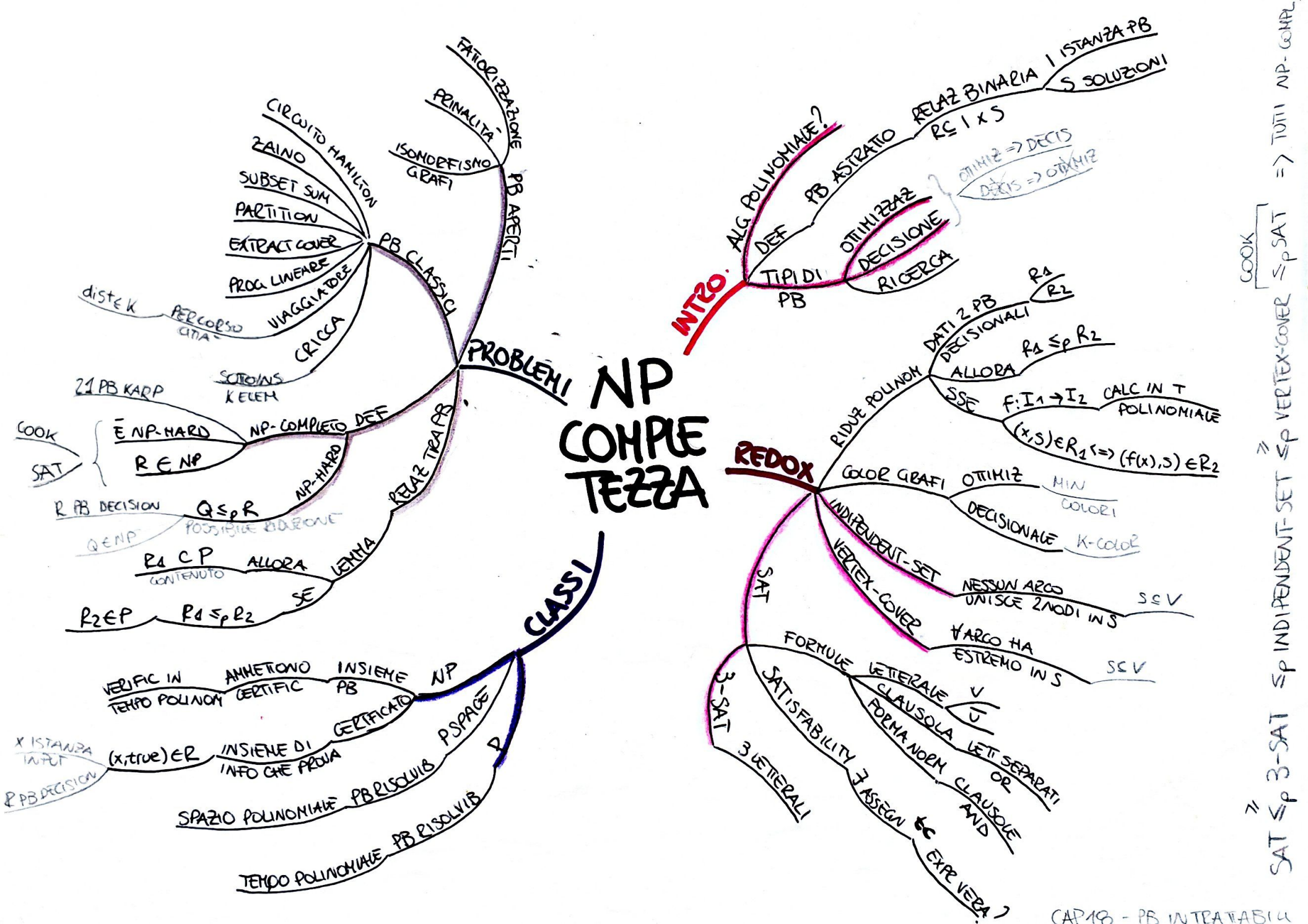
K-SOTTOINSIEMI
- CHIAMATA RICORSIVA S[i] = 0
- ∀c ∈ {0,1}
- i ≤ n AND 0 < miss ≤ n-(i-1)
- PRUNED IF CLEAN
- MISSING = 0
- PROCESSOL(S,i-1)

SUBSET-SUM
- O(2^n) COMPLESSITÀ
- A, n, Missing - A[i]. c
- S, i+1
- OK CHIAM REC IF ∀c ∈ {0,1}
- TRUE
- S[i] = c
- FALSE
- i ≥ n OR MISS ≤ 0
- VAL TERMINATI / SOMMA ECCESSIVA
- MISSING = 0
- STAMA INDICI SOLUZ
- TRUE
- PROCSOL(S,i-1)

INVILUPPO CONV
- O(n log n) COMPLESSITÀ
- GRAHAM
  - ORDINO i PUNTI
  - UTILIZZO PILA
  - RETTA TRA Pi e Pi-1
  - PRIMO E PENULTIMO VERTICE
  - Pi E Pi STESSO LATO ?
    - SI
      - TOGLI Pj
      - AGGIUNGI Pi
    - NO
- ALGORITMO GENERICO
  - PB
    - n PUNTI
    - MIN POLIGONO CONTIENE TUTTI
  - UTILIZZA RETTA
    - SE TUTTI n-2 PUNTI STESSO LATO
    - BORDO

ALGORITMI PROBABILISTICI




NP COMPLETEZZA
INTRO
ALG POLINOMIALE?
DEF
PB ASTRATTO
RELAZ BINARIA
I ISTANZA PB
S SOLUZIONI
R ⊆ I x S
TIPI DI PB
OTTIMIZZAZ
DECISIONE
RICERCA
OTTIMIZ ⇒ DECIS
DECIS ⇒ OTTIMIZ
PROBLEMI
PB APERTI
FATTORIZZAZIONE
PRIMALITÀ
ISOMORFISMO GRAFI
PB CLASSICI
CIRCUITO HAMILTON
ZAINO
SUBSET SUM
PARTITION
EXTRACT COVER
PROG LINEARE
VIAGGIATORE
CRICCA
dist ≤ K
PERCORSO CITTÀ
SOTTOINS K ELEM
21 PB KARP
COOK
SAT
È NP-HARD
R ∈ NP
NP-COMPLETO
DEF
NP-HARD
R PB DECISION
Q ≤p R
POSSIBILE RIDUZIONE
Q ∈ NP
R1 ⊆ P
CONTENUTO
ALLORA
LEMMA
RELAZ TRA PB
SE
R2 ∈ P
R1 ≤p R2
CLASSI
VERIFIC IN TEMPO POLINOM
ANNETTONO CERTIFIC
INSIEME PB
NP
CERTIFICATO
X ISTANZA INPUT
R PB DECISION
(x, true) ∈ R
INSIEME DI INFO CHE PROVA
SPAZIO POLINOMIALE PB RISOLVIB
PSPACE
TEMPO POLINOMIALE PB RISOLVIB
P
REDOX
RIDUZ POLINOM
DATI 2 PB DECISIONALI
R1
R2
ALLORA
R1 ≤p R2
SSE
F: I1 → I2
CALC IN T POLINOMIALE
(x,s) ∈ R1 ⟺ (f(x), s) ∈ R2
COLOR GRAFI
OTTIMIZ
MIN COLORI
DECISIONALE
K-COLOR
INDIPENDENT-SET
VERTEX-COVER
NESSUN ARCO UNISCE 2 NODI IN S
S ⊆ V
∀ ARCO HA ESTREMO IN S
S ⊆ V
SAT
3-SAT
3 LETTERALI
FORMULE LETTERALE
V
V
CLAUSOLA LETT SEPARATI OR
FORMA NORM CLAUSOLE AND
SATISFABILITY ∃ ASSEGN & EXPR VERA?
COOK
SAT ≤p 3-SAT ≤p INDIPENDENT-SET ≤p VERTEX-COVER ≤p SAT ⇒ TUTTI NP-COMPL
CAP18 - PB INTRATTABILI

SOL PB INTRATTABILI

$$lb(d,S,i) = cost[i] + \left\lceil \frac{out + \sum_{h \notin S} transfer[h] + back}{2} \right\rceil$$

```
bbTsp(Item[] S, int cost, Set R, int n, int i)
BT choices = copy(R)
foreach c ∈ choices do
    S[i] = c
    R.remove(c)
    if i < n then
        {calcola out, back, e transfer[h] per ogni h ∈ R}
        int lb = cost[i] + ⌈(out + Σ_{h∈R} transfer[h] + back) /2⌉
        if lb < min(cost then
            bbTsp(S, cost + d[S[i-1]][S[i]], R, n, i + 1)
    else
        lb = cost + d[S[i]][S[1]]
        if lb < min(cost then
            minSol = S
            minCost = lb
    R.insert(c)
```

ALGORITMO TSP BRANCH & BOUND

- COST_i
  - $O(n)$ TRANSFER[i] = min $d[i][h]$ $h \notin S[2..i]$ — CITTÀ H DELLE n-i DA ATTRAVERSARE — ATTRAVERSAMENTO
  - $O(n)$ back = min $d[h][S[i]]$ $h \notin S$ — TORNARE $S[1]$
  - $O(n)$ out = min $d[S[i]][h]$ $h \notin S$ — USCIRE $S[i]$
  - $cost[i] = cost[i-1] + d[S[i-1]][S[i]]$ $i \geq 1$
- RITORNARE A $S[1]$ — $S[1..i] \rightarrow$ UNA $n-i-1$ CITTÀ — ATTRAVERSO $S[1..i]$ UNA CITTÀ
- AGGIORNO MIN VARIABILE GLOBALE — SE TROVO COSTO MINORE — if $lb <$ min cost — int $lb = lb()$ — $S[i] = c$ — $\forall c \in C$ SCHEMA
- PROCEDO CON SCELTA $i+1$ — if $i < n$
- ABBASSA TEMPO ESECUZIONE — NON MIGLIORA COMPLESSITÀ — LOWER BOUND
- IL + POSSIBILE COSTO SOLUZ OTTIMA — STIMA APPROSSIMA — lb (costi pd), S, i, (costi pd+1) — FUNZIONE — UPPER BOUND
- NUOVE SCELTE COSTO $\geq lb()$ — GARANTISCE TUTTE SOLUZ — DIMENTE ENUM SOLUZ — ROTARE
- $S[1..i]$ — DIPENDE DA SEQ SCELTE FATTE — RIPETERE SCELTE
- È LIMITE SUP X COSTO SOLUZ OTTIMA — COSTO min cost — BEST SOLUZ AMMISSIBILE min sol
- ENUMERATIONE (BACKTRACKING) — NON PORTANO A SOLUZ OTTIMA

$C(x') \leq SC(x^*)$ con $S \geq 2$ $\forall$ ALG $A(n)$-APPROX x TSP TH APPROSSIMABILITÀ

△ TSP

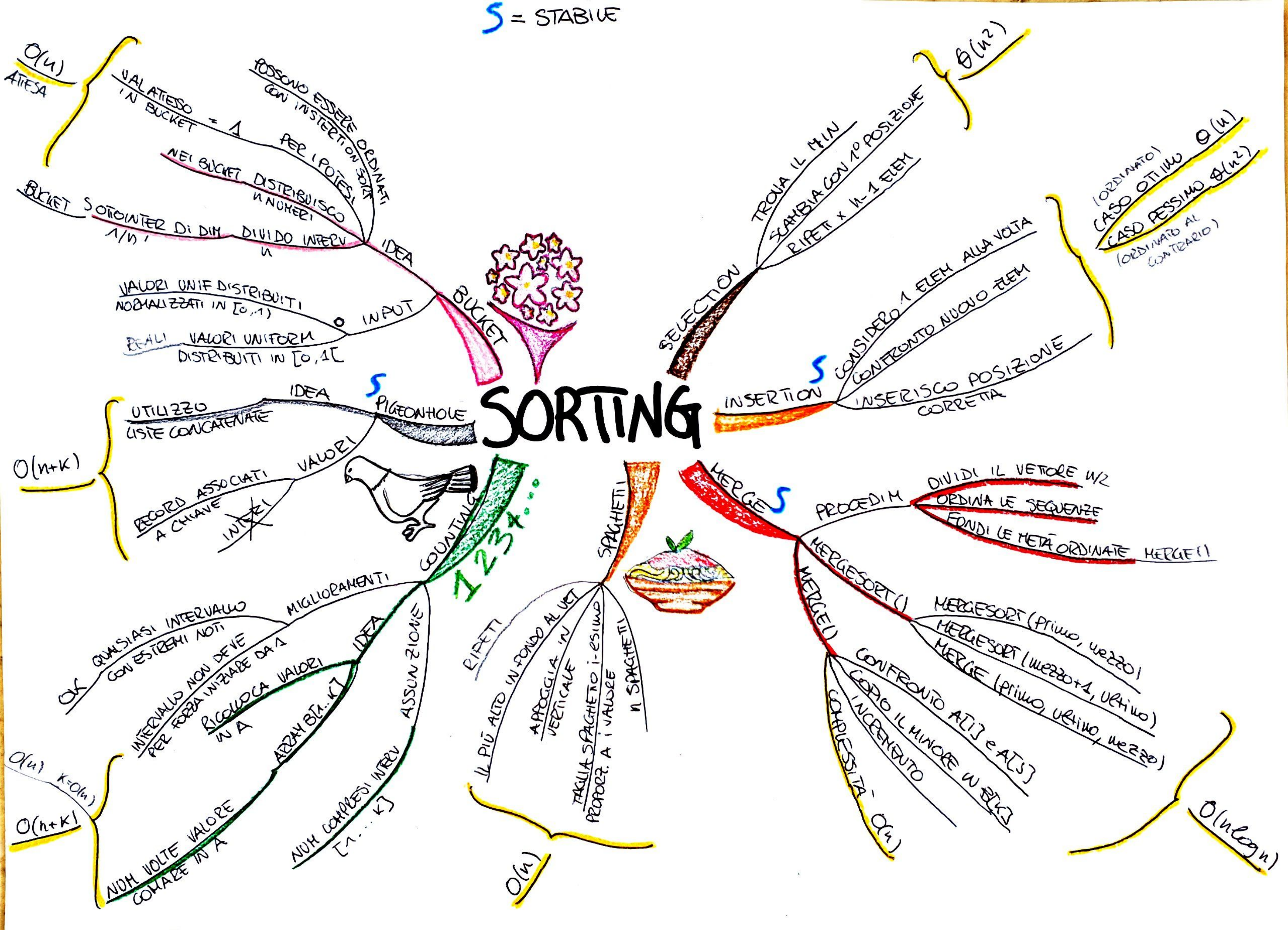
- ALGORITMO
  - COMPLESSITÀ
    - RADDOPPIATO CON 2n ARCHI — VISITA PROFONDITÀ $O(n)$ WST
    - ARCHI/GE — $O(n^2 \log n)$ $n = |A|$
  - NON È PRECISO
    - $A(n) = 2$
    - $C(m) \leq 2 \cdot wst \leq 2C(m^*)$
    - $C(m^*)$ COSTO CAT+HAM OTTIMO
    - DISUG TRIANG $C(m) \leq 2 \cdot wst$
    - EVITO CITTÀ VISITATE
    - PERCORRO ARCHI 2 VOLTE (ANCHE INDIETRO)
    - INDIVIDUO WST
    - DIMOSTRAZIONE
      - TOGLIAMO ARCO
      - SUPPONIAMO CHE GERMANO HAMILT
      - COSTO $C(m) \leq wst$
      - $wst < C(m)$
      - ASSURDO
    - $wst < C(m)$
    - $\triangle$ NON HAMILT DIST TOT $\geq wst$
    - $wst < C(m)$
    - ASSURDO $wst < C(m)$
  - IDEA
    - SFRUTTO CIRC HAMILTON
    - TROVATA
    - MST ALBERO MIN DI COPERTURA
    - CIRCUITO HAMILTONIANO DI COSTO $C(m)$
    - NP-COMPLETO
    - $\forall$ CITTÀ
    - $d[i,j] \leq d[i,k] + d[k,j]$
    - $2 \leq (i,j) \leq E$
    - $1 \leq (i,j) \leq E$
    - $d[i,j]$ DISTANZE
    - $d[i,j] \leq d[i,k] + d[k,j]$ VALGONO DISUG. TRIANGOLARI
    - $\forall i,j,k$ t.c. $1 \leq i,j,k \leq n$
    - $d[i,j] \leq 2 \leq d[i,k] + d[k,j]$ MIN 2
    - MAX 2
- APPROSSIMAZIONE
  - BIN PACKING
    - FIRST-FIT
      - $\exists$ ALG MIGLIORI
      - DIM
        - $N^* \geq \sum A[i] / K$ 1:1
        - $N \geq 4 \cdot$ NUM SCATOLE
        - NUMERO MIN SCATOLE
      - ASSEGNATO NELLA PRIMA SCATOLA CHE PUÒ CONTENERLO — CIASCUN OGGETTO — NUMERO NUMERO DI OGGETTI
      - N CAPACITÀ SCATOLA VETTORE (NUMERI OGGETTI) t.c. $A[i] \leq K$ $\forall i$
      - 2 SCATOLE PIENE x METÀ NUMERO/DI METÀ
      - $N \geq \sum A[i] / K \geq 2 \sum A[i] / K \geq 2N^* = \alpha(n) N^*$
      - USO AL PIÙ IL DOPPIO DELLE SCATOLE $\alpha(n) = 2$
  - DEF
    - $C(x) \leq \alpha(n) \cdot C(x^*)$ $\forall x$ ISTANZA
    - $C(x)$ COSTO SOL ALGORITMO
    - $C(x^*)$ COSTO SOL OTTIMA
    - $\alpha(n)$ FATTORE DI APPROX
    - $\alpha(n)$ COSTANTE — $\alpha(n)$ NUMERO > 1 $< +\infty$

PSEUDO-POLINOM

- DEF
  - RISOLVE PB R DATO L'INPUT
  - TEMPO $p(\#d)$ — # VAL MAX DATI INPUT — d DIMENSIONE
  - ALTRE DEF
    - NESSUN PB FORTEMENTE NP-COMP — A MENO CHE $P = NP$
    - FORTE NP-COMP — ROOT E NP-COMP — A PROBLEMA (LIK SOL $p(d)$)
    - DEBOLMENTE NP-COMPLETO — SE NP-COMPL NON FORTEMENTE
    - NB: NON È POLINOMIALE
    - BELLMEN $DP[n][K]$ — $\Theta(nK)$ — $K = \log \#$
    - SUBSET SUM
      - ESISTE SOTTOINS t.c. $\sum A[i] = K$?
      - $DP[i][r]$ — TRUE $r=0$ — FALSE $i<0 \wedge i=0$ — $i>0 \wedge i>0 \wedge A[i]>r$ — $DP[i-1][r]$ or $DP[i-1][r-A[i]]$ — $O(2^n)$
      - PROG DIN — $DP[i-1][r]$ — $ssrec(A,i-1,r)$ or $ssrec(A,i-1,r-A[i])$
      - BACKTRACKING — CASI BASE $A[i] > r$ — else $ssrec(A,i-1,r)$ — $O(nK)$ $O(2^n)$
      - MEMOIZATION — CASI BASE — DICTIONARY — RES — COPPIA INDICI — INSERISCO — RETURN RES
      - DEBOLMENTE NP-COMPL — $\forall A[i] \leq K$ se $K = O(n^c)$ — ASSURDO $\# = \max\{n, K, a_1 \dots a_n\} = O(n^c)$
      - PROG DIN $O(nK) = O(n^{c+1})$ IN P
      - NO FORT NP COMPL
      - ENTRAMBI VALGONO CONSIDERO IL + BASSO
- CRICCA
  - SOTTOINS K NODI — GRAFO NON ORIENT
  - FORTEMENTE NP-COMPL — $K \leq n$ — $\# = \max\{n, m, K\}$ — $d = O(n+m+\log \#) = O(n+m)$
  - PB RISTRETTO USUALE a CRICCA — NP-COMPL
- COMMESSO VIAGGIATORE
  - PERCORSO CITTÀ — DISTANZA MINIMA
  - FORTEMENTE NP-COMPL — GRAFO G HA CIRCUITO HAMILTON — SSE $\exists$ PERCORSO TSP DI COSTO N
  - SE ALG PSEUDOPOL x TSP — HAMILTON POTREBBE ESSERE RISOLTO IN POLINOM ASSURDO
- 3 PARTITION
  - PARTITION
    - VETTORE n INTERI POSITIVI — $\exists$ SOTTOINSIEME t.c. $\sum A[i] = \sum A[i]$?
    - FORTEMENTE NP-COMPLETO — DEBOLMENTE — RIDUCO A SUBSET SUM

EURISTICI

- DEF — SOLUZ CHE FUNZIONA — NON RIESCO A DEFINIRE LIMITE SUP
- TECNICHE POSSIBILI — GREEDY — RICERCA LOCALE
- TSP
  - GREEDY
    - SHORTEST EDGES FIRST
      - ORDINO ARCHI PESI CRESCENTI
      - AGGIUNGO ARCHI SOL — FINO A $n-1$ ARCHI — n NUMERO NODI
      - NEAREST NEIGHBOR — RAGGIUNGERE ARCO — VERIFICO — CIRCUITO — ANODO → 2 ARCHI
      - OTTENGO CATENA HAMILTONIANA — AGGIUNGO ARCO TRA ESTREMI
      - $O(n^2 \log n)$
    - NEAREST NEIGHBOR
      - PARTENZA DA CITTÀ
      - SCEGLI PROSSIMA LA + VICINA
      - EVITO CITTÀ VISITATE
      - TORNO CITTÀ INIZIALE
      - $O(n^2)$
  - RICERCA LOCALE
    - π CIRCUITO HAMILTON
    - INTORNO $I_k(\pi)$ = π' OTTENUTO DA π CANCELLO 2 ARCHI NON CONSECUTIVI SOSTITUISCO CON 2 ARCHI ESTERNI
    - COSTO
      - $O(n^2)$
      - DEVO CONSIDERARE TUTTE LE POSS COPPIE



S = STABILE
SORTING
BUCKET
IDEA
BUCKET SOTTOINTER DI DIM 1/n
DIVIDO INTERV IN n
NEI BUCKET DISTRIBUISCO n NUMERI
VAL ATTESO IN BUCKET = 1 PER IPOTESI
POSSONO ESSERE ORDINATI CON INSERTION SORT
O(n) ATTESA
INPUT
VALORI UNIF DISTRIBUITI NORMALIZZATI IN [0,1)
REALI VALORI UNIFORM DISTRIBUITI IN [0,1[
SELECTION
TROVA IL MIN
SCAMBIA CON 1° POSIZIONE
RIPETI x n-1 ELEM
Θ(n²)
INSERTION S
CONSIDERO 1 ELEM ALLA VOLTA
CONFRONTO NUOVO ELEM
INSERISCO POSIZIONE CORRETTA
(ORDINATO) CASO OTTIMO Θ(n)
CASO PESSIMO Θ(n²) (ORDINATO AL CONTRARIO)
MERGE S
PROCEDIM
DIVIDI IL VETTORE IN 2
ORDINA LE SEQUENZE
FONDI LE METÀ ORDINATE MERGE()
MERGESORT()
MERGESORT(primo, mezzo)
MERGESORT(mezzo+1, ultimo)
MERGE(primo, ultimo, mezzo)
MERGE()
CONFRONTO A[i] e A[j]
COPIO IL MINORE IN B[k]
INCREMENTO
COMPLESSITÀ O(n)
O(n log n)
SPAGHETTI
RIPETI
IL PIÙ ALTO IN FONDO AL VET
APPOGGIA IN VERTICALE
TAGLIA SPAGHETTO i-ESIMO PROPORZ. A i VALORE
n SPAGHETTI
O(n)
PIGEONHOLE S
IDEA
UTILIZZO LISTE CONCATENATE
VALORI
RECORD ASSOCIATI A CHIAVE
INTERI
O(n+K)
COUNTING
1 2 3 4 ...
MIGLIORAMENTI
OK QUALSIASI INTERVALLO CON ESTREMI NOTI
INTERVALLO NON DEVE PER FORZA INIZIARE DA 1
IDEA
RICOLLOCA VALORI IN A
ARRAY B[1..K]
NUM VOLTE VALORE COMPARE IN A
ASSUNZIONE
NUM COMPRESI INTERV [1...K]
O(n) K=O(n)
O(n+K)