Προηγμένη Σχεδίαση Αλγορίθμων και Δομών Δεδομένων

2ο σύνολο ασκήσεων

Vlachos Dimitris

Άσκηση 1 (Ενημέρωση μέγιστης ροής)

Έστω $G = (V, E)$ ένα δίκτυο ροής με αφετηριακό κόμβο s και τερματικό κόμβο t, όπου οι χωρητικότητες των ακμών είναι ακέραιες. Μας δίνεται μία μέγιστη ροή f στο G και ζητούμε έναν αποδοτικό τρόπο να την ανανεώσουμε στις παρακάτω δύο περιπτώσεις:

α. Όταν η χωρητικότητα μιας ακμής (x, y) αυξάνεται κατά 1 μονάδα.

β. Όταν η χωρητικότητα μιας ακμής (x, y) μειώνεται κατά 1 μονάδα.

Δώστε αποδοτικούς αλγορίθμους που να ενημερώνουν τη μέγιστη ροή του δικτύου για κάθε μία από τις παραπάνω περιπτώσεις.

Υπόδειξη: Και στις δύο περιπτώσεις η ενημέρωση της μέγιστης ροής είναι εφικτή σε χρόνο $O(|V| + |E|)$.

(α) Η μέγιστη ροή του γράφου G(V,E) θα αυξηθεί κατά 1 μονάδα το μέγιστο. (Δηλαδή μπορεί να μην αυξηθεί καθόλου!)

Για να δούμε τι από τα 2 ισχύει μπορούμε να κάνουμε χρήση Αλγορίθμου με **augmented paths**.

Λογική:

> **Η μέγιστη ροή μπορεί να μην αυξηθεί καθόλου** λόγω κάποιου περιορισμού που μπορεί να υπάρξει από την “δομή”(ας πούμε ακολουθία από capacities) του δεδομένου γράφου. Μπορούμε να τσεκάρουμε αν residual (από το residual graph - υπολειπόμενο δίκτυο) χωρητικότητα από το x στο y ειναι ≥ 0 πριν την αύξηση της χωρητικότητας της (x,y).

>**(1)** Αν η υπολειπόμενη χωρητικότητα από το x στο y = 0, τώρα θα γίνει 1!

>**(2)** Τώρα θα προσπαθήσουμε να στείλουμε μια επιπλέον μονάδα πληροφορίας ξέκινώντας από τον αφετηριακό κόμβο s, φτάνοντας στον (χ,y) και καταλήγωντας στον τερματικό t. Αυτό μπορεί να γίνει τάχυτατα με χρήση του **DFS**. Δουλεύουμε στον υπολειπόμενο(residual) γράφο πάντα.

>**(2.1)** Τσεκάρουμε αν υπάρχει μια μονάδα επιπλέον χωρητικότητας από τον αφετηριακό **s** στον **x**.
>**(2.2)** Επίσης, Τσεκάρουμε αν υπάρχει μια μονάδα επιπλέον χωρητικότητας από τον αφετηριακό **x** στον **t**.

>**(3)** Αν (2.1) και (2.2) μπορούν να ικανοποιηθούν τότε μπορούμε να αυξήσουμε κατά μια μονάδα την χωρητικότητα(capacity) του μονοπατιού από τον source στον sink. Διαφορετικά δεν μπορούμε.

> Time Complexity = **O(|V| + |E|)** εφόσον ψάξαμε για 2 μονοπάτια μέσω DFS.

(β) Η μέγιστη ροή του γράφου G(V,E) θα μειωθεί κατά 1 μονάδα το μέγιστο.

Λογική:

>**(1)** Ελέγχουμε αν η (μέγιστη) ροή από την ακμή (x, y) == χωρητικότητά της πριν την μείωση κατά μια μονάδα ή < χωρητικότητα.
Αν ήταν < χωρητικότητα η μέγιστη ροή προφανώς δεν μπορεί να μεταβληθεί.

>**(2)** Στην 2η παραπάνω περίπτωση θα μειώσουμε την ροή κατά μια μονάδα και θα τσεκάρουμε αν μπορεί να βρεθεί κάποιο μονοπάτι στο υπολειπόμενο δίκτυο(residual graph) ώστε να μην "χάσουμε" την μονάδα της πληροφορίας βρίσκοντας μια νέα αυξητική διαδρομή(augmenting path)
Κάνουμε χρήση **DFS** πάλι.

>**(3)** Αν δεν καταφέρουμε να βρούμε μονοπάτι, τότε η μέγιστη ροή θα μειωθεί κατά μια μονάδα. (Διαφορετικά θα μείνει ίδια.)

> Time Complexity = **O(|V| + |E|)**

Άσκηση 2 (Μέγιστη ροή με χωρητικότητες κόμβων)

α. Έστω $G = (V, E)$ ένα δίκτυο ροής με αφετηριακό κόμβο s και τερματικό κόμβο t, όπου κάθε ακμή έχει θετική ακέραιη χωρητικότητα $c : E \rightarrow \mathbb{Z}^+$, και επιπλέον κάθε κόμβος $v \in V$ έχει μια θετική ακέραιη χωρητικότητα $\hat{c}_v$. Η ροή $f^{in}(v)$ που περνά από ένα κόμβο v ορίζεται ως το άθροισμα των ροών των ακμών που εισέρχονται στον v. Μια ροή f στο G ικανοποιεί επιπλέον τον περιορισμό χωρητικότητας κόμβων, δηλαδή για κάθε κόμβο $v \in V$ ισχύει ότι $f^{in}(v) \leq \hat{c}_v$. Περιγράψτε ένα αποδοτικό αλγόριθμο για τον υπολογισμό της μέγιστης ροής του G. *Υπόδειξη: Μπορεί να γίνει με αναγωγή στο τυπικό πρόβλημα μέγιστης ροής. Δημιουργήστε ένα βοηθητικό δίκτυο ροής το οποίο περιλαμβάνει επιπρόσθετους κόμβους και ακμές.*

β. Μας δίνεται ένα μη κατευθυνόμενο γράφημα $G = (V, E)$ με ένα αφετηριακό κόμβο s και ένα τερματικό κόμβο t. Σχεδιάστε ένα αποδοτικό αλγόριθμο που να υπολογίζει ένα απλό κύκλο που να περιέχει τους κόμβους s και t αν υπάρχει, ή να αναφέρει ότι δεν υπάρχει κανένας τέτοιος κύκλος στο G. Ποιος είναι ο χρόνος εκτέλεσης του αλγορίθμου σας; *Υπόδειξη: Μπορείτε να εφαρμόσετε τον αλγόριθμο του (a).*

(a)
> Βασικά έχουμε να κάνουμε με μια παραλλαγή του maximum flow που έχουμε δει.

> Στην περίπτωσή μας δεν έχουν μόνο ακμές χωρητικότητα αλλά και οι κόμβοι.

> Μπορούμε να μετατρέψουμε το flow network **G(V,E)** που μας δώθηκε ως είσοδο σε ένα κλασικό flow network **G'(V',E')** όπως μας προτάθηκε στην υπόδειξη. Δηλαδή δεν θα έχουμε κόμβους με χωρητικότητα στο G'.

> Τέλος θα μπορέσουμε να βρούμε τη μέγιστη ροή χρησιμοποιώντας έναν από τους γνωστούς Αλγορίθμους.

> Ακολουθεί αναλυτική περιγραφή του Αλγοριθμου:

----

(1)

> **for**(∀ node u in V){

• **χώρισε** u σε u_{input} & u_{out} (εκτός των source & sink)

• **σύνδεση** u_{input} & u_{out} σε 1 edge(u_{input}, u_{out}) με capacity c_u (να χρησιμοποιείται του u)

}

> **for**(∀ edge (u,v) in E του G){

• **άλλαξε** την (u,v) σε (u_{out}, v_{input})

• **διατήρησε** την capacity ∀ ακμής c(u,v)

> • Αν u είναι ο source, τότε θέσε s_{out} ως το νέο source στο G'

• Ομοίως, αν u είναι ο sink, τότε θέσε t_{input} ως το νέο sink στο G'.

(2)

> Input: G'(V', E')

• Κάνε χρήση ενός γνωστού Αλγορίθμου για το maximum flow problem (Ford-Fulkerson, Dinic...)

• Υπολόγισε το με source το u_{out} & sink τον v_{input}.

Συμπέρασμα: Η μέγιστη ροή του G' ⟺ Η μέγιστη ροή του G

(β)

> // find a path from s(source) to t(end-terminal)
  >Use DFS to find if there exists a path
  >if(it does not exist a path from s to t){
    print("A cycle does not exist);
    exit(1);
  }

> // find a path from t to s now
  >Starting from t, try to find a path back to s that does not reuse the edges in the path P from s to t.
  >We can mark the edges used in P (put them in a list) to avoid using them again in the new DFS.

> // Cycle detection
  If you find a path P' from t to s, then the combination of P & P' form a cycle containing s & t. <==> cycle ≡ P ∪ P'

```
> if( it does not exist path P'){
      print("A cycle does not exist");
      exit(1);
   }
> print("A cycle exists!");
```

Complexity: **O(|V| + |E|)** since we traverse the graph with DFS

Άσκηση 3 (Υπολογισμός μέγιστης ροής με κλιμάκωση χωρητικότητας)

Έστω $G = (V, E)$ ένα δίκτυο ροής με αφετηριακό κόμβο s και τερματικό κόμβο t, όπου κάθε ακμή έχει θετική ακέραιη χωρητικότητα $c : E \rightarrow \mathbb{Z}^+$. Έστω C η μέγιστη χωρητικότητα ακμής του G, δηλαδή $C = \max\{c(u, v) : (u, v) \in E\}$.

Ο παρακάτω αλγόριθμος υπολογίζει τη μέγιστη ροή όπως η μέθοδος Ford-Fulkerson, με τη διαφορά ότι χωρίζεται σε φάσεις ως εξής. Σε κάθε φάση του αλγορίθμου, προσπαθούμε να βρούμε αυξητικά μονοπάτια χωρητικότητας τουλάχιστον K. Αν δεν υπάρχει άλλο τέτοιο αυξητικό μονοπάτι, τότε υποδιπλασιάζουμε την τιμή του K και ξεκινάμε την επόμενη φάση.

Αλγόριθμος Υπολογισμού της Μέγιστης Ροής με Κλιμάκωση Χωρητικότητας

$C = \max\{c(u, v) : (u, v) \in E\}$
/* αρχικοποίηση: η ροή f είναι μηδενική σε κάθε ακμή */
for all $(u, v) \in E$ **do**
    $f(u, v) = 0$
end
$K = 2^{\lfloor \lg C \rfloor}$
while $K \geq 1$ **do** /* φάση του αλγορίθμου */
    while (υπάρχει αυξητική διαδρομή p με υπολειπόμενη χωρητικότητα $\geq K$) **do**
        αυξάνουμε τη ροή f κατά μήκος της διαδρομής p
    end
    $K = K/2$
end

α. Εξηγήστε γιατί η ελάχιστη τομή (και επομένως και η μέγιστη ροή) του G έχει τιμή το πολύ $C \cdot |E|$.

β. Περιγράψτε πως μπορούμε να υπολογίσουμε ένα αυξητικό μονοπάτι χωρητικότητας τουλάχιστον K (αν υπάρχει) σε $O(|E|)$ χρόνο.

γ. Δικαιολογήστε γιατί ο αλγόριθμος υπολογίζει στο τέλος μια μέγιστη ροή του G.

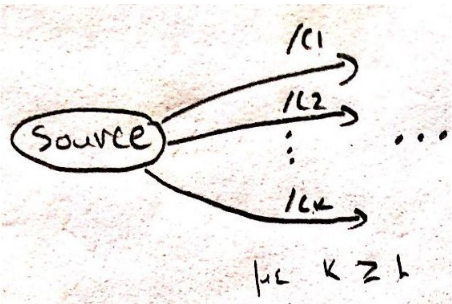
δ. Δείξτε ότι στην αρχή κάθε φάσης, η ελάχιστη τομή στο υπολειπόμενο δίκτυο G_f έχει τιμή το πολύ $2 \cdot K \cdot |E|$.

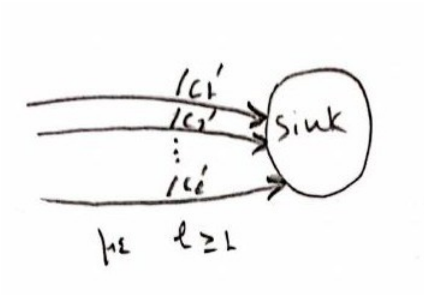
ε. Δείξτε ότι σε κάθε φάση μπορούν να εκτελεστούν το πολύ $O(|E|)$ υπολογισμοί αυξητικών διαδρομών.

στ. Τέλος, με βάση τα παραπάνω, προσδιορίστε τη συνολική πολυπλοκότητα χρόνου του αλγορίθμου.

(α)

> Προφανώς σε κάθε flow network η τιμή της μέγιστης ροής δεν μπορεί να υπερβεί το άθροισμα των χωρητικοτήτων όλων των ακμών που "εξέρχονται" από το source κόμβο. ⟺ Ίδια λογική για την χωρητικότητα των ακμών που "εισέρχονται" του κόμβου sink.





Σχόλιο: Το κ δεν έχει σχέση με το Κ του ερωτήματος (β)

> **Συνεπώς εφόσον έχουμε πλήθος ακμών = |E| η μέγιστη δυνατή ολική χωρητικότητα για κάθε δυνατή τομή θα είναι C*|E| (δηλαδή αυτό είναι το άνω φράγμα).**

> ⟺ Η συνολική χωρητικότητα θα είναι ≤ |E|*C Άρα η μέγιστη ροή του δικτύου δεν μπορεί να υπερβεί την C*|E|.

(β)

> Θα κάνουμε χρήση του **DFS** με τη μόνη διαφορά ότι θα ασχοληθούμε με τις ακμές με residual capacity ≥ K (Αυτές οι ακμές "δημιουργούν" το residual γράφο)

> **Διασχίζουμε το G με αφετηριακό κόμβο τον s χρησιμοποιώντας ακμές με residual capacity ≥ K.** (Δηλαδή αγνοούμε τις ακμές που δεν μπορουν να "ειναι κομμάτι" του augmenting path με την ζητούμενη χωρητικότητα)

> **Αν καταφέρουμε να φτάσουμε τον sink κόμβο t τότε προφανώς έχει βρεθεί augmenting μονοπάτι.**

> **Διαφορετικά δεν υπάρχει τέτοιο μονοπάτι.**

Time Complexity: **O(|E|)**

Γιατί κάθε ακμή θα χρησιμοποιηθεί το πολύ 2 φορές στη διερεύνηση.(Βασικά 1 φορα σε κάθε κατεύθυνση του residual graph)

(γ)

>Ο Αλγόριθμος που μας δώθηκε κάνει χρήση της **Capacity Scaling** που είναι περισσότερο μια **heuristic** τεχνική.

>Λογική: Θα δώσουμε προτεραιώτητα στις ακμές με μεγαλύτερες χωρητικότητες. Με αυτό τον τρόπο θα αποφύγουμε να καταλήγουμε σε path με μικρότερο **bottleneck**.

Algorithm():

> **C** = Η τιμή της μέγιστης χωρητικότητας ακμής στο flow network.

> **K** = Αρχικά η υψηλότερη δύναμη του 2 ≤ C

> **Capacity Scaling heuristic** = Παίρνουμε ακμές που η υπολοιπόμενη χωρητικότητα ≥ K ώστε να πετύχουμε καλύτερο χρόνο-που είναι το ζητούμενο.

> Η παράμετρος K δεν θα μειωθει κατά το μισό ενόσω βρίσκουμε augmenting paths με υπολοιπόμενη χωρητικότητα ≥ K

> Ύστερα K = K/2 & επαναλαμβάνουμε την διαδικασία όσο K > 0.

Σχόλιο: Το πως θα βρούμε augmenting paths μπορεί να αφεθεί στην ευχέρεια του προγραμματιστή.

(δ)

> Στην αρχή ∀ φάσης με δεδομένο το K, το G_f περιέχει ακμές που ακόμα έχουν ροή που αυξάνει.

> ∀ τομή σε αυτό το γράφο μπορεί το πολύ να έχει για ∀ ακμή "συμμετοχή" K, γιατί αν ∃ path με capacity > K θα είχε γίνει augmented στο προηγούμενο loop-φάση.(2K)

> Εφόσον ∃ |E| edges στη χείριστη περίπτωση ∀ ακμή στη τομή μπορεί να φτάσει το πολύ K, συνεπώς η max flow για ∀ τομή είναι **2*K*|E|** το πολύ.

(ε)

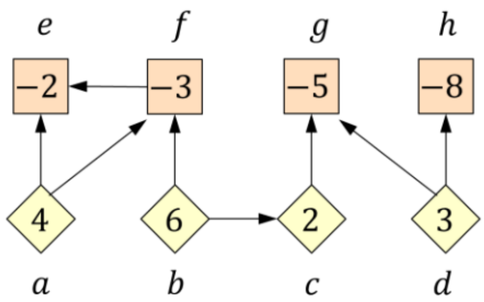
(e)
- Σε ∀ φάση-loop, προσθέτουμε flow ώστε να βρούμε το max flow μέσω augmenting paths.
- Η χωρητικότητα κάποιων ακμών (αν και δεν ο γράφος θα μπορούμε να προσδιορίσουμε ποιές ακριβώς) θα μειωθούν κατά K (τουλ.) στο residual graph.
- Δεδομένου ότι το άθροισμα των χωρητικοτήτων $\leq C*|E|$ αρχικά, μετά από ένα # φάσεων δεν θα μπορεί να υπάρξει μια ct.
- Έχοντας με μέγιστη ροή $\leq 2*K|E|$, (από ερώτημα δ) το πλήθος των μονοπατιών $\leq 2*|E|$
  Δηλαδή ∀ μονοπάτι με χωρητικότητα K μετά
  Το # των μονοπατιών ανά φάση θα μειώνεται & θα οδηγηθεί σε $O(|E|)$ συνολικώς για ∀ φάση

Άσκηση 4 (Επιλογή έργων)

Ας υποθέσουμε ότι μας δίνεται ένα σύνολο από n έργα που θα μπορούσαμε εκτελέσουμε. Ορισμένα έργα έχουν εξαρτήσεις, δηλαδή δεν μπορούν να ξεκινήσουν έως ότου ορισμένα άλλα έργα έχουν ολοκληρωθεί.

Τα έργα και οι εξαρτήσεις τους περιγράφονται από ένα κατευθυνόμενο άκυκλο γράφημα $G = (V, E)$, οι κόμβοι του οποίου είναι τα έργα, και κάθε ακμή (u, v) υποδηλώνει ότι το έργο u δεν μπορεί να ξεκινήσει χωρίς να έχει ολοκληρωθεί το έργο v. Επίσης, κάθε έργο v έχει ένα κέρδος $p(v)$ που θα μας δοθεί αν το ολοκληρώσουμε. Ορισμένα έργα έχουν αρνητικά κέρδη, τα οποία ερμηνεύουμε ως κόστος που πρέπει να πληρώσουμε για να ολοκληρωθούν.

Το παρακάτω σχήμα δίνει ένα παράδειγμα με 8 έργα, όπου τα a, b, c, d έχουν θετικό κέρδος, ενώ τα e, f, g, h έχουν αρνητικό κέρδος.



Μπορούμε να επιλέξουμε οποιοδήποτε υποσύνολο X των έργων, αρκεί να περιλαμβάνει όλα τα εξαρτώμενα έργα. Δηλαδή, για κάθε έργο $x \in X$, το σύνολο X θα πρέπει να περιέχει και κάθε έργο από το οποίο εξαρτάται το x. Στόχος μας είναι να βρούμε ένα έγκυρο υποσύνολο X των έργων με το μέγιστο δυνατό συνολικό κέρδος $P(X) = \sum_{x \in X} p(x)$. (Π.χ., εάν όλα τα έργα έχουν αρνητικό κέρδος, η σωστή απάντηση είναι να μην κάνουμε κανένα.)

Μπορούμε να επιλέξουμε οποιοδήποτε υποσύνολο X των έργων, αρκεί να περιλαμβάνει όλα τα εξαρτώμενα έργα. Δηλαδή, για κάθε έργο $x \in X$, το σύνολο X θα πρέπει να περιέχει και κάθε έργο από το οποίο εξαρτάται το x. Στόχος μας είναι να βρούμε ένα έγκυρο υποσύνολο X των έργων με το μέγιστο δυνατό συνολικό κέρδος $P(X) = \sum_{x \in X} p(x)$. (Π.χ., εάν όλα τα έργα έχουν αρνητικό κέρδος, η σωστή απάντηση είναι να μην κάνουμε κανένα.)

Για παράδειγμα, στο παραπάνω σχήμα, τα σύνολα $X_1 = \{a, e, f\}$ και $X_2 = \{a, b, c, e, f, g\}$ είναι εφικτά και έχουν κέρδος $P(X_1) = 4 - 2 - 3 = -1$ και $P(X_2) = 4 + 6 + 2 - 2 - 3 - 5 = 2$.

Περιγράψτε ένα αποδοτικό αλγόριθμο που να υπολογίζει μια βέλτιστη λύση για αυτό το πρόβλημα, δηλαδή να βρίσκει ένα έγκυρο υποσύνολο έργων με μέγιστο συνολικό κέρδος. Δικαιολογήστε την απάντηση σας. Επίσης, δείξτε πως εφαρμόζεται ο αλγόριθμός σας στο παράδειγμα του σχήματος.

Υπόδειξη: Δώστε μια αναγωγή στο πρόβλημα υπολογισμού της ελάχιστης s-t τομής ενός κατάλληλου δικτύου G'. Το δίκτυο κατασκευάζεται από το γράφημα των εξαρτήσεων G, τοποθετώντας ένα αφετηριακό κόμβο s με εξερχόμενες ακμές προς κάθε κόμβο v με θετικό κέρδος $p(v)$ και ένα τερματικό κόμβο t με εισερχόμενες ακμές από κάθε κόμβο v με αρνητικό κέρδος $p(v)$. Καθορίστε κατάλληλα τις χωρητικότητες των ακμών του G' και δείξτε πως μπορούμε να υπολογίσουμε τη βέλτιστη λύση στο πρόβλημα μας υπολογίζοντας την ελάχιστη s-t τομή του δικτύου.

> Σύμφωνα με την υπόδειξη δημιουργούμε 1 δίκτυο ροής G' από το G.
Προσθέτουμε 1 source s & 1 sink κόμβο t.

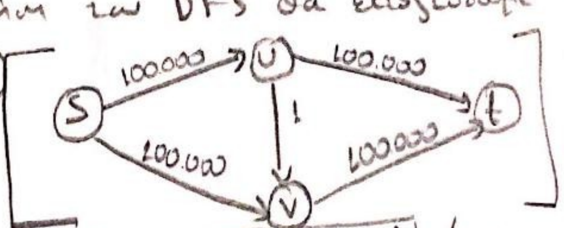
> Συνδέουμε τον s με τους κόμβους-έργα v με $p(v)>0$ με ακμές που έχουν <u>capacity</u> = με το κέρδος v κόμβου $p(v)$

> Συνδέουμε όλους τους κόμβους έργα με $p(v)<0$ με ακμές που έχουν <u>capacity</u> = $-$(με το κέρδος v κόμβου $p(v)$) ώστε να έχουμε θετικές ακμές.

> Όταν επιλέγουμε 1 έργο τότε πρέπει να διασφαλίσουμε ότι θα επιλεχθούν όλες οι εξαρτώμενες-ακμές του.

> Για να βρούμε την ελάχιστη τομή s-t στο G' χρησιμοποιούμε ένα γνωστό αλγόριθμο.
Έστω τον Edmond-Karp γιατί κάνει χρήση του BFS.
Έτσι δεν θα έχουμε το problem του Ford-Fulkerson όπου αν κάνουμε ατυχή επιλογή του path & με χρήση του DFS θα εκτοξεύουμε το κόστος

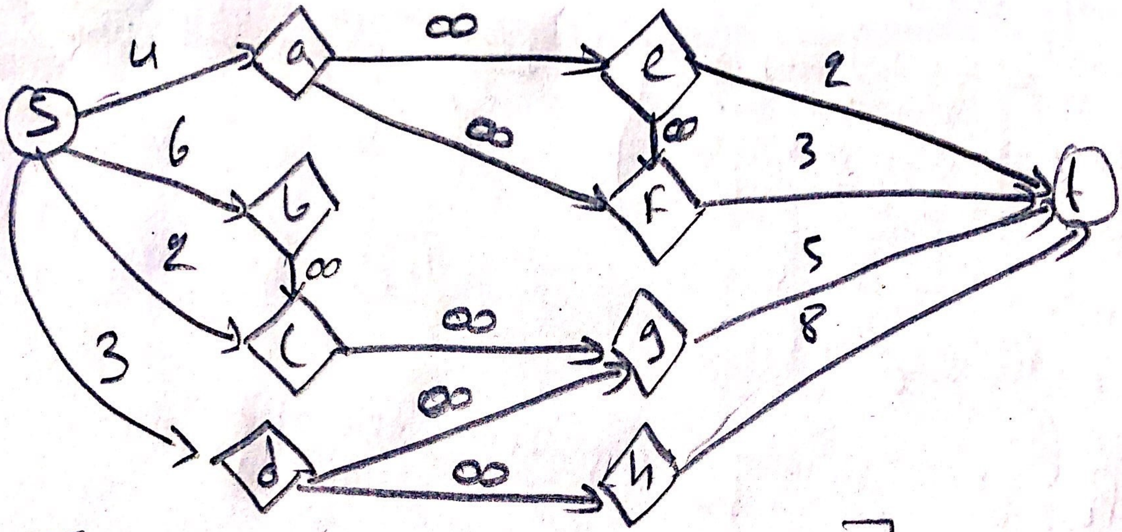
ΠΧ



<u>Total Time</u> = $\boxed{O(V * E^2)}$ (Αντίστοιχο του max flow)

> Η ελάχιστη τομή s-t ⟺ βέλτιστο υποσύνολο έργων.
Αυτό γιατί θα επιλέγουμε ακμές με θετικό κέρδος (θετικές ακμές) ώστε να μεγιστοποιήσουμε το συνολικό κόστος.

Συνεπώς το υποσύνολο έργων X που θα επιλεγεί θα αποτελεί την βέλτιστη λύση - μέγιστο κέρδος.



Το ∞ ουσιαστικά "δηλώνει" ότι αν
το ζεύγος είναι (u,v) θα υπάρχει
Αν u ∈ A ανήκει το u, θα ανήκει
και το v