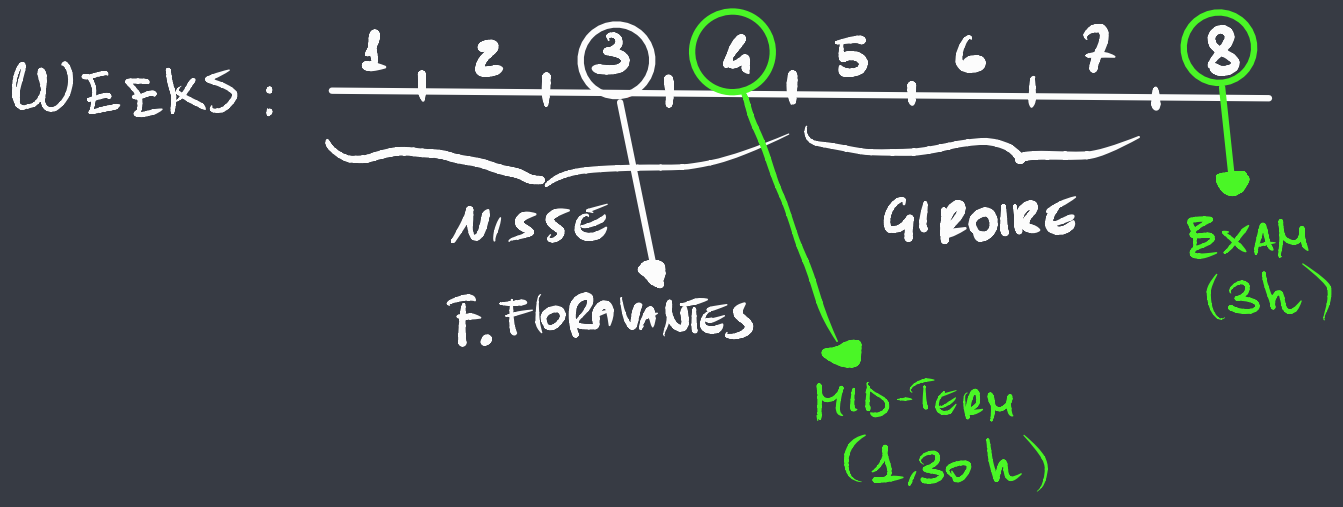
Graph algos



2021

INTRO TO GRAPHS

TH

- $T=(V,E)$ is a Tree $\iff$ T is CONNECTED and $|V| = |E| + 1$

PROOF ($\Leftarrow$)

- (RIC T is a TREE if CONNECTED and NO CYCLES)
- Suppose T not a tree $\Rightarrow \exists$ cycle $(v_1, \_, v_9) \Rightarrow$ we can so remove $\{v_1, v_9\} \in E$ obtaining T' connected
- $|E'| = |E| - 1 = |V| - 2 = |V'| - 2 < |V| - 1$

$\Rightarrow |E'| < |V'| - 1 \Rightarrow |V'| > |E'| + 1$ ⊥ $\Rightarrow T'$ NOT CONNECTED

($\Rightarrow$)

- INDUCTION ON $|V|$

PART 1

- let $P = (v_1, \_, v_9)$ be the longest PATH in T
- let v_1 be a LEAF
- suppose that $\deg(v_1) > 1$ AND $\exists\, x \in N(v_1) \setminus \{v_2\}$
  - $\Rightarrow x \notin P$ otherwise would create a cycle
  - $\Rightarrow$ so there should be $P' = (x, v_1, \_, v_9)$ but would be longer than P ⊥

PART 2

(cause it's connected!)

- let $S = T \setminus \{v_1\}$ be a subtree
- we know $|V_S| = |E_S| + 1$
- $|V| = |V_S| + 1$ AND $|E| = |E_S| + 1$

$$\Rightarrow \begin{cases} |V| = |V_S| + 1 \\ |E| = |E_S| + 1 \\ |V_S| = |E_S| + 1 \end{cases} \quad \begin{cases} |V_S| = |V| - 1 \\ |E_S| = |E| - 1 \\ |V| - 2 = |E| - 1 \end{cases} \Rightarrow |V| = |E| + 1$$

□

CONNECTIVITY AND DISTANCES

LENGHT of PATHS · sum of weights of arcs in PATH

DISTANCE · minimum lenght from among (s,t)-PATH

BFS (UNWEICHED GRAPHS)

INPUT $G=(V,E)$, $r \in V$

OUTPUT: $d(v) = d(r,v)$ $\forall v \in V$

ALGO

Horizon → is a QUEUE

$d(r) = \emptyset$, $d(v) = \infty$ $\forall v \in V \setminus \{r\}$

$Horizon = \{r\}$

$Explored = \{\emptyset\}$

$T = (V_T, E_T) = (\{r\}, \{\emptyset\})$

COMPLEXITY $O(|E|)$

WHILE $(Horizon \neq \{\emptyset\})$

- $v = head(Horizon)$
- FOR $(u \in N(v) \setminus \{Horizon \cup Explored\})$
  - $d(u) = d(v) + 1$
  - $V_T = V_T \cup u$
  - $E_T = E_T \cup (u,v)$
  - addInTail$(Horizon, u)$
- $Horizon = Horizon \setminus \{v\}$
- $Explored = Explored \cup \{v\}$

NOTE 1

T is the shortest path tree rooted in r

NOTE 2

We can check connectivity

G connected ⇔ $d(r,v) < \infty$ $\forall v \in V$

DIAMETER OF A GRAPH

- **INPUT** { UNWEIGHTED TREE $T=(V,E)$; $r \in V$
- **OUTPUT** $\text{diam}(G) = \max_{\forall u,v \in V} d(v,u)$

ALGO

- BFS(r)
- let u be such that $d(r,u) = \max_{v \in V} d(r,v)$
- BFS(u)
- let w be such that $d(u,w) = \max_{v \in V} d(u,v)$

RETURN $\text{diam}(G) = d(u,w)$

COMPLEXITY: 2 BFS

DIJKSTRA (WEIGHTED GRAPHS)

NOTE BFS doesn't work on WEIGHTED GRAPHS

- **INPUT** $G=(V,E)$, w positive weights, $r \in V$ — **OUTPUT** $d(v) = d(r,v)\ \forall v \in V$

ALGO

$d(r)=0$, $d(v)=\infty\ \forall v \in V \setminus \{r\}$
$T=(V_T, E_T) = (\{\emptyset\}, \{\emptyset\})$
EXPLORED $= \{\emptyset\}$
PARENT$(v) = \{\emptyset\}\ \forall v \in V$
WHILE (EXPLORED $\neq V$)

- let $v \in V \setminus$ EXPLORED be the one with min. $d(v)$
- $V_T = V_T \cup \{v\}$
- $E_T = E_T \cup \{(v, \text{Parent}(v))\}$
- $S = S \cup \{v\}$
- FOR ($u \in N(v) \setminus$ EXPLORED)
  - IF ($d(u) > d(v) + w_{vu}$)
    - $d(u) = d(v) + w_{vu}$
    - $\text{Parent}(u) = v$

COMPLEXITY WITH MIN HEAP: $O(|E| + |V| \log |V|)$

NOTE 1: T is the shortest path tree rooted in r

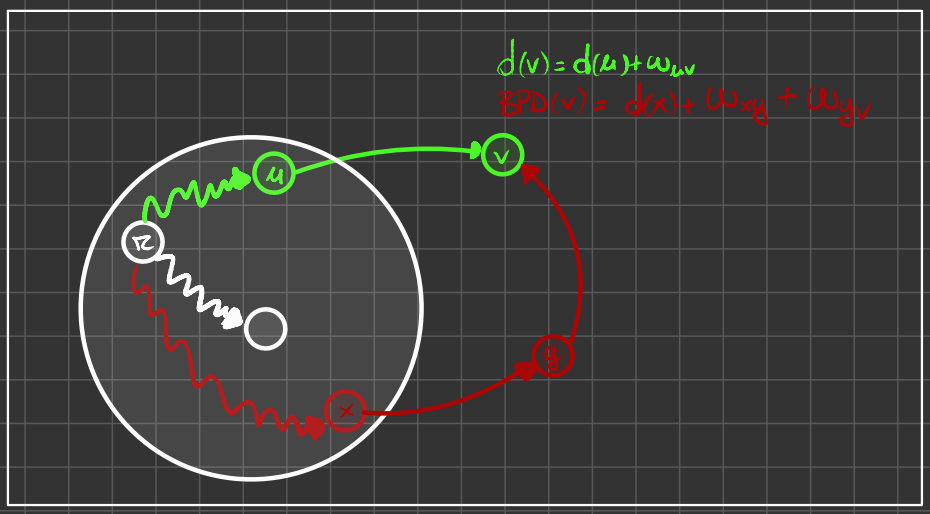
NOTE 2: We can check connectivity: G connected $\Leftrightarrow d(r,v) < \infty\ \forall v \in V$

TERMINATION

- After i^{TH} iteration $|EXPLORED| = i$ so it terminates in $|V|$ iterations

CORRECTNESS

- TH $d(v)$ is optimal when v is added to EXPLORED
- PROOF (By induction on iters.)
  - BASE: for $i=1 \Rightarrow d(s) = BPD(s) = \emptyset$ ✓
  - Assume, at the start of i^{TH} STEP, $d(u) = BPD(u)$ $\forall u \in EXPLORED$
  - let $d(v) = d(u) + w_{uv}$ the best choice of i^{TH} STEP
  - * it's TRIVIAL that, at STEP i, $d(v) < d(z)$ $\forall z$ not yet explored by the GREEDY CHOICE of the ALGO
  - Assume now that there's a shortest path $P = (s, \_, x, y, v)$ of lenght $BPD(v)$ | $BPD(v) < d(v)$



- So

$$\begin{cases} \textcircled{1}\ d(v) > BPD(v) \\ \textcircled{2}\ d(x) + w_{xy} \leq BPD(v) \\ \textcircled{3}\ d(y) \leq d(x) + w_{xy} \\ \textcircled{4}\ d(y) \geq d(v) \end{cases} \quad * \quad \begin{cases} \textcircled{1}\ d(v) > BPD(v) \\ \textcircled{2,3}\ d(y) \leq BPD(v) \\ \textcircled{4}\ d(y) \geq d(v) \end{cases} \quad \begin{cases} \textcircled{1,(2,3)}\ d(y) \leq BPD(v) < d(v) \\ \textcircled{4}\ d(y) \geq d(v) \end{cases}$$

$$\Rightarrow \begin{cases} d(y) < d(v) \\ d(y) \geq d(v) \end{cases} \Rightarrow d(v) \leq d(y) < d(v) \Rightarrow d(v) < d(v) \ \lightning$$

KRUSKAL (MST)

INPUT $G=(V,E)$ CONNECTED
w positive weight

OUTPUT MST $T=(V_T, E_T)$

COMPLEXITY $O(|E|\log|E|)$

ALGO

$T=(V, \{\phi\})$

Let $(e_1, \_, e_m)$ NON-DECR. order of edges

FOR($i=1, \_, m$)

  IF($E_T \cup e_i$ NOT create CYCLE)

    $E_T = E_T \cup e_i$

TERMINATE

- OBV.

CORRECT

- TH The spanning tree T returned is a MST
- Proof
  - T is obv. a SPANNING TREE, because G is connected, and T is acyclic by def.
  - Suppose by contradd. T NOT MST
  - So you have $e_i \in E$ such that by adding it to E_T, and removing another $e_j \in E$ you get $T' \mid w(T') < w(T)$
  - By the ordering e_j, that has already been added to E_T, should be such that $w(e_j) < w(e_i)$ so you get a CONTRADDICTION ↯

20/09/20

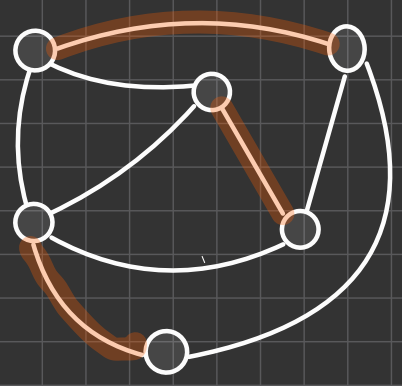
APPROXIMATION ALGO. FOR NP-HARD PROBLEM

MAX MATCHING (P)

MATCHING

- Given $G=(V,E)$ a matching in G is a partition of disjoint edges

$$M \subseteq E \mid e \cap f = \{\emptyset\}, \forall e,f \in M$$



max MATCHING PRB.

- $\exists$ Matching of size $\geq k \in \mathbb{N}$?

LEMMA

- $\forall G=(V,E)$ and any matching $M \Rightarrow |M| \leq |V|/2$

Proof

M-ALTERNATING PATH

- A path in G is M-ALTERNATING if for any 2 consecutive edges exactly one is in the matching M

AUGMENTING PATH

- $P=(e_1,\_,e_2)$ AUGMENTING if P is M-ALTERNATING $\wedge\ e_1 \notin M \wedge e_2 \notin M$

- if $\exists$ AUGMENTING PATH $P \Rightarrow M$ is not MAX

Proof

- suppose M max
- Build $M' := \{e \in E \mid e \in P \wedge e \in M\}$
- $|M'| = |M| + 1 \Rightarrow |M'| > |M| \Rightarrow M$ is not Max $\perp$

TH [BERGE']

- A matching M in G is maximum $\Leftrightarrow$ $\nexists$ an M-Augmenting Path P

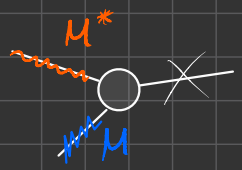
Proof

($\Rightarrow$) look at TH done before.

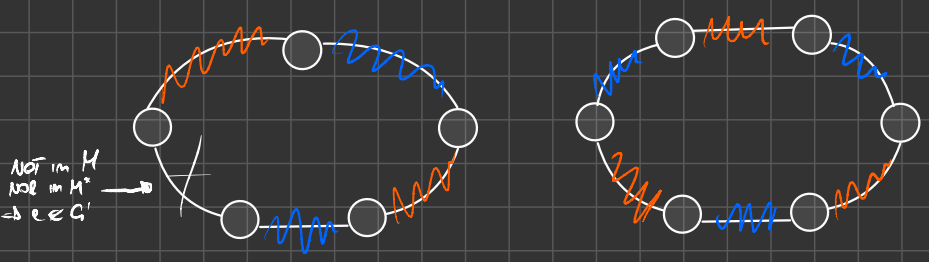
($\Leftarrow$)
- let M be a matching without augmenting Path
- let M^* be a maximum matching
- Consider $A := M \setminus M^*$ and $B := M^* \setminus M$
- Let $G' = (V, A \cup B)$ subgraph of G

- **CLAIM** every edge in G' has degree ≤ 2

Proof



- So in G graphs with $d \leq 2$ we have 2 cases:



$\Rightarrow |M^*| = |M|$

ALGO

Find a matching M

WHILE($\exists$ M-AUGMENTING PATH) // $n/2$ ITERS

| AUGMENT M // $O(n^3)$ TO FIND AUGM. ?

[EDMONDS] • We can find an M-AUGMENTING path in poly. time $O(n^3)$

$\Rightarrow$ We can so COMPUTE MAX-MATCHING IN Poly time

$$O(n^3) \cdot \underbrace{\frac{n}{2}}_{\text{BC LEMMA } |M| \leq \frac{|V|}{2}} = O(n^4)$$ COMPLEXITY

BIPARTITE GRAPH

- $G = (V, E)$ is BIPARTITE if $\exists$ a partition $V = V_1 + V_2 \mid V_1, V_2$ are STABLE (STABLE if no edges in same partition)

NOTE

- in BIP. GRAPH the maximum MATCHING can be solved efficiently

VERTEX COVER (NP COMPLETE)

VERTEX COVER

- Given $G=(V,E)$ a vert. cov. is $Q \subseteq V \mid i \in Q \vee s \in Q, \forall (i,s) \in E$

MIN Vert Cov PROB

- $\exists$ a Vert. Cover of size $\leq k \in \mathbb{N}$?

Sol

- Try all SUBSET $S \subseteq V$, check if it's a vertex Cover, Keep the one of minimum size.

$$O(\underbrace{2^{|V|}}_{\#\text{ subset of } V} \cdot \underbrace{|E|}_{\#\text{ Check if is a vertex cover}})$$

How can we solve NP-HARD efficiently? For example reducing the set of instances

TH[Konig] In BIPARTITE GRAPH, the size of a max MATCHING is equal to the size of maximum vertex cover

VERTEX COVER is P in bipartite graphs

- Let $m(G)$ the max. size of a matching in G
- Let $vc(G)$ the min size of a vert. cover in G

LEMMA $\forall G \; m(G) \leq vc(G) \leq 2m(G)$

ALGO

APPROX COVER

$M = \emptyset$

WHILE($\exists e \in E/M \mid M \cup \{e\}$ is a matching) | add e to M } COMPUTE A MAXIMAL MATCHING

RET $V(M)$ - vertices touched by $|M|$

- it's a GREEDY ALGO that calculate in COMPLEXITY $O(|E|)$ a vert. cover of size TWICE of the minimum vertex cover

$$|V(M)| = 2|M| \leq 2m(G) \leq 2vc(G)$$

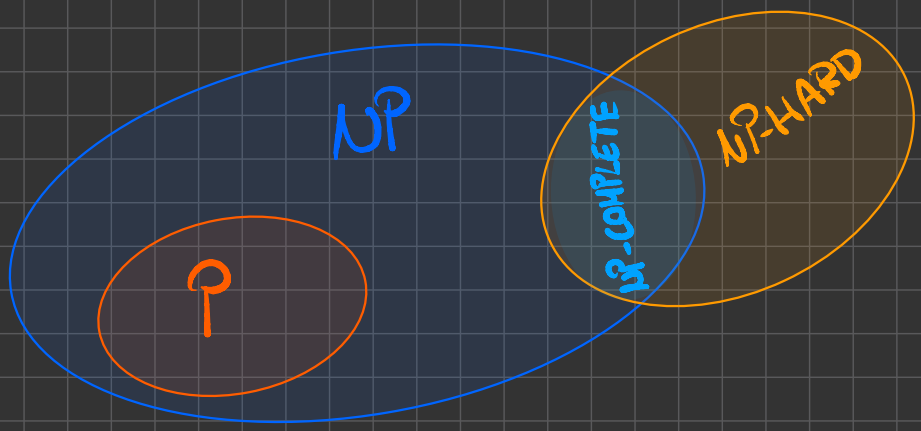
(HANDLE NP-HARD P) → C-APPROXIMATION ALGOS

P
- set of Probs. solvable in POLYNOMIAL TIME on a DETERMINISTIC TM.

NP
- set of Probs. solvable in POLYNOMIAL TIME on a NON-DETERMINISTIC TM, and their SOLs can be verified in POLYNOMIAL TIME on a DETERMINISTIC TM.

NP-HARD
- set of Probs. | $\forall \pi \in NP$ ∃ POLYNOMIAL TIME REDUCTION to any $\pi' \in$ NP-HARD, an NP-HARD prob can or cannot be in NP

NP-COMPLETE
- set of Probs. in $NP \cap$ NP-HARD | $\forall \pi \in NP$ ∃ Poly.T. REDUCTION to any $\pi' \in$ NP-COMPLETE



C-APPROX ALGOS
- WORKS in POLY TIME
- Finds a valid SOLUTION

Has a guaranty of the "SOLUTION QUALITY"

→ MIN $SOL \leq C \cdot OPT$, $C \geq 1$

→ MAX $SOL \geq C \cdot OPT$, $C \leq 1$

29/09/21

FLOW

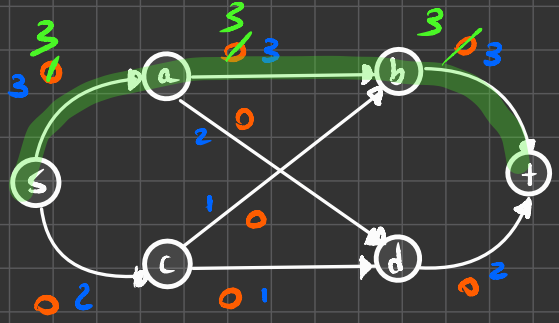
CAPACITY CONST. · $f_{ij} \leq u_{ij}$

CONSERVATION CONST. · $\sum_{u \in V \mid v \in N(u)} f_{uv} = \sum_{w \in N(v)} f_{vw} \quad \forall v \in V$

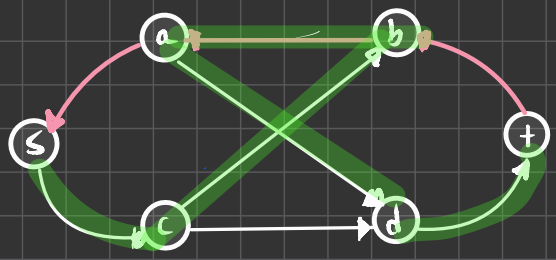
VALUE OF FLOW · $f(x)$

TH: MAX-FLOW MIN CUT TH (STRONG DUALITY) · Given a maximum flow X of G, it's value $f(x)$ it's equal to the value $f(R)$ of the min-cut(S)

FF. ALGO

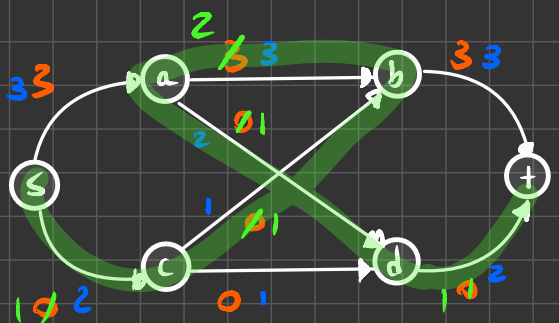


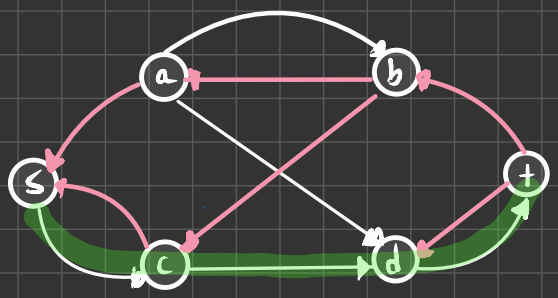
1) CHOOSE A STARTING FEASIBLE FLOW

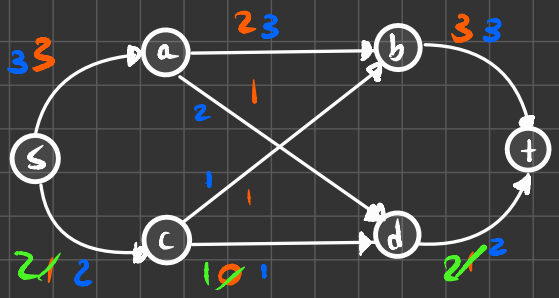


2) DRAW G'

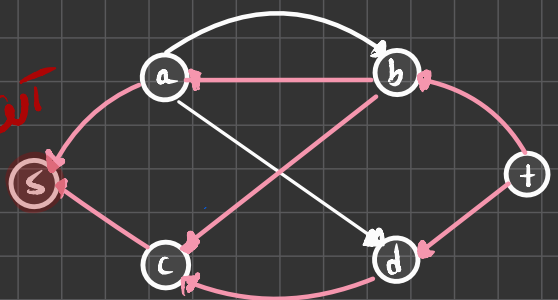
3) FIND ANOTHER PATH







CUT



$f(s,t) = \delta(s,t) = \{v \mid v$ reachable by s in $G'\}$

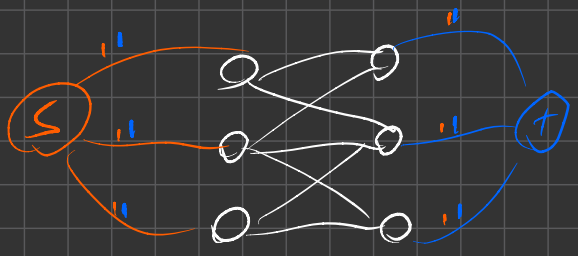
FF ALGO

- $O(f_{max} \cdot |E|)$ // In the worst case you can increment $f(x)$ of 1 by the path you find in G'. So you must build G' at most f_{max} times and apply DFS to find a path
  - f_{max}: # TIMES YOU BUILD G'
  - $|E|$: DFS

MENGER'S TH (1927)

- $\forall G=(V,E)$, $s,t \in V$ NON-ADJ, $k \in \mathbb{N}$ $\exists k$ INTERNALLY-VERTEX-DISJOINT PATHS FROM s TO t $\iff \nexists$ s,t SEPARATOR of SIZE $< k$.

MATCHING IN BIPARS. GRAPH W/ FF



HALL'S TH

- $\exists$ M saturating $A \iff |N(S)| \geq S$ $\forall S \subseteq A$

HUNGARIAN METHOD

- MAX MATCHING CAN BE FOUND IN $O(n^3)$