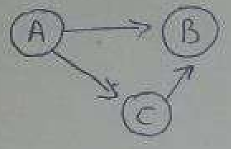
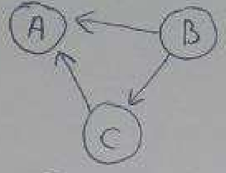
EXO 7

Problem 1 : Draw a few undirected graphs and a fewer directed graphs

1. Direcfld graph → transpose



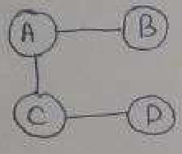
originar



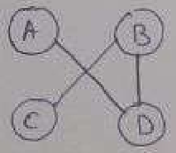
Transpose
reversed all arrow

Transposed graph = reverse all arrow directions. If orignal A → B, transpose: B → A
Transpose helps us find paths going beckwards. If you want to know: "who points to A", look at a transpose to see which vertices A points to.

2. Undirected graph → inverse



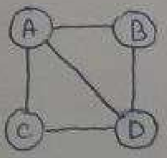
original



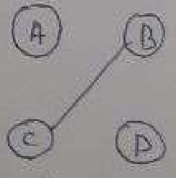
inverse

Inverse = complement, remove existing edges
add missing edges betwean all verticles
orginal A - B ⇒ NOT A - B.
orignall missing B-D ⇒ WILL B-D
Sometimes we want to find verticles that are NOT connected. Inresre shows us their clear

3. Dense graph → inverso



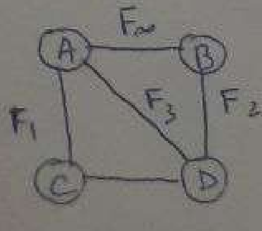
original (4 verticles, 5edges)



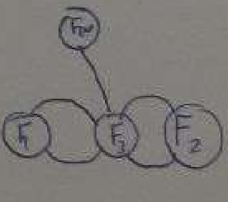
inverse

If original is dense (many edges), the inverse graph become sparce few edges.
Dense → Sparce
if original will have 5 verticlas → completed
Dense and sparce are opposito. If one is big other is small, They balance each other

4. Dual graph



original

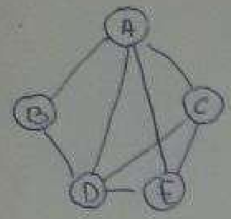


Dual (each face → vertex)

Dual graph: each face (region) become a vertex.
If two faces share an edge, connect their vertex
Face is region (area) in graph surronded by edge
like room with walls
Dual helpsus to solve problems about region by tening them into vertex problem

5. Why dual need plane graph?

non planer graph



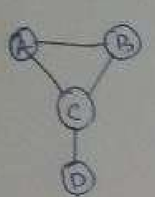
non planer cannot be be drown without edges crossing, so face are not well defined

dual graph only work for plan ar graphs where face are well-defines

when edges cross we cannot tell which region is which

Only planer graphs Can have dual graph. Non planer not.

Problem 2: Consider the pllowing undirected graph C with vertices $V = \{A, B, C, D\}$ and edges $E = \{AB, AC, BC, CD\}$



graph = { A: [B, C], B: [A, C], C: [A, B, D], D: [C] }

1. intiall call to the algorthm

- R (the curreet clique being built): [] - empty set.
- P: thet set of potential candidates to exted theclique: [A, B, C, D]
- X - the set of excluded verticles: []

2. First recursive call:

we need to try ecah vertex from P and mache recursive call to exted R.

we choose A from P

R = [A]
P = [B, C].
X = []

resursive 1 with A:

R = [A]
P = [B, C] ⇒ B:
X = []

R = [A, B]
P = [C]
X = [].

resursive 2 with A and B:

R = [A, B]
P = [C]
X = []

⇒ C:
R = [A, B, C]
P = [] no candida
X = []

⇒ The maximal clique: [A, B, C]

3. Second recursive. Call from anothe maxinal clique.

After we return from the first recursive branch, we move to other vertiles fom P.

After many operation all branch, we find maximum cliqup.

Maximal Clique of G:

[A, B, C]. - has 3 verticles.

[C, D].

This is simple explanation Bron-kerbosch algorithm works.